DAFx23

Proceedings of the
26th International Conference on Digital Audio Effects

Aalborg University Copenhagen

September 4th - 7th, 2023

These proceedings have been created using pdfLaTeX

ISSN: 2413-6689

bibTeX:

```
@proceedings{dafx23Proceedings,
editor  = "Serafin, S. and Fontana, F. and Willemsen, S.",
title   = "Proc. of the 26th Int. Conf. on Digital Audio Effects (DAFx23)",
address = "Copenhagen, Denmark",
issn    = "2413-6689",
url     = "www.dafx.de",
year    = "2023"}
```

Volume editing

F. Fontana and S. Willemsen

Copyright

Organising Committee

Conference Chair

Stefania Serafin, Aalborg University Copenhagen

Paper Co-Chairs

Federico Fontana, University of Udine

Silvin Willemsen, Eindhoven University of Technology

Local Organizers

Ali Adjorlu, Aalborg University Copenhagen

Jesper Andersen, Royal Danish Academy of Music

Doga Buse Cavdir, Stanford University

Cumhur Erkut, Aalborg University Copenhagen

Francesco Ganis, Aalborg University Copenhagen

Prithvi Ravi Kantan, Aalborg University Copenhagen

Rolf Nordahl, Aalborg University Copenhagen

Daniel Overholt, Aalborg University Copenhagen

Razvan Paisa, Aalborg University Copenhagen

Anders Riddersholm Bargum, Aalborg University Copenhagen

Simon Rostami, Aalborg University Copenhagen

Thomas Albert Rushton, Aalborg University Copenhagen

Program Committee

Alexander Adami, IIS Fraunhofer Institute

Benoit Alary, IRCAM

Adrian Artacho, University of Music and Performing Arts Vienna

Federico Avanzini, University of Milan

Isabel Barbancho, Universidad of Málaga

Stefan Bayer, International Audiolabs Erlangen

Jose Ramon Beltran, University of Zaragoza

Gilberto Bernardes, University of Porto

Stefan Bilbao, The University of Edinburgh

Ivica Bukvic, Virginia Tech

Guilherme Campos, University of Aveiro

Sergio Canazza, University of Padova

Elliot Canfield-Dafilou, Institut Jean le Rond d'Alembert, Sorbonne Université/CNRS

F. Amílcar Cardoso, DEI, CISUC, University of Coimbra

Thibaut Carpentier, IRCAM

Jason Corey, University of Michigan

Gloria Dal Santo, Aalto University

Stefano D'Angelo, Orastron Srl

Roger Dannenberg, Carnegie Mellon University

Orchisama Das, Stanford University

Vincent Debut, CTN/IST/UTL Lisbon

Kristjan Dempwolf, PreSonus Software Inc.

Sascha Disch, Fraunhofer

Christian Dittmar, IIS Fraunhofer Institute

Maurício Do Vale Madeira da Costa, Federal University of Rio de Janeiro and Télécom-ParisTech/Université Paris-Saclay

Michele Ducceschi, University of Bologna

Lars Engeln, Technische Universität Dresden

Philippe Esling, IRCAM

Fabian Esqueda, Independent researcher

Gianpaolo Evangelista, University of Music and Performing Arts Vienna

Jon Fagerström, Aalto University

Christof Faller, Illusonic GmbH

Leonardo Fierro, Aalto University

Federico Fontana, University of Udine

Diamantino Freitas, University of Porto

Anastasia Georgaki, University of Athens

University of Padova

Jean-Louis Giavitto, IRCAM-CNRS

Dan Gillespie, Newfangled Audio Inc.

Bruno Gingras, Goldsmiths University of London

Joachim Heintz, HMTM Hannover

Robert Hoeldrich, University of Music and Performing Arts Graz

Nicki Holighaus, Austrian Academy of Sciences

Bo Holm-Rasmussen, Technical University of Denmark

Andrew Horner, The Hong Kong University of Science and Technology

Alexander Refsum Jensenius, University of Oslo

Pierre Jouvelot, MINES Paris, PSL University

David Kim-Boyle, The University of Sydney

Filipe Cunha Monteiro Lopes, Politécnico do Porto

Alexey Lukin, iZotope Inc.

Eric Lyon, Virginia Tech

Piotr Majdak, Austrian Academy of Sciences

Sylvain Marchand, University of La Rochelle

Thomas Mejstrik, University of Vienna

Romain Michon, INRIA Lyon

Fabio Morreale, The University of Auckland

Shahan Nercessian, iZotope Inc.

Søren Nielsen, SoundFocus

Oriol Nieto, Pandora Media Inc.

Stavros Ntalampiras, University of Milan

Rui Pedro Paiva, University of Coimbra

Julian Parker, TikTok/Bytedance UK

Sandra Pauletto, Royal Institute of Technology Stockholm

Miller Puckette, University of California San Diego

Marcelo Queiroz, University of São Paulo

Rudolf Rabenstein, University Erlangen-Nuremberg

Stephane Ragot, Orange Labs

Pavel Rajmic, Brno University of Technology

Gaël Richard, Telecom Paris

Davide Rocchesso, University of Palermo

Antonio Rodà, University of Padova

Axel Roebel, IRCAM

Robert Rowe, New York University

David Roze, IRCAM-CNRS

František Rund, FEE CTU Prague

Sigurd Saue, Nomono AS

Carla Scaletti, Symbolic Sound Corp.

Maximilian Schäfer, Friedrich-Alexander Universität Erlangen-Nürnberg

Jiri Schimmel, Brno University of Technology

Rod Selfridge, Edinburgh Napier University

Sertan Şentürk, Universitat Pompeu Fabra

Stefania Serafin, Aalborg University

Xavier Serra, Universitat Pompeu Fabra

Kai Siedenburg, Carl von Ossietzky University of Oldenburg

Martin Supper, Berlin University of the Arts

Tapio Takala, Aalto University

Tony Tew, University of York

Etienne Thoret, McGill University

Vesa Välimäki, Aalto University

Maarten Van Walstijn, Queen's University Belfast

Gualtiero Volpe, University of Genoa

Henrik von Coler, TU Berlin

Russell Wedelich, Eventide Inc.

Jeremy Wells, Creative Ltd.

Kurt Werner, Soundtoys Inc.

Silvin Willemsen, Eindhoven University of Technology

Alec Wright, Aalto University

Stefano Zambon, Elk Audio

DAFx Board

Welcome to DAFx23!

This year the DAFx conference responds to the tireless creativity our community with a lot of novelty. By substantially shortening oral presentations in favour of a prolonged discussion after each session, and by considering late breaking results whenever a team of researchers proposed an ongoing yet convincing idea, the 2023 edition disrupts the traditional setting. Additionally, the logo, which was established as early as 1998 for the first conference edition in Barcelona, has been renovated. With its five thematic oral sessions, a poster session, a session including demos and late breaking results, two mindopening keynotes and one special session animated by the industry, DAFx23 testifies the continuously increasing interest for digital audio effects as flagship products, squeezing the best from what acoustics, virtual analog, software engineering and product design can offer today. In these proceedings the interested reader in fact will discover unexpectedly rich and advanced applications of sound wave theory, musical circuit modelling, artificial intelligence for music, high-performance algorithms, software coding, real-time programming, human-computer interaction and subjective evaluation, to mention some.

The social part further revolutionises the get together at the conference hall, the breaks and the social events: we will be invited to extend our discussions and to follow off-events later in the evening, yet without setting the alarm clock just in time as yoga early morning sessions will be organised to engage us, and optimally set our minds to receive the science to come next across the day.

And, last but not least, let us not miss the chance to submit an extended version of our research to the special issue on Directions of Digital Audio Effects, to appear on the EURASIP Journal of Audio, Speech and Music processing under the editorial supervision of Stefania Serafin, Federico Fontana and Silvin Willemsen.

Twenty-five years have passed since the first edition in Barcelona, and DAFx shines younger than ever. So, loud and again, welcome to DAFx23!

Stefania, Federico, Silvin
Copenhagen, September 2023

Sponsors and Credits

Gold

Silver

Bronze

In collaboration with

DET KONGELIGE DANSKE MUSIKKONSERVATORIUM

CHBC Center for Hørelse og Balance

REGION Rigshospitalet

Danish Sound Cluster

Supported by

NORDIC SMC SOUND & MUSIC COMPUTING

NordForsk

Keynotes

Sustainable perceptual evaluation for digital audio applications: motivation, methods, and best practice

Hanna Järveläinen, Zurich University of the Arts

Listener tests are a standard, partly even standardized, procedure in the development of new audio technology. This kind of testing is often carried out to measure the degradation of perceived audio quality. Experimental research with participants is also performed in basic and applied psychoacoustic research, user experience studies with digital musical instruments and interfaces, and in designing applications for special groups. Within the DAFx community, methods from the audio testing field are presently most widely used. However, long-term development requires considering the end user both in passive perception and active interaction, often in multisensory, ecological, or creative settings. The presentation will discuss state-of-the art procedures and analysis methods that could contribute to digital audio research in this challenging environment.

Linking Sound, Morphology and Perception: Towards a Language of Sound

Mitsuko Aramaki, CNRS – Aix-Marseille University

The Sciences of Sound and Music have considerably grown notably thanks to the development of digital audio processing. Today, we have numerous synthesis tools, models and methods for sound creation to generate sounds of impressive realism. However, the perceptual control of these synthesis processes remains a current challenge. Based on the ecological approach to perception, a synthesis control paradigm enabling the creation of sounds from evocations has been proposed and led to the development of environmental sound synthesizers which can be directly controlled from perceptual attributes. In this talk, we will present the methodology through a series of perceptual studies to better understand the relationship between sound morphology and human perception for synthesis purposes with an interdisciplinary perspective.

Tutorials

Virtualization of Acoustic Transducers based on Direct and Inverse Circuital Modeling

Alberto Bernardini, Politecnico di Milano

Audio systems can be often accurately described using equivalent circuit models that are capable to represent their behavior in multiple physical domains (e.g., electrical, magnetic, mechanical, acoustic) in a unified fashion. Moreover, such models allow us to efficiently emulate audio systems in the digital domain by employing circuit simulation methods. In this tutorial, we will highlight a further advantage of representing audio systems using circuits. In fact, we will show how, given an audio system represented as a circuit with input and output signals, it is possible to design the corresponding inverse circuital system. As a first example of application of this inverse system design approach, we will describe a method for loudspeaker virtualization through digital audio signal preprocessing. This method can be used to alter the behavior of a physical loudspeaker in such a way to match that of a target loudspeaker. Special cases of this approach are loudspeaker linearization and equalization. The proposed virtualization algorithm is extensively tested both through simulations and applications to real loudspeakers. Moreover, we will show how a similar reciprocal approach can be used for the virtualization of acoustic sensors, like microphones or guitar pickups. Finally, further possible examples of application and related future research works are discussed.

Performance analysis of DSP algorithms

Stefano D'Angelo, Orastron Srl

Performance analysis of DSP algorithms in scientific literature is usually limited to counting the number and type of operations involved and sometimes determining their algorithmic complexity. While these metrics are important, they can only give a rough idea on the computational cost of actual implementations. This tutorial touches on theoretical and practical aspects of trying to achieve high-performance when implementing DSP algorithms on modern platforms, such as computer architectures, instruction sets, operating systems, and numerical analysis.

Recent developments in Topological Signal Processing

Georg Essl, University of Wisconsin-Milwaukee

This workshop will present recent developments in Topological Signal Processing that spring from combinatorial Hodge theory. Graph and simplicial versions of frequency analysis and filter constructions have emerged in this framework. We will cover combinatorial Hodge theory and its relationship to Homology and classical vector calculus. We then develop graph signal processing and basic notions such as the graph Fourier transform. Finally, we will extend these ideas to arrive at simplicial signal processing, which is a topological version of higher dimensional signal processing.

Workshops

Applying Machine Learning to Virtual Analog Modeling

Boris Kuznetsov, TikTok UK

How do we apply advances in the machine learning and deep learning fields to the discipline of classical signal processing and virtual analog modeling? We will talk about how you can use machine learning to both help you design and iterate on virtual analog models faster and how to potentially replace entire subsets of your work with deep learning.

Daisy Dub Hackathon for Digital Audio Effects Enthusiasts and DSP developers (Max/gen, Rust, C++, PureData, Arduino)

Rasmus Kjærbo & Leo Fogadić, Componental

Join us for an exhilarating hackathon workshop centered around the groundbreaking Daisy Dub pre-production prototype: the ultimate swiss army knife tailored for music producers, DSP developers, and DJs. Dive into the expansive world of quadrophonic real-time audio engineering and live performance and experience the processing power of Electro-Smith's Daisy Seed DFM module.

Co-designing Tactile Experiences for Musical Creativity

Doga Buse Cavdir, Multisensory Experience Lab, Aalborg University

This design workshop invites all participants to co-create new mappings for audio-tactile languages. The workshops will introduce and demo new haptic interfaces while developing discussions on the relationships between vibrotactile stimuli and music. During this two-part workshop, we hope to explore the relationships between musical and tactile communications and identify new frameworks which allow Deaf/Hard of Hearing (DHH) individuals to learn and create music using vibrotactile experiences.

Special sessions

Advances, Barriers, and Future Direction for Hearing Aid Effects

Niels H. Pontoppidan, Danish Sound

During the last 20 years advanced applications for hearing instruments only possible with machine learning (ML) emerged. However, for many and for long the computational complexity did not allow for actual implementation. Nevertheless, in 2020 core signal processing based on ML principles came in use for enhancing speech in the presence of noise. It is interesting to look back at the interplay of applications, algorithms, connectivity, and hardware to speculate about the next core signal processing areas in hearing instrument that ML will enhance.

Leveraging Deep Learning for Enhanced Signal Processing in Telecommunication Devices: A Step towards Futuristic Audio

Clément Laroche, Jabra

With the rapid evolution of artificial intelligence and deep learning algorithms, it's essential to discern their transformative impacts on telecommunication devices' audio performance, specifically headsets, speakerphones, and videobars. This presentation will start by delineating the challenges faced by conventional signal processing techniques in the current digital age, such as the inability to effectively filter ambient noises in varying environments or adapt to different speech characteristics. We then explore how can deep learning-based approaches aid in overcoming these challenges by learning complex, non-linear relationships from vast audio data. However, the computational demand and memory footprint of such advanced models can often pose a challenge for their deployment on resource-constrained embedded devices. Limitations imposed by the computational and memory requirements of deep learning models, highlighting the importance of model optimization for their practical use in real-time telecommunication devices. Spotlight will be put on dynamic neural networks, a compelling concept that allows 'early exiting' from computations. This approach facilitates rapid decisions when the network encounters less complex tasks, thus conserving computational resources — a valuable attribute for real-time applications on embedded devices. In addition to the technical aspect, we believe in the invaluable role of human listeners in validating our models. Hence, we will share results from a study conducted on Amazon's Mechanical Turk platform, where a diverse crowd sourced the rating of audio quality. The insights gathered from these human ratings provided a more nuanced understanding of the perceived audio quality, underlining the importance of a human-centric approach in our technical advancements.

Contributions

Contributions

Oral Session 1:
Physical Modelling

REAL-TIME GONG SYNTHESIS

Stefan Bilbao

Acoustics and Audio Group
University of Edinburgh
Edinburgh, United Kingdom
sbilbao@ed.ac.uk

Craig Webb

Physical Audio Ltd.
London, United Kingdom
craig@physicalaudio.co.uk

Zehao Wang

Department of Music
University of California San Diego
La Jolla, California, USA
zehaowang@ucsd.edu

Michele Ducceschi *

Department of Industrial Engineering
University of Bologna
Bologna, Italy
michele.ducceschi@unibo.it

ABSTRACT

Physical modeling sound synthesis is notoriously computationally intensive. But recent advances in algorithm efficiency, accompanied by increases in available computing power have brought real-time performance within range for a variety of complex physical models. In this paper, the case of nonlinear plate vibration, used as a simple model for the synthesis of sounds from gongs is considered. Such a model, derived from that of Föppl and von Kármán, includes a strong geometric nonlinearity, leading to a variety of perceptually-salient effects, including pitch glides and crashes. Also discussed here are input excitation and scanned multichannel output. A numerical scheme is presented that mirrors the energetic and dissipative properties of a continuous model, allowing for control over numerical stability. Furthermore, the nonlinearity in the scheme can be solved explicitly, allowing for an efficient solution in real time. The solution relies on a quadratised expression for numerical energy, and is in line with recent work on invariant energy quadratisation and scalar auxiliary variable approaches to simulation. Implementation details, including appropriate perceptually-relevant choices for parameter settings are discussed. Numerical examples are presented, alongside timing results illustrating real-time performance on a typical CPU.

1. INTRODUCTION

Physical modeling synthesis has now reached a certain level of maturity. It has become possible to perform audio rate simulations of relatively complex systems in real time. One reason for this follows from the steady increase in available computing power, accompanied by newer tools allowing code acceleration using low-level parallelisation on the CPU [1]. More important, though, have been advances in algorithm efficiency, particularly for systems exhibiting a strong nonlinearity, the subject of this paper.

Perhaps the strongest nonlinear mechanism in any acoustic musical instrument is found in gong-like percussion instruments leading to characteristic pitch glides, crashes and swells. A linear model is grossly insufficient to capture such effects, and the nonlinearity is distributed throughout the entire vibrating structure, normally modelled as a thin plate or shell. Physical models have been available for some time [2, 3], and used for sound synthesis purposes [4], but have been limited to offline use. Computational cost is large, due to the essentially 2D nature of such models, and increased further due to the complexity of the geometric nonlinearity, alongside various practical algorithm design constraints.

The most important of these constraints is the requirement for numerically stable behaviour. Though simple efficient explicit time domain simulation methods, such as Störmer-Verlet integration are available [5], stability is not ensured, and indeed such methods are highly prone to explosive instability, particularly at high amplitudes, exactly at the onset of perceptually salient nonlinear effects. One approach to ensuring numerical stability is through the use of energy-conserving numerical designs, where the solution size is bounded by a numerical invariant (the energy or pseudo-energy). In the present case of nonlinear plate vibration, such methods have been proposed, and allow for designs of so-called linearly implicit character—costly iterative methods such as Newton Raphson are avoided, but potentially large linear systems must be both constructed and solved in the run-time loop [6]—real-time performance is ruled out for such methods. A more recent approach follows from invariant energy quadratisation [7, 8] and scalar auxiliary variable [9, 10] methods applied in geometric numerical integration. In general, these also lead to algorithms with the same linearly-implicit character. However, recent results have allowed for fully explicit numerical solutions through the exploitation of structure in the linear system to be solved, increasing the speed of calculation by an order of magnitude at least [11, 12], while maintaining stable numerical behaviour. This paper is concerned with the range of algorithmic and programming techniques necessary in order to generate gong-like sounds in real time.

A model of nonlinear plate vibration at high amplitudes, based on the dynamic analogue of the model of Föppl and von Kármán, and including effects of loss as well as input excitation and scanned multichannel output, is presented in Section 2. An energy balance is also presented. The basic steps leading to a discrete-time simulation algorithm are presented in Section 3, beginning from the definition of a basic spatial grid and difference operators, and proceeding to a semi-discrete form. This form may then be written directly

* Michele Ducceschi has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, Grant agreement No. 950084 - NEMUS.

as an extension of a Hamiltonian system. A fully discrete time numerically stable algorithm accompanied by an energy balance may be constructed, and has the advantage that the nonlinearity is dealt with fully explicitly. Approaches to the numerical solution of the remaining required linear system, involving the biharmonic operator are also outlined. Implementation details, including the simplification of user-supplied instrument design and control parameters, are described in Section 4. Numerical results, including timings illustrating real time performance on a standard CPU, as well as spectrograms of representative outputs are provided in Section 5. Concluding remarks appear in Section 6. Sound examples are available at the companion page [1].

2. MODEL

Realistic sound synthesis from a gong-like instrument requires a nonlinear model of plate vibration—necessary in order to capture strong amplitude-dependent effects such as pitch glides, crashes and swells. Linear models (such as, e.g. the thin model due to Kirchhoff [13] or even thick models of Mindlin-Reissner form [14]) are insufficient for this purpose. Simplified nonlinear models such as that of Berger [15] are able to replicate pitch glides, but not the energy cascade to high frequencies characteristic of crashes.

The simplest suitable model is the dynamic analogue of the system of Föppl and von Kármán [16, 17, 18], and describes the high amplitude vibration of thin plates. When accompanied by additional terms emulating loss, and a forcing term, it may be written as the following coupled system of partial differential equations:

$$\begin{aligned} \rho\xi\partial_t^2 w &= -Q\Delta\Delta w - 2\rho\xi\sigma_0\partial_t w + 2\rho\xi\sigma_1\partial_t\Delta w \\ &\quad +\mathcal{L}(w,\Phi) + \delta(\mathbf{r}-\mathbf{r}_\mathrm{i})f \qquad (1a) \\ \frac{2}{E\xi}\Delta\Delta\Phi &= -\mathcal{L}(w,w)\,. \qquad (1b) \end{aligned}$$

Here, $w(\mathbf{r},t)$ and $\Phi(\mathbf{r},t)$ are the transverse plate deflection and Airy stress function, respectively. Both are functions of time $t \geq 0$, and spatial coordinates $\mathbf{r} = (x,y) \in \mathcal{D} \subset \mathbb{R}^2$. In this paper, the rectangular domain $\mathcal{D} = [-L_x/2, L_x/2] \times [-L_y/2, L_y/2]$ will be considered, for plate side lengths L_x, L_y in m. See Figure 1. The other material and geometric parameters that define the plate are the density ρ, in kg m^{-3}, the plate thickness ξ, in m, and Young's modulus E in Pa—the flexural rigidity Q is defined as $Q = E\xi^3/12(1-\nu^2)$, where ν is Poisson's ratio for the plate material. The two parameters σ_0, in s^{-1} and σ_1, in m^{2}s^{-1} give two-parameter control over frequency-dependent loss [19]—see also Section 4.1. ∂_t represents partial differentiation with respect to time t, and Δ the two-dimensional Laplacian, where $\Delta = \partial_x^2 + \partial_y^2$, for spatial partial derivatives ∂_x and ∂_y with respect to coordinates x and y respectively. $\Delta\Delta$ is the biharmonic operator.

The nonlinear operator $\mathcal{L}$, defined, in terms of its operation on two functions $\alpha(x,y)$ and $\beta(x,y)$, as:

$$\mathcal{L}(\alpha,\beta) = \partial_x^2\alpha\partial_y^2\beta + \partial_y^2\alpha\partial_x^2\beta - 2\partial_x\partial_y\alpha\partial_x\partial_y\beta\,. \qquad (2)$$

For simplicity, boundary conditions are chosen to be of simply supported type over the boundary $\partial\mathcal{D}$ of $\mathcal{D}$, so that

$$w = \Delta w = 0 \qquad \Phi = \Delta\Phi = 0 \qquad \text{for} \qquad \mathbf{r} \in \partial\mathcal{D}\,. \qquad (3)$$

Initial conditions are assumed to be zero, so that

$$w(\mathbf{r},0) = \partial_t w|_{\mathbf{r},t=0} = 0\,. \qquad (4)$$

[1] https://physicalaudio.co.uk/modelling-gongs/

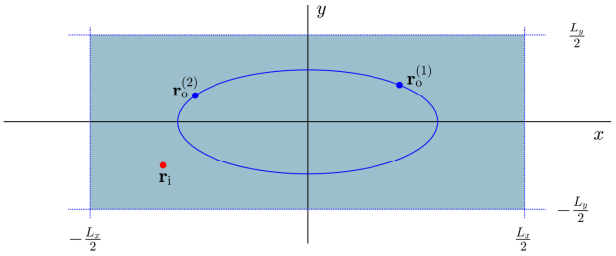


Figure 1: *Plate geometry, with side lengths L_x and L_y. The driving location $\mathbf{r}_\mathrm{i}$ is indicated, as well as the two output locations $\mathbf{r}_\mathrm{o}^{(1)}(t)$ and $\mathbf{r}_\mathrm{o}^{(2)}(t)$, drawn from an elliptical trajectory.*

Initial conditions for Φ do not need to be set independently.

Versions of system (1) have been used in various studies of percussion instruments, and in particular, the cases of circular plates [2] and the generalisation to the case of curved plates or shells [3]. Shells of variable thickness have also been examined in the context of cymbal acoustics [20]. Here, a simplified flat rectangular geometry is chosen, as it reproduces many of the features that are characteristic of gong-like instruments, and also leads to simulation algorithm designs that are very well suited for acceleration.

2.1. Input and Output

Also included in system (1) is a point-like excitation term. $f(t)$, in N, is a forcing function applied at location $\mathbf{r}_\mathrm{i} = (x_\mathrm{i}, y_\mathrm{i})$; δ is a 2-dimensional Dirac delta function. A full model of the interaction between a striking object (such as a mallet) and the plate could be included here, as in earlier models of percussion instruments [21]. Because the interaction time is generally extremely short (on the order of 1-5 ms), a much simpler approach is to model this interaction using an externally supplied excitation function of the form of a short pulse. A suitable candidate is a time-limited raised sinusoidal distribution [19] of the form

$$f(t) = \begin{cases} f_\mathrm{max}\sin^2\left(\frac{\pi(t-t_0)}{T}\right), & t_0 \leq t \leq t_0+T \\ 0, & \text{otherwise} \end{cases}\,. \qquad (5)$$

See Figure 2. Other excitation functions can be applied—including steady sinusoidal functions, and possibly even audio, in which case the physical model behaves as an effect instead of a synthesizer.

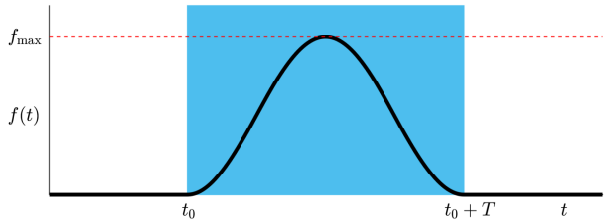


Figure 2: *Excitation function $f(t)$, as defined in* (5).

For output, the picture is slightly different. Output audio signals $w_\mathrm{o}^{(p)}(t)$ can be drawn from the plate at P distinct locations $\mathbf{r}_\mathrm{o}^{(p)}$, $p = 1,\ldots,P$, and defined in terms of plate displacement, as

$$w_\mathrm{o}^{(p)}(t) = w(\mathbf{r}_\mathrm{o}^{(p)}, t)\,. \qquad (6)$$

It can be useful, as an additional effect, to allow the output locations to be time varying. Here, outputs $\mathbf{r}_{\mathrm{o}}^{(p)} = (x_{\mathrm{o}}^{(p)}, y_{\mathrm{o}}^{(p)})$ are drawn from an elliptical distribution as

$$x_{\mathrm{o}}^{(p)}(t) = \frac{L_x R}{2} \cos(2\pi f_{\mathrm{o}}^{(p)} t + \phi_{\mathrm{o}}^{(p)}) \quad (7a)$$

$$y_{\mathrm{o}}^{(p)}(t) = \frac{L_y R}{2} \sin(2\pi f_{\mathrm{o}}^{(p)} t + \phi_{\mathrm{o}}^{(p)}) \,. \quad (7b)$$

Here, $0 \leq R < 1$ is a dimensionless parameter controlling the size of the ellipse, $f_{\mathrm{o}}^{(p)}$ is a scan frequency (sub audio rate), and $\phi_{\mathrm{o}}^{(p)}$ is an initial phase. See Figure 1. If different values are used for each output, with fixed R, then a natural phasing effect is obtained, while maintaining uniform normalisation for all channels.

2.2. Energy Balance

System (1) satisfies an energy balance of the form

$$\dot{\mathcal{H}} = -\mathcal{Q} + \mathcal{P} \,, \quad (8)$$

where here, a dot indicates ordinary time differentiation. $\mathcal{H}(t)$ is the total stored energy in the plate, $\mathcal{Q}(t)$ is power loss, and $\mathcal{P}(t)$ is input power. These can be defined explicitly [19] as

$$\mathcal{H} = \iint_{\mathcal{D}} \frac{\rho\xi}{2}(\partial_t w)^2 + \frac{Q}{2}(\Delta w)^2 + \frac{1}{2E\xi}(\Delta\Phi)^2 d\mathbf{r} \quad (9a)$$

$$\mathcal{Q} = \iint_{\mathcal{D}} 2\rho\xi\sigma_0(\partial_t w)^2 + 2\rho\xi\sigma_1|\nabla\partial_t w|^2 d\mathbf{r} \quad (9b)$$

$$\mathcal{P} = f\partial_t w|_{\mathbf{r}_{\mathrm{i}},t} \,. \quad (9c)$$

Here, ∇ indicates a gradient with respect to $\mathbf{r}$. All are scalar functions. In particular, both $\mathcal{H}$ and $\mathcal{Q}$ are non-negative, meaning that the system is dissipative under zero input conditions (and lossless when $\sigma_0 = \sigma_1 = 0$, meaning that the energy $\mathcal{H}(t)$ is constant).

3. FDTD METHODS

3.1. Spatial Grid and Difference Operators

As a first step, suppose that the plate surface is discretised with a 2D grid of spacing $h = L_x/N_x$, for some integer N_x. Then, set $N_y = \lfloor L_y/h \rfloor$, where $\lfloor\cdot\rfloor$ indicates a flooring operation. Here, for simplicity, the plate side length in the y direction is set to $N_y h \approx L_y$. The semi-discrete grid functions $w_{l,m}(t)$ and $\Phi_{l,m}(t)$, indexed by integers l and m with $1 \leq l \leq N_x - 1$ and $1 \leq m \leq N_y - 1$, represent approximations to $w(\mathbf{r}, t)$ and $\Phi(\mathbf{r}, t)$ at $\mathbf{r} = -1/2(L_x, L_y) + h(l, m)$. Due to the choice of simply supported boundary conditions (3), the grid functions are assumed to take on values of zero at $l = 0$, $l = N_x$, $m = 0$ and $m = N_y$—and thus, such points may be excluded from the algorithm entirely.

For a given grid function $u_{l,m}(t)$, forward and backward spatial differences in the x and y directions, approximating ∂_x and ∂_y, are defined (suppressing time dependence) as

$$D_x^{\pm} u_{l,m} = \pm\frac{u_{l\pm1,m} - u_{l,m}}{h} \quad D_y^{\pm} u_{l,m} = \pm\frac{u_{l,m\pm1} - u_{l,m}}{h} \,. \quad (10)$$

Second derivative approximations follow directly as:

$$D_{xx} = D_x^+ D_x^- \qquad D_{yy} = D_y^+ D_y^- \,. \quad (11)$$

The Laplacian and biharmonic may then be approximated as

$$D_\Delta = D_{xx} + D_{yy} \qquad D_{\Delta\Delta} = D_\Delta D_\Delta \,. \quad (12)$$

Finally, four approximations D_{xy}^{ab} to $\partial_x\partial_y$ may be defined as

$$D_{xy}^{ab} = D_x^a D_y^b \qquad \text{for} \qquad a, b \in \{+, -\} \,. \quad (13)$$

3.2. Semi-discrete Form

Before moving directly to a semi-discrete form, it is useful to recast the $(N_x - 1) \times (N_y - 1)$ grid functions $w_{l,m}(t)$ and $\Phi_{l,m}(t)$ as $N \times 1$ column vectors $\mathbf{w}(t)$ and $\mathbf{\Phi}(t)$, through concatenation of consecutive columns of $w_{l,m}(t)$ and $\Phi_{l,m}(t)$. Here, $N = (N_x - 1)(N_y - 1)$ is the total number of grid points in either grid function. The linear operators D_{xx}, D_{yy}, D_{xy}^{ab}, D_Δ and $D_{\Delta\Delta}$ can thus be represented as sparse $N \times N$ matrices $\mathbf{D}_{xx}$, $\mathbf{D}_{yy}$, $\mathbf{D}_{xy}^{ab}$, $\mathbf{D}_\Delta$ and $\mathbf{D}_{\Delta\Delta}$, respectively. Simply supported boundary conditions are assumed directly encoded into these matrices [19]. Also necessary is an approximation to the Dirac delta function which selects the excitation location in (1). This may be represented as an $N \times 1$ column vector $\frac{1}{h^2}\mathbf{j}$. Many approximations to the delta function over a grid are available [22]; for simplicity, excitation is assumed to occur directly at a grid point, so that $\mathbf{j}$ is all zero except for a single value of 1 at the excitation location.

A semi-discrete form of (1) may be written directly as

$$\rho\xi\ddot{\mathbf{w}} = -Q\mathbf{D}_{\Delta\Delta}\mathbf{w} - 2\rho\xi\sigma_0\dot{\mathbf{w}} + 2\rho\xi\sigma_1\mathbf{D}_\Delta\dot{\mathbf{w}} + \ell(\mathbf{w}, \mathbf{\Phi}) + \frac{1}{h^2}\mathbf{j}f \quad (14a)$$

$$\frac{2}{E\xi}\mathbf{D}_{\Delta\Delta}\mathbf{\Phi} = -\ell(\mathbf{w}, \mathbf{w}) \,. \quad (14b)$$

Here, a discrete counterpart to the nonlinear operator $\mathcal{L}$, as defined in (2), may be written in terms of its action on two $N \times 1$ vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, as

$$\ell(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbf{D}_{xx}\boldsymbol{\alpha} \odot \mathbf{D}_{yy}\boldsymbol{\beta} + \mathbf{D}_{yy}\boldsymbol{\alpha} \odot \mathbf{D}_{xx}\boldsymbol{\beta} - \tfrac{1}{2} \sum_{a,b\in\{+,-\}} \mathbf{D}_{xy}^{ab}\boldsymbol{\alpha} \odot \mathbf{D}_{xy}^{ab}\boldsymbol{\beta} \quad (15)$$

where $\odot$ denotes element-wise multiplication of two vectors. The approximation ℓ to $\mathcal{L}$ is bilinear and possesses various important symmetry properties, including that of triple self-adjointness [19] inherited from the continuous operator $\mathcal{L}$—the reader is referred to the literature for further discussion [23].

3.3. Energy Balance and Potential Energy Quadratisation

For analysis purposes, it is useful to rewrite the system of ordinary differential equations (14) in terms of displacement $\mathbf{w}$ and momentum $\mathbf{p}$, as:

$$\dot{\mathbf{w}} = \nabla_{\mathbf{p}} H \qquad \dot{\mathbf{p}} = -\nabla_{\mathbf{w}} H - \mathbf{R}\mathbf{p} + \mathbf{j}f \,. \quad (16)$$

Here, $H(t)$ is the total system energy, defined as

$$H = \tfrac{1}{2M}\mathbf{p}^T\mathbf{p} + V_0 + V' \quad (17)$$

and $\nabla_{\mathbf{p}}$ and $\nabla_{\mathbf{w}}$ represent gradients with respect to $\mathbf{p}$ and $\mathbf{w}$, respectively. The first term in H represents kinetic energy, where $M = \rho\xi h^2$ is the mass per grid point, in kg. V_0 and V' represent contributions to the potential energy due to linear and nonlinear effects, respectively, and are defined by

$$V_0 = \tfrac{1}{2}\mathbf{w}^T\mathbf{K}_0\mathbf{w} \qquad V' = \tfrac{1}{2}\mathbf{\Phi}^T\mathbf{K}'\mathbf{\Phi} \,, \quad (18)$$

where

$$\mathbf{K}_0 = Qh^2\mathbf{D}_{\Delta\Delta} > \mathbf{0} \qquad \mathbf{K}' = \tfrac{h^2}{E\xi}\mathbf{D}_{\Delta\Delta} > \mathbf{0}\,. \tag{19}$$

The positive definiteness conditions above hold under simply supported boundary conditions. The plate nonlinearity intervenes through the relationship (14b) between $\mathbf{\Phi}$ and $\mathbf{w}$.

The equations (16) are an extension of Hamilton's equations, including loss and a forcing term. Loss is encoded here through the matrix $\mathbf{R}$, defined as

$$\mathbf{R} = \underbrace{2\sigma_0\mathbf{I}_N}_{\mathbf{R}_0} + \underbrace{-2\sigma_1\mathbf{D}_\Delta}_{\mathbf{R}_1} \geq \mathbf{0}\,, \tag{20}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix. Under unforced conditions, with $f = 0$, the system is dissipative, so that

$$\dot{H} = -\tfrac{1}{M}\mathbf{p}^T\mathbf{R}\mathbf{p} \leq 0\,. \tag{21}$$

$H(t)$ is the semi-discrete counterpart of the total plate energy, as defined in (9a). When $\sigma_0 = \sigma_1 = 0$, the system is lossless.

V', as defined in (18), is non-negative. Scalar auxiliary variable methods follow from the definition of a new variable ψ, as

$$V' = \tfrac{1}{2}\psi^2\,. \tag{22}$$

Notice here that only the contribution V' to the energy due to nonlinear effects has been quadratised here—other possibilities are avalable [12]. System (16) can then be rewritten, using this definition, as well as the quadratic dependence of H on $\mathbf{p}$, as

$$\dot{\mathbf{w}} = \tfrac{1}{M}\mathbf{p} \qquad \dot{\mathbf{p}} = -\mathbf{K}_0\mathbf{w} - \psi\mathbf{g} - \mathbf{R}\mathbf{p} + \mathbf{j}f\,, \tag{23}$$

where

$$\mathbf{g} \triangleq \nabla_{\mathbf{w}}\psi\,. \tag{24}$$

Furthermore, using the chain rule,

$$\dot{\psi} = (\nabla_{\mathbf{w}}\psi)^T\dot{\mathbf{w}} = \mathbf{g}^T\dot{\mathbf{w}}\,. \tag{25}$$

Given that H is quadratic in $\mathbf{p}$, a second order form of (23) follows immediately as

$$M\ddot{\mathbf{w}} = -\mathbf{K}_0\mathbf{w} - \psi\mathbf{g} - M\mathbf{R}\dot{\mathbf{w}} + \mathbf{j}f\,. \tag{26}$$

This equation, alongside (25), describing the time evolution of the scalar auxiliary variable ψ, and the nonlinear relationship (14b), forms a complete system describing the vibration of the plate.

3.4. Fully Discrete Form

A discrete update preserving an energy balance may be arrived at directly, generalising results in [12]. First, beginning from system (23) and (25), define the time series $\mathbf{w}^n$ and $\mathbf{g}^n$, representing approximations to $\mathbf{w}(t)$ and $\mathbf{g}(t)$ at times $t = nk$, for a given time step k in s, and for integer n, and $\mathbf{p}^{n+1/2}$, $\psi^{n+1/2}$, interleaved approximations to $\mathbf{p}(t)$ and $\psi(t)$ at times $t = (n + 1/2)k$. Consider the following time-interleaved scheme, resulting from basic centered differences and averaging of (23) and (25):

$$\mathbf{w}^{n+1} = \mathbf{w}^n + \tfrac{k}{M}\mathbf{p}^{n+\frac{1}{2}} \tag{27a}$$

$$\mathbf{p}^{n+\frac{1}{2}} = \mathbf{p}^{n-\frac{1}{2}} - k\mathbf{K}_0\mathbf{w}^n - \frac{k}{2}\left(\psi^{n+\frac{1}{2}} + \psi^{n-\frac{1}{2}}\right)\mathbf{g}^n - \frac{k}{2}\mathbf{R}_0\left(\mathbf{p}^{n+\frac{1}{2}} + \mathbf{p}^{n-\frac{1}{2}}\right) - k\mathbf{R}_1\mathbf{p}^{n-\frac{1}{2}} + k\mathbf{j}f^n \tag{27b}$$

$$\psi^{n+\frac{1}{2}} = \psi^{n-\frac{1}{2}} + \tfrac{1}{2}\left(\mathbf{g}^n\right)^T\left(\mathbf{w}^{n+1} - \mathbf{w}^{n-1}\right)\,. \tag{27c}$$

Notice that in (27b), the linear and nonlinear parts of the plate dynamics have been approximated using different integration rules; and, in order to arrive at an explicit update, as will be seen shortly, the loss terms have been approximated separately, according to the decomposition in (20). f^n is an approximation to $f(t)$ at $t = nk$; the calculation of $\mathbf{g}^n$ will be returned to in Section 3.6.

The updates (27a) and (27b) can be consolidated into a single two-step update in $\mathbf{w}^n$

$$\mathbf{A}^n\mathbf{w}^{n+1} = \mathbf{U}\mathbf{w}^n - \mathbf{C}^n\mathbf{w}^{n-1} + \tfrac{k^2}{M}\left(f^n\mathbf{j} - \psi^{n-\frac{1}{2}}\mathbf{g}^n\right) \tag{28}$$

and depends upon the three matrices $\mathbf{A}^n$, $\mathbf{U}$ and $\mathbf{C}^n$ defined by

$$\mathbf{A}^n = \mathbf{I}_N + \tfrac{k}{2}\mathbf{R}_0 + \tfrac{k^2}{4M}\mathbf{g}^n\left(\mathbf{g}^n\right)^T \tag{29a}$$

$$\mathbf{U} = 2\mathbf{I}_N - \tfrac{k^2}{M}\mathbf{K}_0 - k\mathbf{R}_1 \tag{29b}$$

$$\mathbf{C}^n = \mathbf{I}_N - \tfrac{k}{2}\mathbf{R}_0 - k\mathbf{R}_1 - \tfrac{k^2}{4M}\mathbf{g}^n\left(\mathbf{g}^n\right)^T\,. \tag{29c}$$

This update is apparently implicit, requiring the inversion of $\mathbf{A}^n$ at each time step. However, $\mathbf{A}^n$ is of the form of a multiple of the identity plus a rank-one perturbation, or $\mathbf{A}^n = d\mathbf{I}_N + \boldsymbol{a}^n(\boldsymbol{a}^n)^T$, where $d = 1 + k\sigma_0$ and $\boldsymbol{a}^n = \frac{k}{2\sqrt{M}}\mathbf{g}^n$. A closed-form inverse is available through the Sherman-Morrison formula [24] as

$$(\mathbf{A}^n)^{-1} = d^{-1}\left(\mathbf{I}_N - \frac{\boldsymbol{a}^n(\boldsymbol{a}^n)^T}{d + (\boldsymbol{a}^n)^T\boldsymbol{a}^n}\right)\,. \tag{30}$$

Thus the linear system solution required in (28) can be performed in $O(N)$ operations, and the update can be viewed as effectively explicit.

3.5. Discrete Energy Balance and Stability Condition

In the lossless and source-free case, the scheme (27) is lossless to machine precision, as demonstrated recently [12]. When loss and sources are present, an energy balance of the following form holds:

$$\frac{1}{k}\left(\mathfrak{h}^{n+\frac{1}{2}} - \mathfrak{h}^{n-\frac{1}{2}}\right) = -\mathfrak{q}^n + \mathfrak{p}^n\,, \tag{31}$$

where

$$\mathfrak{h}^{n+\frac{1}{2}} = \tfrac{1}{2M}|\mathbf{p}^{n+\frac{1}{2}}|^2 + \tfrac{1}{2}\mathbf{w}^{n+1}\mathbf{K}_0\mathbf{w}^n + \tfrac{1}{2}\left(\psi^{n+\frac{1}{2}}\right)^2 - \tfrac{k}{4M}(\mathbf{p}^{n+\frac{1}{2}})^T\mathbf{R}_1\mathbf{p}^{n+\frac{1}{2}} \tag{32a}$$

$$\mathfrak{q}^n = \tfrac{1}{4M}(\mathbf{p}^{n+\frac{1}{2}} + \mathbf{p}^{n-\frac{1}{2}})^T(\mathbf{R}_0 + k\mathbf{R}_1)(\mathbf{p}^{n+\frac{1}{2}} + \mathbf{p}^{n-\frac{1}{2}}) \tag{32b}$$

$$\mathfrak{p}^n = \tfrac{1}{2M}(\mathbf{p}^{n+\frac{1}{2}} + \mathbf{p}^{n-\frac{1}{2}})^T\mathbf{j}f^n\,. \tag{32c}$$

This energy balance mirrors that of the continuous system, from (9); the major difference is that the expression $\mathfrak{h}^{n+1/2}$ for the stored energy is only non-negative under the condition

$$h \geq 2\sqrt{k}\sqrt{\sigma_1 + \sqrt{\sigma_1^2 + Q/\rho\xi}}\,, \tag{33}$$

which is the numerical stability condition for scheme (28). This is the same condition that follows from an analysis of numerical stability for the plate under linear conditions (i.e., the Kirchhoff plate) [19]. In practice, h will be set as close to possible above this lower bound. Another difference, with respect to the energy for the continuous system, from (9a), is the appearance of additional stored energy above due to the loss term—this is the result of using a non-centered (backward) difference for the frequency-dependent loss term, allowing an explicit update for the plate.

3.6. Linear System Solution: The Biharmonic Operator

The scheme (28), as written, appears to be fully explicit, once Sherman-Morrison inversion is employed in order to solve the linear system involving $\mathbf{A}^n$. A hidden aspect here, though, is the determination of $\mathbf{g}^n$, approximating $\mathbf{g}(t)$ as defined in (24) at time $t = nk$. Regardless of the form of this approximation, the scheme (28) will be dissipative under zero-input conditions—note that $\mathbf{g}^n$ does not appear explicitly in the energetic expressions in (32). It must, however, be chosen to be consistent with the definition of $\mathbf{g}(t)$ in order to lead to a convergent algorithm. Note, from the definition of $\mathbf{g}$, that one may furthermore write, using $V' = \frac{1}{2}\psi^2$,

$$\mathbf{g} = \nabla_{\mathbf{w}}\psi = \tfrac{1}{\sqrt{2V'}}\nabla_{\mathbf{w}}V' \,. \tag{34}$$

From the definition of V' in terms of $\mathbf{\Phi}$, from (18), and also the definition of Φ in terms of $\mathbf{w}$, from (14b), one may ultimately arrive at the following form [12] for $\mathbf{g}^n$:

$$\mathbf{g}^n = -\tfrac{h^2}{\sqrt{2(V')^n}}\ell(\mathbf{w}^n, \mathbf{\Phi}^n) \,, \tag{35}$$

where, in discrete time,

$$(V')^n = \tfrac{1}{2}(\mathbf{\Phi}^n)^T\mathbf{K}'\mathbf{\Phi}^n \quad \text{and} \quad \mathbf{D}_{\Delta\Delta}\mathbf{\Phi}^n = -\tfrac{EH}{2}\ell(\mathbf{w}^n, \mathbf{w}^n) \,, \tag{36}$$

and where the bilinear operator ℓ is as defined in (15).

Most of the operations above required in order to form $\mathbf{g}^n$ are simple. The exception is the linear system involving the biharmonic operator $\mathbf{D}_{\Delta\Delta}$ required in (36) in order to determine $\mathbf{\Phi}^n$ from $\mathbf{w}^n$. This is the remaining computational bottleneck, and is inherent to all numerical solutions to the Föppl-von Kármán equations. Using the fact that, under simply supported conditions, the biharmonic may be separated into a product of two Laplacians as $\mathbf{D}_{\Delta\Delta} = \mathbf{D}_\Delta\mathbf{D}_\Delta$, the linear system to solved at each time step is:

$$\mathbf{D}_\Delta\mathbf{D}_\Delta\mathbf{y} = \mathbf{c} \tag{37}$$

for a known $N \times 1$ vector $\mathbf{c}$, yielding an $N \times 1$ vector $\mathbf{y}$.

Standard linear system solvers (such as, e.g., those relying on Cholesky or LU factorisation) are far out of real time for reasonable plate sizes. See Section 5.1. Here, a very recently developed solver [25] that exploits the structure of $\mathbf{D}_\Delta$ is employed, leading to an efficient variant of the Thomas algorithm [26]. It is closely related to the method proposed by Buzbee [27]. To this end, note that the scaled $N \times N$ Laplacian operator $\tilde{\mathbf{D}}_\Delta = h^2\mathbf{D}_\Delta$ may be written explicitly as the Toeplitz-block-Toeplitz form:

$$\tilde{\mathbf{D}}_\Delta = \begin{bmatrix} \mathbf{T} & \mathbf{I} & & & \bullet \\ \mathbf{I} & \mathbf{T} & \mathbf{I} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{I} & \mathbf{T} & \mathbf{I} \\ \bullet & & & \mathbf{I} & \mathbf{T} \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} -4 & 1 & & & \bullet \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ \bullet & & & 1 & -4 \end{bmatrix} . \tag{38}$$

Here, block sizes are $(N_y - 1) \times (N_y - 1)$. $\mathbf{I}$ is an identity matrix, and $\mathbf{T}$ a Toeplitz matrix as given above. Zero entries are indicated by $\bullet$, and $\tilde{\mathbf{D}}_\Delta$ is extremely sparse. The matrix $\mathbf{T}$ has the closed form eigendecomposition $\mathbf{T} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}$ where

$$[\mathbf{S}]_{\beta\gamma} = \sqrt{\tfrac{2}{N_y}}\sin(\beta\gamma\pi/N_y) \quad [\mathbf{\Lambda}]_{\beta\beta} = 2\cos(\beta\pi/N_y) - 4 \tag{39}$$

for $1 \leq \beta, \gamma \leq N_y - 1$. Note that $\mathbf{S} = \mathbf{S}^T = \mathbf{S}^{-1}$ is an orthogonal $(N_y - 1) \times (N_y - 1)$ matrix. $\mathbf{\Lambda}$ is diagonal, and contains the eigenvalues of $\tilde{\mathbf{D}}_\Delta$.

Now, form the $N \times N$ matrix $\mathbf{Q}$ as a Kronecker product $\mathbf{Q} = \mathbf{I}_{N_x-1} \otimes \mathbf{S}$. Using the orthogonality of $\mathbf{S}$, one has

$$\tilde{\mathbf{D}}_\Delta = \mathbf{Q}\mathbf{\Xi}\mathbf{Q} \qquad \mathbf{\Xi} = \begin{bmatrix} \mathbf{\Lambda} & \mathbf{I} & & & \bullet \\ \mathbf{I} & \mathbf{\Lambda} & \mathbf{I} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{I} & \mathbf{\Lambda} & \mathbf{I} \\ \bullet & & & \mathbf{I} & \mathbf{\Lambda} \end{bmatrix} . \tag{40}$$

Given that $\mathbf{Q}$ inherits orthogonality from $\mathbf{S}$, one may then write the solution to (37) as

$$\mathbf{y} = h^4\mathbf{Q}\mathbf{\Xi}^{-1}\mathbf{\Xi}^{-1}\mathbf{Q}\mathbf{c} \,. \tag{41}$$

Note here that $\mathbf{Q}$ is sparse, with $N_x - 1$ blocks of size $(N_y - 1) \times (N_y - 1)$. Also, $\mathbf{\Xi}$ is block tridiagonal, with diagonal blocks, and thus the Thomas algorithm may be applied directly (twice, here).

Other approaches to Toeplitz-block-Toeplitz linear system solution are available, and were tested during the course of this work. These include extensions of the Levinson-Durbin algorithm due to Wax and Kailath [28]. One may also note that the matrix $\mathbf{S}$ corresponds to a discrete sine transform (DST); fft-based methods were tried, but were not efficient, due to the small size of N_y, and exhibited great variation depending on the factorisation of N_y. The method presented above performed best in all tests.

3.7. Output and Interpolation

Moving outputs are assumed drawn at locations $(\mathbf{r}_\text{o}^{(p)})^n$ sampled from the elliptical trajectories $\mathbf{r}_\text{o}^{(p)}(t)$, $p = 1, \ldots, P$ as defined in (7), and at times $t = nk$. Interpolation is a necessity in this setting, in order to avoid numerical artefacts ("zipper noise").

Suppose, at a given time instant, the coordinates of one of the trajectories takes on the value $\boldsymbol{\eta} = (\mathbf{r}_\text{o}^{(p)})^n$. One may write

$$\boldsymbol{\eta} = h(l_\text{o}, m_\text{o}) + h(\zeta_x, \zeta_y) \tag{42}$$

uniquely for integer grid indeces (l_o, m_o) and fractional addresses (ζ_x, ζ_y), where $0 \leq \zeta_x, \zeta_y < 1$. Assuming an interpolation width of $2J$ points, one may form an interpolant w_o from the two-dimensional grid function $w_{l,m}$ as:

$$w_\text{o} = \sum_{\nu_x=-J+1}^{J}\sum_{\nu_y=-J+1}^{J} b_x^{\nu_x}(\zeta_x)b_y^{\nu_y}(\zeta_y)w_{l_\text{o}+\nu_x, m_\text{o}+\nu_x} \,. \tag{43}$$

Here, the interpolant is assumed separable, so that $b_x^{\nu_x}(\zeta_x)$ and $b_y^{\nu_y}(\zeta_y)$, $-J+1 \leq \nu_x, \nu_y \leq J$ are one-dimensional interpolation coefficients. In this work, approximations are assumed to be of Lagrange type, with $J = 2$.

When the grid function $w_{l,m}$ is reconstituted as an $N \times 1$ column vector $\mathbf{w}$, then this linear operation may be expressed as an inner product

$$w_\text{o} = \mathbf{b}^T\mathbf{w} \,, \tag{44}$$

where the $N \times 1$ vector $\mathbf{b}$ incorporates the interpolation coefficients b_x and b_y The extension to the case of P time-varying output trajectories is straightforward, and may be represented as

$$\mathbf{w}_\text{o}^n = (\mathbf{B}^n)^T\mathbf{w}^n \,. \tag{45}$$

Here, $\mathbf{w}_\text{o}^n$ is a $P \times 1$ column vector of output signals, and $\mathbf{B}^n = [\mathbf{b}^{n,(1)} \ldots \mathbf{b}^{n,(P)}]$ is an $N \times P$ matrix of interpolation coefficients.

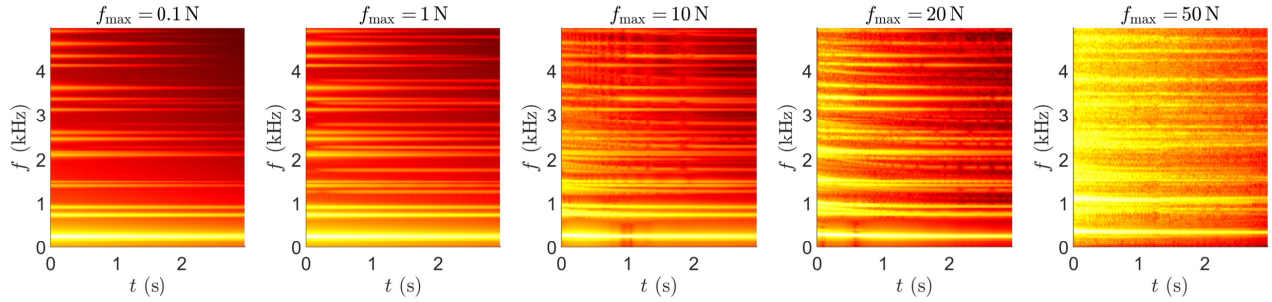


Figure 3: *Spectrograms of output from a small plate model, at different excitation amplitudes, as indicated.*

4. IMPLEMENTATION

4.1. Parameter Sets

Complete parameter sets for the gong, assuming a struck excitation, and P outputs over an elliptical trajectory, are as follows:

$$\mathcal{O}_{\text{plate}} = \{E, \rho, \xi, \nu, L_x, L_y, \sigma_0, \sigma_1\} \quad (46a)$$

$$\mathcal{O}_{\text{I}} = \{t_0, T, f_{\max}\} \quad (46b)$$

$$\mathcal{O}_{\text{O}} = \{R, f_{\text{o}}^{(1)}, \ldots, f_{\text{o}}^{(P)}, \phi_{\text{o}}^{(1)}, \ldots, \phi_{\text{o}}^{(P)}\}\,. \quad (46c)$$

The sample rate F_s must also be supplied. For the given system, this set of parameters is redundant, in terms of the space of possible sound outputs. One approach to reducing the number of defining parameters in $\mathcal{O}_{\text{plate}}$ is to non-dimensionalise the system (1) with respect to w and Φ, and to spatially scale the domain to unit area. Equally, and perhaps more intuitively, one could fix the plate material and thickness, leaving only the plate dimensions variable[2]. Furthermore, it is useful to introduce the equivalent parameters

$$A = L_x L_y \qquad \alpha = L_y / L_x \quad (47)$$

which are the plate surface area in m^2 and dimensionless aspect ratio respectively. A is useful, as it will scale directly with computational cost, independently of α. Furthermore, it is more intuitive to set the loss parameters in terms of perceptually-relevant decay times $T_{60,0}$ and $T_{60,c}$ at 0 Hz and f_c Hz, respectively as

$$\sigma_0 = \frac{6\ln(10)}{T_{60,0}} \quad \sigma_1 = \frac{6\ln(10)\sqrt{Q/\rho\xi}}{2\pi f_c}\left(\frac{1}{T_{60,c}} - \frac{1}{T_{60,0}}\right), \quad (48)$$

with $T_{60,0} \geq T_{60,c}$. A reduced parameter set $\mathcal{O}'_{\text{plate}}$ results:

$$\mathcal{O}'_{\text{plate}} = \{A, \alpha, T_{60,0}, T_{60,c}\}\,. \quad (49)$$

For the reduced set, A and α must remain fixed over the course of a simulation. It is possible to vary $T_{60,0}$ and $T_{60,c}$, though in this case one must back off slightly from the stability condition given in (33) to accommodate the range of such variations.

4.2. Real-time Implementation

Prototyping was carried out in Matlab, using a sparse vector/matrix representation following directly from the form of the scheme in (28). Such a representation is optimal in terms of performance as well as readability and debugging in Matlab. (The additional biharmonic linear system solution in (36) was carried out using a generic Cholesky factorisation computed outside the runtime loop.)

In order to achieve real-time performance as in, e.g., an audio plug-in format, highly-optimised single-threaded C++ code is required. Previous testing has shown that a direct "matrix-unrolled" approach is up to 10× faster than using a sparse matrix representation directly in C++ [29]. There are two main reasons why unrolling the sparse matrix form and applying stencil operations directly is more efficient. First, the vector/matrix representation, while sparse, is highly redundant with many repeated values, and can be reduced to a very small number of stencil coefficients. Second, in unrolled form the compiler is much more likely to be able to vectorize the code using a suitable optimisation level, and even if not it is relatively simple to manually apply vector intrinsics to the update. Sparse matrix data structures are more complicated in this respect due to the irregular data patterns used in typical reduced memory formats such as CSR (Compressed Spare Row).

For the gong algorithm described here, there are 12 different array operations that are required at each time-step, not including the core element of the linear system solution, as described in Section 3.6. By unrolling the sparse matrix representation and using -O3 optimisation level in Clang, the compiler was able to fully vectorize each array operation with AVX vectors without having to write any manual intrinsics at all. The solver element, however, did require manual application of intrinsics in order to achieve maximum efficiency.

5. NUMERICAL RESULTS

5.1. Timings

For numerical tests, three machines were used: (1) *LinuxLap*: a Linux laptop with an Intel 12th Gen quad-core i7-1260P CPU; (2) *WinPC*: a Windows desktop with an AMD Ryzen 7 8-core 5800X CPU; and (3) *MBA*: MacBook Air with an Intel 10th Gen quad-core i5 CPU; All tests were written in C++ and compiled with -O3 and -mavx2 flags.

Computation times for the complete plate simulation algorithm, for various choices of plate area and aspect ratio on different machines are shown in Table 1. These indicate that computation time tracks the total surface area A reasonably closely, regardless of the aspect ratio, which is as expected. They indicate faster than real time performance for these plate areas, which are small, but definitely within the realm of musical gongs.

Table 2 shows the comparison between computation times for the biharmonic solver presented in Section 3.6 and those for heavily-

[2] In this article, the material is taken to be steel, with $E = 2 \times 10^{11}$ Pa, $\rho = 7850$ kg m^{-3}, and $\nu = 0.3$, and the plate thickness is $\xi = 0.5$ mm.

A (m^2)	α	N_x	N_y	LinuxLap	WinPC	MBA
0.06	1.24	25	31	0.484	0.635	0.908
0.06	0.80	31	25	0.569	0.649	0.993
0.05	1.38	21	29	0.365	0.441	0.636
0.05	0.72	29	21	0.393	0.510	0.695
0.05	2.06	17	35	0.278	0.360	0.761
0.05	1.00	25	25	0.393	0.476	0.815
0.05	3.46	13	45	0.280	0.355	0.664
0.04	1.32	19	25	0.295	0.397	0.562
0.04	0.76	25	19	0.296	0.421	0.653
0.03	1.24	17	21	0.150	0.183	0.264
0.03	2.08	13	27	0.150	0.190	0.287

Table 1: Timings, in s, to compute 1 s output for plates of different areas A and aspect ratios α. Grid sizes N_x and N_y are as indicated.

	14×14	16×20	23×17	25×25
Our solver	**0.054**	**0.135**	**0.118**	**0.279**
LU	0.375	0.652	0.865	1.735
Cholesky	0.254	0.476	0.531	0.978

Table 2: Comparison between computation times, in s, for the biharmonic solver presented in Section 3.6 and alternative solvers from Eigen on WinPC, for typical grid sizes $N_x \times N_y$.

used generic linear system solvers like based on LU and Cholesky decompositions. Here we have used realisations of these solvers from Eigen[30], a well-known high-level C++ library for linear algebra.

Table 3 shows the percentage split between the biharmonic solver element of the timeloop code and the remaining sections. The solver is by far the most significant element, taking up to 76% of the computation time at each time-step.

5.2. Sound Output

It is useful to examine the effect of the plate nonlinearity through spectrograms of sound output. The sample rate is chosen as $F_s = 44.1$ kHz, and the material parameters and thickness are fixed as in the footnote on page 6. Reduced plate parameters are chosen as $\alpha = 1.4$, $T_{60,0} = 20$ s and $T_{60,c} = 10$ s, with $f_c = 1$ kHz. The excitation is of the form of (5), with $t_0 = 0$ s. Spectrograms are calculated using a window size of 2048 points, with a hop size of 128 points and Hann windowing applied.

Consider first a very small plate with $A = 0.01$ m^2, and the effect of increased excitation amplitude $f_{\max}$, where all other parameters are held constant. Output is drawn at the fixed location $\mathbf{r}_o = (L_x/5, 0)$. In this case, the excitation duration is $T = 2$ ms. See Figure 3. Under low-amplitude excitation amplitude, the linear behaviour of the plate is recovered, and distinct constant modal frequencies are observed. At higher amplitudes, effects of pitch glides are observed—these glides are not uniform across all partials, however, as they would be if a simpler model of nonlinear plate vibration (e.g. that of Berger [15]) were used. At very high amplitudes, the partials themselves are replaced by wideband noise—the crash.

Plate size	Biharmonic Solver	Remaining
19×25 $(0.04, 1.2)$	72.3%	27.7%
25×31 $(0.06, 1.2)$	75.3%	24.7%
19×35 $(0.05, 1.8)$	72.4%	27.6%
29×21 $(0.05, 0.7)$	76.2%	23.8%

Table 3: Comparison of computation time for the solver vs remaining elements of the optimised C++ code at each time-step.

As a further illustration, consider now the case of a larger plate, with $A = 0.16$ m^2, again under increasing excitation amplitude, and now with excitation duration $T = 4$ ms. See Figure 4. In this case, the characteristic "swell" of a gong-like instrument may be observed—a slow migration of energy to the high frequency range over the first several hundred milliseconds.

As a final example, consider a comparison between spectrograms of sound output for a small plate with $A = 0.01$ m^2, and with an excitation amplitude $f_{\max} = 20$ N and duration $T = 4$ ms, in the presence of time-varying monophonic output, drawn from an ellipse with $R = 0.4$, and with a scan frequency of 1 Hz. See Figure 5. Easily visible are complex modulations of the individual frequency components, characteristic of that which occurs in an instrument that may be free to exhibit rigid-body oscillation relative to the listener.

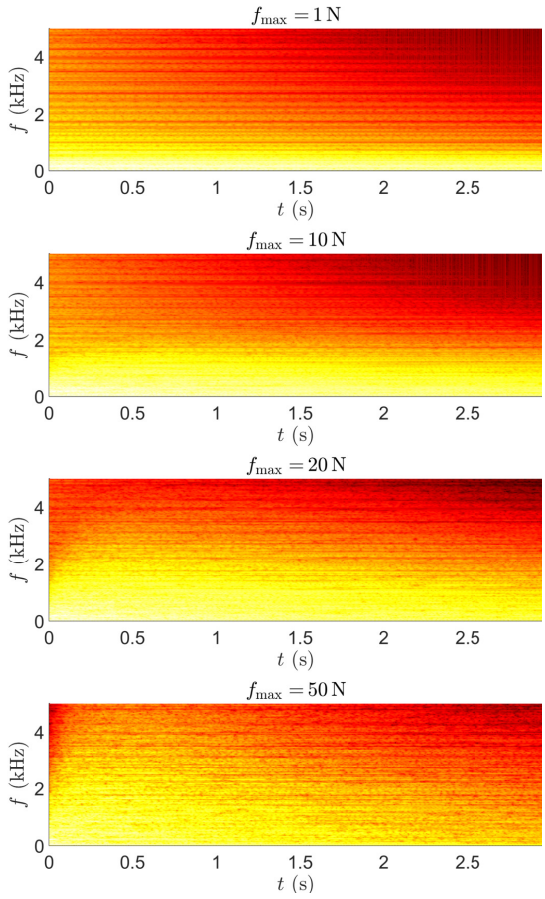


Figure 4: *Spectrograms of output from a large plate model, at different excitation amplitudes, as indicated.*

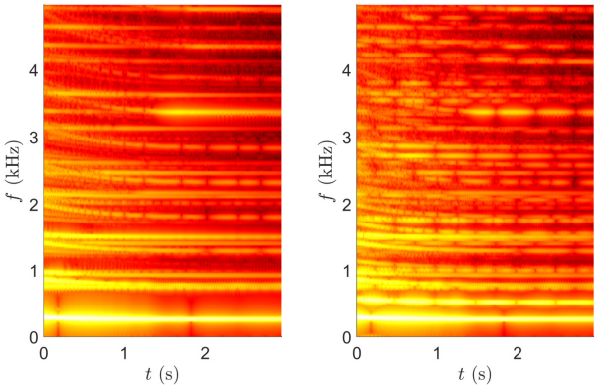


Figure 5: *Spectrograms of output from a small plate model, with a static output location (left), and a time-varying output location (right).*

6. CONCLUDING REMARKS

In this paper, it has been shown that it is possible to implement computationally-intensive physical models in real time—in this case the Föppl-von Kármán model of high amplitude plate vibration that is necessary in order to emulate gongs. Due to the complexity of the system, achieving real-time performance requires optimisation at multiple levels. First: the design of a numerically stable explicit integrator, as presented here, has been the key to breaking the real-time barrier. The computational advantage here hinges on the exploitation of matrix structure (in this case, rank-1 perturbation of the identity—a general property of SAV designs for Hamiltonian or near-Hamiltonian systems [12]) But even when this bottleneck has been removed, there remains the problem of linear system solution (of the biharmonic operator) in the run-time loop. Fast solution has been approached by exploiting a different type of matrix structure (block Toeplitz in this case). This type of acceleration is much more targeted at the particular case of the Föppl-von Kármán system. Further acceleration depends on the use of low-level parallelisation tools. The larger lesson here is that for physical modeling synthesis from any reasonably complex system, a silver bullet is likely not available—rather, in order to achieve good performance, great attention must be paid to the specifics of the system at hand.

Many simplifications to more realistic models of percussion instruments have been made in order to arrive at a real-time implementation. Among these are: a) the restriction to a rectangular geometry with simply-supported conditions; b) the assumption of a flat plate rather than a curved shell, which is more usual; c) the assumption of a uniform thickness; and d) the consolidation of effects of loss due to various mechanisms (radiation, viscothermal) to a basic two-parameter loss model. Including any of these effects would have no impact on the explicit integration method, provided the system may still be written in terms of the extension of a Hamiltonian system. On the other hand, the block-Toeplitz solver is highly dependent on restrictions a) to c). Coping with restriction d) would necessarily increase the temporal order of the scheme as a whole, leading to a larger footprint in terms of both memory usage (not a major concern here) as well as computational cost.

7. REFERENCES

[1] N Firasta, M. Buxton, P. Jinbo, K. Nasri, and S. Kuo, "Intel AVX: New frontiers in performance improvements and energy efficiency," *Intel White Paper*, 2008.

[2] C. Touzé, O. Thomas, and A. Chaigne, "Asymmetric non-linear forced vibrations of free-edge circular plates, part I: theory," *J. Sound Vib.*, vol. 258, no. 4, pp. 649–676, 2002.

[3] O. Thomas, C. Touzé, and A. Chaigne, "Non-linear vibrations of free-edge thin spherical shells: Modal interaction rules and 1:1:2 internal resonance," *Int. J. Solids Structures*, vol. 42, pp. 3339–3373, 2005.

[4] S. Bilbao, "Sound synthesis for nonlinear plates," in *Proc. 8th Int. Conf. Digital Audio Effects*, Madrid, Spain, Sept. 2005, pp. 243–248.

[5] E. Hairer, C. Lubich, and G. Wanner, "Geometric numerical integration illustrated by the Störmer–Verlet method," *Acta Numerica*, vol. 12, pp. 399–450, 2003.

[6] S. Bilbao, "A family of conservative finite difference schemes for the dynamical von Kármán plate equations," *Num. Meth. Partial Diff. Eq.*, vol. 24, no. 1, pp. 193–216, 2008.

[7] X. Yang, J. Zhao, and Q. Wang, "Numerical approximations for the molecular beam epitaxial growth model based on the invariant energy quadratization method," *J. Comp. Phys.*, vol. 333, pp. 104–127, 2017.

[8] J. Zhao, "A revisit of the energy quadratization method with a relaxation technique," *Appl. Math. Lett.*, vol. 120, pp. 107331, 2021.

[9] Z. Liu and X. Li, "The exponential scalar auxiliary variable (e-sav) approach for phase field models and its explicit computing," *SIAM J, Sci. Comp.*, vol. 42, no. 3, pp. B630–B655, 2020.

[10] M. Jiang, Z. Zhang, and J. Zhao, "Improving the accuracy and consistency of the scalar auxiliary variable (SAV) method with relaxation," *J. Comp. Phys.*, vol. 456, pp. 110954, 2022.

[11] S. Bilbao and M. Ducceschi, "Fast explicit algorithms for Hamiltonian numerical integration," in *Proc. Eur. Nonlinear Dynamics Conf.*, Lyon, France, July 2022.

[12] S. Bilbao, M. Ducceschi, and F. Zama, "Explicit exactly energy-conserving methods for Hamiltonian systems," *J. Comp. Phys.*, vol. 427, pp. 111697, 2023.

[13] K. Graff, *Wave Motion in Elastic Solids*, Dover, New York, New York, 1975.

[14] R. Mindlin, "Influence of rotatory inertia and shear on flexural motions of isotropic elastic plates," *J. Appl. Mech.*, vol. 18, pp. 31–38, 1951.

[15] H. Berger, "A new approach to the analysis of large deflections of plates," *J. Appl. Math.*, vol. 22, pp. 465–472, 1955.

[16] A. Föppl, *Vorlesungen über technische Mechanik*, Druck und Verlag von B.G. Teubner, Leipzig, 1907.

[17] T. von Kármán, "Festigkeitsprobleme im maschinenbau," *Encyklopädie der Mathematischen Wissenschaften*, vol. 4, no. 4, pp. 311–385, 1910.

[18] A. Nayfeh and D. Mook, *Nonlinear Oscillations*, John Wiley and Sons, New York, New York, 1979.

[19] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, John Wiley and Sons, Chichester, UK, 2009.

[20] Q. Nguyen and C. Touzé, "Nonlinear vibrations of thin plates with variable thickness: Application to sound synthesis of cymbals," *J. Acoust. Soc. Am.*, vol. 145, pp. 977–988, 2019.

[21] L. Rhaouti, A. Chaigne, and P. Joly, "Time-domain modeling and numerical simulation of a kettledrum," *J. Acoust. Soc. Am.*, vol. 105, no. 6, pp. 3545–3562, 1999.

[22] B. Hosseini, N. Nigam, and J. Stockie, "On regularizations of the Dirac delta distribution," *J. Comp. Phys.*, vol. 305, pp. 423–447, 2016.

[23] O. Thomas and S. Bilbao, "Geometrically nonlinear flexural vibrations of plates: In-plane boundary conditions and some symmetry properties," *J. Sound Vib.*, vol. 315, no. 3, pp. 569–590, 2008.

[24] J. Sherman and W. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Ann. Math. Stat.*, vol. 21, no. 1, pp. 124–127, 1950.

[25] Z. Wang and M. Puckette, "A fast algorithm for the inversion of the biharmonic in plate dynamics applications," in *Proceedings of the 5th Stockholm Music Acoustic Conference*, Stockholm, Sweden, June 2023.

[26] L. Thomas, "Elliptic problems in linear differential equations over a network: Watson scientific computing laboratory," Tech. Rep., Columbia Univ., 1949.

[27] B. Buzbee, G. Golub, and C. Nielson, "On direct methods for solving Poisson's equations," *SIAM J. Num. Anal.*, vol. 7, no. 4, pp. 627–656, 1970.

[28] M. Wax and T. Kailath, "Efficient inversion of Toeplitz-block Toeplitz matrix," *IEEE Trans. Acoust. Speech Sig. Proces.*, vol. 31, no. 5, pp. 1218–1221, 1983.

[29] C. Webb and S. Bilbao, "On the limits of real-time physical modelling synthesis with a modular environment," in *Proc. 18th Int. Conf. Digital Audio Effects*, Trondheim, Norway, Sept. 2015, pp. 65–72.

[30] G. Guennebaud and B. Jacob et al., "Eigen v3," http://eigen.tuxfamily.org, 2010.

OPTIMIZATION TECHNIQUES FOR A PHYSICAL MODEL OF HUMAN VOCALISATION

Mateo Cámara
Information Processing & Telecomm. Center
Universidad Politécnica de Madrid
Madrid, Spain
mateo.camara@upm.es

Zhiyuan Xu
Centre for Digital Music
Queen Mary University of London
London, UK
zhiyuan.xu@qmul.ac.uk

Yisu Zong
Centre for Digital Music
Queen Mary University of London
London, UK
y.zong@qmul.ac.uk

José Luis Blanco
Information Processing & Telecomm. Center
Universidad Politécnica de Madrid
Madrid, Spain
jl.blanco@upm.es

Joshua D. Reiss
Centre for Digital Music
Queen Mary University of London
London, UK
joshua.reiss@qmul.ac.uk

ABSTRACT

We present a non-supervised approach to optimize and evaluate the synthesis of non-speech audio effects from a speech production model. We use the Pink Trombone synthesizer as a case study of a simplified production model of the vocal tract to target non-speech human audio signals –yawnings. We selected and optimized the control parameters of the synthesizer to minimize the difference between real and generated audio. We validated the most common optimization techniques reported in the literature and a specifically designed neural network. We evaluated several popular quality metrics as error functions. These include both objective quality metrics and subjective-equivalent metrics. We compared the results in terms of total error and computational demand. Results show that genetic and swarm optimizers outperform least squares algorithms at the cost of executing slower and that specific combinations of optimizers and audio representations offer significantly different results. The proposed methodology could be used in benchmarking other physical models and audio types.

1. INTRODUCTION

Articulatory synthesis provides a unique opportunity to delve into the mechanics of speech production [1, 2]. Unlike black box models, physical models achieve an *interpretable representation* of the inner characteristics of the vocal tract. This allows for a deeper understanding of the processes involved in speech production. They also provide precise control of the speech's articulatory, resonance, and phonatory characteristics, such as the position of the tongue, lips, existing constrictions, or nose size; as well as informed control of model parameters. This makes natural-sounding synthetic speech samples less prone to artifacts than other synthesis models. Furthermore, they may produce any type of human sound coming out of the mouth and the nose. Those include sounds that are not words, such as sighs, laughs, yawns, and so on.

These *non-speech sounds* are becoming increasingly important in today's audiovisual productions and digital interactions. From the sound effects in movies and videogames to the soundscapes in podcasts and audiobooks [3, 4, 5], the ability to generate these sounds has become a critical aspect to produce realistic performances. Analyzing the ability of models to construct these types of sounds is crucial to understand the limits and limitations of models [6], as well as the underlying complexities of producing naturally sounding audio samples. Answering those questions opens up new possibilities for sound and user-experience designers, video-game developers, and audio production professionals looking for new and innovative ways to create high-quality, realistic, and engaging sound experiences.

Physical models for speech synthesis pose challenges that are extensively reported in the literature. They often include many parameters that are difficult to configure simultaneously to achieve high-quality sounds. Their combined optimization can be demanding, computationally expensive, time-consuming, and challenging to implement in real time. These complications explain the need to improve and optimize the synthesizer.

Our research focuses on articulatory parameters from a black-box point of view. We optimize synthesizers without paying specific attention to what each parameter represents to maximize objective similarity by minimizing the difference between a target signal and the synthesized signal. This ensures superior generalization capabilities for the proposed method and valuable results for other contexts.

In this contribution, we look at the physical model known as the *Pink Trombone (PT)*[1]. This is a simplified version of the vocal tract that uses a small set of fundamental parameters to control the shape and movements of the articulators during speech production [7]. We fixed its articulatory bounds to focus on sounds that a human could physically produce, and used these to optimize the PT and compare its result with human audio samples.

We conduct a case study using synthetic, sustained, and yawning sounds to understand its capabilities and limitations. We test different black-box strategies to predict the synthesizer control parameters, including well-known *optimization techniques* and *Deep Neural Networks*, trained on a set of PT synthetic samples. For experimentation, we use sound files generated by the PT, as well as audio clips downloaded from the Freesound platform [8].

Experiments shall lay the foundations for studies on articulatory and production models with multiple parameters and different

[1] https://dood.al/pinktrombone/

audio types. The dataset, test sounds, and algorithms are available online[2]. Our goals for this contribution are the following:

- *Determine if PT parameters can be accurately predicted exclusively from acoustic features*. We optimize synthesizer control variables from audio samples as a black-box.
- *Determine best optimization technique for articulatory variables*. We evaluate how different optimizers perform in front of increasingly challenging sounds.
- *Determine the error metric that yields a more satisfactory outcome.* We benchmark different techniques for standard error metrics and acoustic parameterizations of audio files.

The remainder of this paper is organized as follows. Sec. 2 expands on the optimization techniques and the parameterizations reported in the literature. Sec. 3 describes the experiments covered to meet our objectives. Sec. 4 analyzes and discusses the results obtained, and Sec. 5 concludes the paper.

2. BACKGROUND

For sound-matching optimization, one may focus on the control parameters of the synthesizer, the input acoustic features extracted from the audio, and the process that leads to optimization. All these aspects provide insights into the methodologies and objectives of various optimization studies in synthesizers. Fig. 1 depicts the overall schematic of the optimization process.

2.1. Optimization Methods

Considering the complexity of sound synthesizers, there is a need for reliable optimization techniques. Numerous optimization methods have been investigated in terms of optimizing parameters for physical models or traditional synthesizers. The related work can be organized into two main categories:

Search-based Methods: these have been widely applied to physical models in audio synthesis due to their ability to handle non-differentiable, non-linear, and non-convex optimization problems. They are universal and regard the synthesizer as a black-box model, focusing solely on parameter space optimization. Standard ways include the use of Evolutionary Algorithms (EA), including Evolution Strategies [9], Genetic Algorithm (GA) [10], or Particle Swarm Optimization (PSO) [11]. Other methods, including Hill Climber [12], Levenberg–Marquardt Algorithm [13] and Nelder-Mead Method [14] are also considered.

Model-based Methods: machine learning (ML) models have become mainstream for synthesizer parameter estimation in recent years. They learn the mapping between the latter and audio features directly from data. In [15], authors used a strided Convolutional Neural Network (CNN) to predict the parameters of a subtractive synthesizer. Recent work proposed differentiable digital signal processing (DDSP) [16] and integrated an additive synthesizer with a filtered noise synthesizer into the end-to-end deep learning framework. These allow direct gradient descent optimization. DDSP is now widely utilized for parameter estimation [17], despite its need for precise reproduction of the target synthesizer in a differentiable manner, which poses difficulties.

[2]https://slash-trombone.github.io/

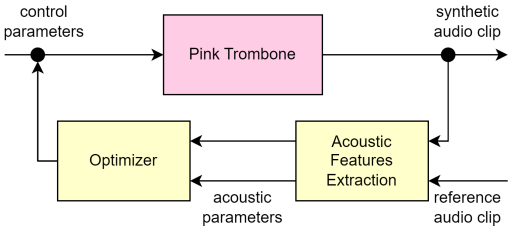


Figure 1: *Schematic on the optimization process.*

Each approach has its own benefits and limitations, leading to ongoing discussions. In [12], authors compared sound-matching performance on a VST synthesizer using two search strategies and three neural network methods. Results indicated that search methods are limited by their computational cost, and modeling methods are restricted to the inductive bias of model structure and data availability. We tested these limitations for the PT, including simple speech and non-speech vocalisations to evaluate the performance of the optimized model parameters to reproduce sounds.

2.2. Parameter Selection

The control parameters of the synthesizer and the input parameters for the optimizer largely depend on the synthesis technique and the desired outcomes, in accordance with Fig. 1. We focus on the PT control parameters –see Table 1. Physical models may alternatively use local constrictions to describe the configuration required for the vocal tract to produce a certain sound. The PT can actually operate on those as well. Nonetheless, we are interested in the primary ones.

Furthermore, inputs to the optimizer must represent the acoustic content of the audio samples so that the model may produce accurate outputs. For this task, we shall look at acoustic features.

2.3. Acoustic Features Extraction

Various acoustic features have been used in synthesizer optimization studies to evaluate and quantify the quality of the synthesized sounds to guide the optimization process. Spectral features are the most common, but finding the best metric with good perceptual consistency is still an open question [18]. In [19] authors focus on the spectral norm error, [10] used spectral norm plus spectral centroid error extracted from short-time Fourier transform (STFT) for each frame, [9] used relative spectral error, which is computed by summing normalized differences between frequency components extracted from two spectra, [20] combined the least squared error of the STFT of two sounds plus the perceptual error applying a narrow band masking curve. On error computation, [12] used Euclidean distance of Mel-Frequency Cepstral Coefficients (MFCCs), [16] used a deep representation extracted from MFCCs, and multiscale spectral loss plus perceptual loss, [15] compared the following features as the Deep Neural Networks (DNN) input: a set of spectral features [21], STFT, and deep representation extracted by a CNN from the raw signal. Results showed that STFT and deep representations seem more representative than handcrafted features. Our aim now is to identify suitable ones for the PT parameters optimization.

3. EXPERIMENTATION

We designed our experiments to focus on three specific characteristics that are relevant to the acoustic-to-articulatory inversion: the *optimizer* used to predict control variables, the *audio representation* to compute the difference between original and synthesized audio, and the *signal complexity*.

3.1. Pink Trombone Fundamentals

The PT is a simple vocal tract model that can be interacted with through a web interface. It is a Kelly-Lochbaum (KL) type model whose technical details can be found in [22]. In our black-box case study, we focused on the number of parameters to be used and their bounds. Table 1 summarizes these, which correspond to physical attributes of the vocal tract. We aimed to decouple the meaning of these parameters from their human meaningfulness for our method to be useful to any synthesizer.

Table 1: *Pink Trombone parameters and their bounds.*

Pink Trombone Parameters	Lower bound	Upper bound
Pitch (Hz)	75	330
Voiceness	0	1
Tongue Index	14	27
Tongue Diameter (cm)	1.55	3
Lips Diameter (cm)	0.6	1.2
Constriction index	12	42
Constriction Diameter (cm)	0.6	1.2
Throat Constriction (cm)	0.5	1.0

3.2. Signal complexity

Signal complexity refers to the challenges we pose to the optimizers to predict the exact parameters. In that sense, we consider three independent characteristics of the signal. First, the *origin of the audio file*: audio generated by a speech synthesizer or a person. Second, *variations over time*: sustained notes or dynamic audio (such as a yawn). Third, *number of variables to optimize*: related to the characteristics of the synthesizer. Hereafter we enumerate all experiments conducted from the least to the most complex.

- PT generated sounds for which:
  - One of the control parameters is unknown.
  - All of the control parameters are unknown.
  - Gaussian white noise is added. This evaluates the robustness of optimizers dealing with non-ideal signals.
  - Control parameters vary over time.
- Audio clips containing:
  - Sustained vowel sounds.
  - Yawnings.

3.3. Audio Representation and Quality Assessment

3.3.1. Representations focusing on spectral difference

To minimize the difference between the target and reconstructed sound, we focused on the spectral features of the audio signals. The following list includes all the transformations evaluated:

- *STFT*. A window size of 1024 samples with a 2x overlap STFT was taken.
- *Multiscale spectrogram*. The window sizes were {64, 128, 256, 512, 1024}, with a 75% overlap.
- *MEL-spectrogram*. We used 128 filters in the MEL bank up to a maximum frequency of 8 KHz.
- *MFCCs*. We took 20 cepstral coefficients from the MEL-spectrograms.

Computations were performed in Python 3.9, using the AuraLoss library [23]. The Mean Absolute Error (MAE) between the input and reconstructed audio was computed as the error function.

3.3.2. Perceptual metrics

In addition to MAE, we also computed a set of perceptual quality and intelligibility metrics. These metrics were not used as error functions in the optimization process. The findings may be representative of the perceptual similarity between sounds. However, we encourage readers to listen to the results we posted online. The following full reference metrics were analyzed:

- PESQ: Perceptual Evaluation of Speech Qlt. [24].
- PEAQ: Perceptual Evaluation of Audio Qlt. [25].
- ViSQOL: Visually-Inspired Speech Qlt. Obj. Listener [26].
- STOI: Short-Time Objective Intelligibility [27].

3.4. Selected Optimizers

We used *optimization algorithms* and a *CNN* to predict the control parameters of the synthesizer. We fed the algorithms with the MAE between the original and the synthesized signal. Hereafter we briefly introduce the selected optimization algorithms, which we have evaluated in terms of computational cost and reconstruction error.

Genetic Algorithm (GA): is an optimization technique inspired by natural selection and genetics [28]. The candidate solutions are defined by a set of genes. In every generation (loop over all candidates), the genes are able to randomly change (mutation), combine with other candidates (crossover), and be selected (optimization) to search for optimal solutions in the solution space. The fitness function seeks to minimize differences in the input/output signals.

We used 32 bits to define the genes, a crossover rate of 0.9, a mutation rate of 0.03, and a population of 10 individuals.

Particle Swarm Optimization (PSO): is a nature-inspired metaheuristic optimization technique that simulates the social behavior of swarms [29]. PSO operates by iteratively adjusting the position of particles within the search space based on their individual and global best experiences, converging towards the optimal solution. In our case, we set acceleration parameters $c_1 = 0.5$, $c_2 = 0.3$ (trust in itself, trust in its neighbors), and inertia weight $w = 0.9$, with 10 particle population.

Trust Region reFlective (TRF): The Trust Region reFlective [30] algorithm is a computational technique for solving least squares optimization problems. It employs a model-based method, seeking to minimize a function by iteratively creating simplified models of

the objective function within certain trusted regions. The term "reflective" refers to the method's way of handling boundaries and constraints: if a proposed step hits a boundary, it is reflected in the feasible region.

Nelder-Mead Method (NM): also known as the downhill simplex method [31], is a multidimensional optimization technique well suited for non-linear problems. The algorithm starts with an initial simplex, a set of $n+1$ points in an n-dimensional space. The algorithm iteratively updates the position of the simplex by reflecting, expanding, contracting, or shrinking it, based on the values of the function being optimized at the vertices of the simplex.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES): is a stochastic optimization algorithm that uses information about the distribution of the samples generated by the algorithm to guide the search for the optimal solution [32]. The algorithm starts with an initial guess for the solution and then generates a set of samples around this point. The distribution of these samples is then updated based on the fitness of the samples, with higher-fitness samples being more likely to be selected. As the algorithm progresses, outcomes increasingly concentrate around the optimal solution.

Neural network prediction (NN): In addition to the optimization techniques, we tested the capability of neural networks to predict the control variables based on the acoustic features. We designed a CNN that admits spectrograms as input and outputs the control variables. To train it we collected a database of 400,000 different PT clips. We trained four different networks, each admitting as input for each audio representation mentioned in subsection 3.3.1. The network has been coded with Pytorch 1.7.1, with 2 convolutional layers, ReLU as the activation function, 0.0001 as the learning rate, ADAM optimizer, and following the 60/20/20 data splitting strategy between training, validation, and test.

3.5. Materials

Attending to the scope of this contribution, the assessment of the performance achieved by the different techniques, audio representations, and optimizers in predicting the parameters of the physical model for sound-matching required two sets of audio files:

- Synthetic audio samples, generated at 48 kHz sampling rate and 1 s long. To generate these, we used the Programmable version of the PT[3] modified to be a Node.js server. We generated 80 audio clips with random control parameters.
- Audio samples downloaded from Freesound containing utterances from different speakers. We focused on sustained vowels (5 clips) and yawnings (8 clips). The vowels are one second long and the yawnings are three seconds on average. All files were recorded at a 48 kHz sampling rate to match the same conditions as the synthetic audio.

4. RESULTS

In this section, we present the results of the experiments aimed at predicting control parameters for PT. We sought to fix the same conditions for all optimizers to ensure a fair comparison. Some considerations apply to all experiments:

[3]https://github.com/zakaton/Pink-Trombone

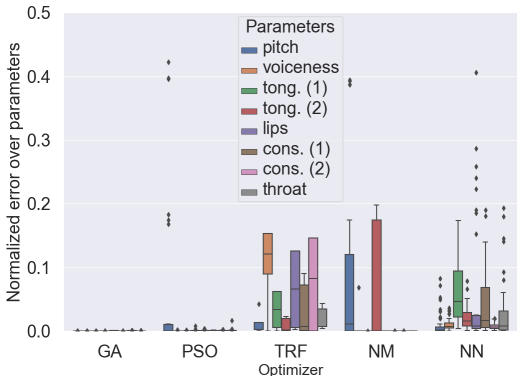


Figure 2: *Optimizer performance over one control parameter. X-axis includes the optimizers. Y-axis represents the normalized error. Each bar is a control parameter.*

- *Error values are normalized* with respect to the maximum and minimum values that each parameter can take.
- The *random seed was fixed* to randomize the PT control parameters in each experiment, such that the optimizers face the same initial conditions in all cases.
- Each experiment was *repeated 20 times*. Initial conditions and target values were randomized.
- All optimizers had the same *stop criterion*: reach an error of less than 0.0001 in the metric or stop to improve the relative error with respect to the previous 20 loops.

4.1. Optimization of PT-generated sounds

Hereafter we present the results of the different tests that were conducted using PT synthetic audio clips as inputs.

4.1.1. Optimization of one control parameter

In this experiment, we fixed all control parameters except for one. We predicted the unknown value. This set of experiments does not include the CMA-ES algorithm because its particular design does not support single-parameter prediction. The results are shown in Figure 2, for the different optimizers and PT control parameters.

Results demonstrate the effectiveness of GA and PSO in accurately predicting the control parameters. No outliers were observed in the genetic algorithm's performance. The NM algorithm successfully reached the absolute minimum for most parameters. However, it struggled to achieve the same for the pitch and one tongue-related parameter. A closer examination of these parameters revealed that their error functions contained multiple local minima. Since the performance of the NM is heavily influenced by its initial conditions, it makes it prone to getting stuck in them.

Despite not always arriving at the optimal values, TRF and NN converge rapidly to the minimum. Once it is trained, NN takes less than a second to reach the minima. TRF algorithm takes 5 seconds on average, which is four times faster to optimize than PSO and NM, and 100 times faster than GA.

In the same line, Figure 3 illustrates the error associated with each audio representation. All audio representations are suitable for optimizing individual control parameters. However, no error is observed in the multiresolution. This makes sense, since it is an extension of the STFT that better represents the spectral characteristics of the signal.

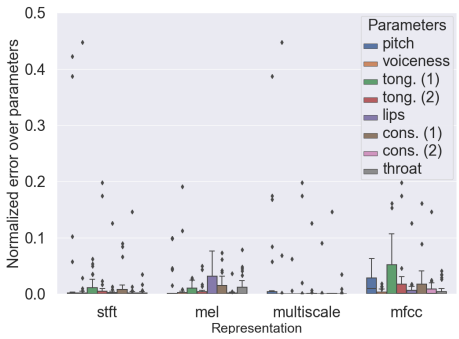


Figure 3: *Audio representation performance over one control parameter. X-axis represents each audio representation. Y-axis represents the normalized error. Each bar is a control parameter.*

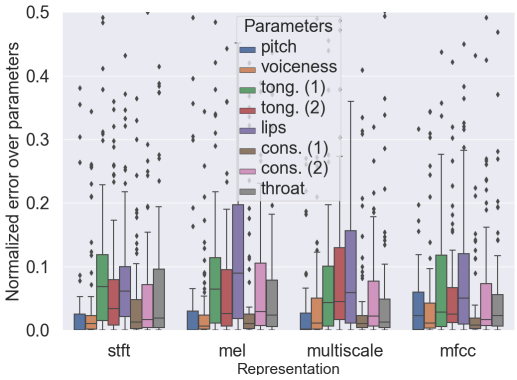


Figure 5: *Audio representation performance while predicting all parameters at a time. X-axis covers the representations. Y-axis represents the normalized error. Each bar corresponds to a control parameter.*

4.1.2. All control parameters

The single-parameter experiments validate that articulatory parameters can be predicted from sound representations alone. However, this scenario is too simplified to clarify which optimizer is more accurate. This can be done by increasing the complexity of the experiment, seeking to predict all control parameters at once. The prediction results for each parameter are shown in Figure 4.

Experiments focusing on predicting all parameters demonstrate the superior performance of GA, CMA-ES, and PSO compared to other methods. In this experiment set, the eight-dimensional search area makes the optimization more challenging. The TRF and NM algorithms yielded unsatisfactory results, deeming them unsuitable for tackling the problem. As the number of potential solutions grows exponentially with increasing dimensions, only the most robust methods can find an optimal solution. Genetic algorithms and PSO can outperform least squares minimization or the downhill simplex method because they are more robust in handling complex search spaces, non-convex functions, and intricate relationships between variables.

On the other hand, observing how the different audio representations behave in this scenario is interesting. They are shown in Figure 5. We can observe that no representation performs significantly better than the rest, not even the multiresolution representation. However, finding a higher error is not necessarily a serious problem when reconstructing the signal. Most control parameters have local minima very close to the global minimum. This means that different PT configurations can produce almost the same reconstruction. This does not apply to the pitch parameter, which is one of the critical parameters in quality and, as can be seen, the MFCCs do not make it easy to reach its minimum.

In fact, Figure 6 illustrates precisely these phenomena. It shows the MAE of the original and reconstructed signal. It is observed that regardless of the optimizer used when the search space is large, the MFCCs do not achieve satisfactory results. Thus, this experiment indicates that the best prediction of the control parameters can be made with GA or the PSO using the MEL scale or Multiresolution spectrograms.

In addition, Figure 7 shows the computational costs of each optimizer. It shows that the NN is the fastest once trained, while PSO is the fastest of the suitable optimization techniques.

4.1.3. All control parameters which vary over time

The next level of complexity we tested was optimizing parameters that varied over time. To achieve this, we created an interpolator that generated intermediate values between two temporal spaces

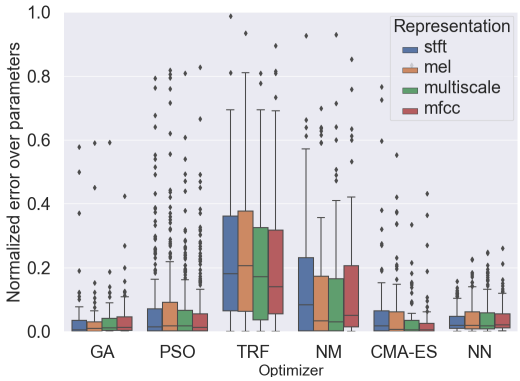


Figure 4: *Optimizer performance when all parameters are predicted at the same time. X-axis includes all optimizers. Y-axis represents the normalized error. Each bar corresponds to an audio representation.*

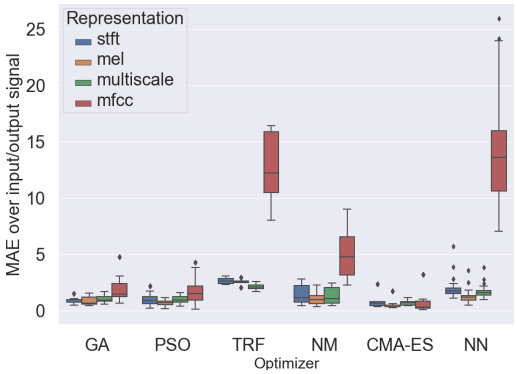


Figure 6: *Absolute performance of the optimizers and representations. X-axis includes all optimizers. Y-axis represents the MAE of the target and reconstructed audio file. Each bar is the audio representation.*

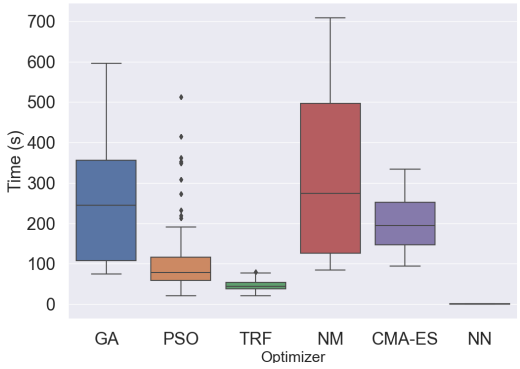


Figure 7: *Computational cost for the different optimizers –in X-axis. Y-axis represents the time to converge, in seconds.*

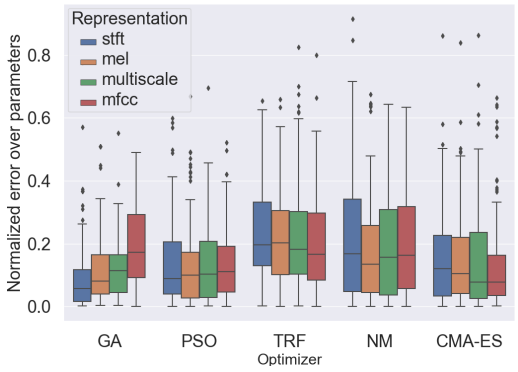


Figure 8: *Performance of optimizers and representations when the signal varies over time. X-axis includes all audio optimizers. Y-axis represents the normalized error between the target and reconstructed audio. Each bar corresponds to an audio representation.*

defined by two sets of articulatory parameters in the PT. As shown in Figure 8, none of the optimizers were able to achieve a satisfactory result when optimizing a time-variant set of parameters. The only optimizer that achieved a result closer to zero was the GA, using the STFT. The tendency is that as more parameters are added to optimize, the search space becomes more complex and therefore very difficult to reach the absolute minimum. It is important to note that this method is not suitable for a neural network. It would be necessary to train new networks depending on the number of parameters to be predicted.

To achieve a more satisfactory, general solution, it was decided that the best strategy for optimizing signals that vary over time is to segment the signal into small windows and optimize them as if they were a static signals. We tested different window sizes and found out a 100 milliseconds length performed optimally. These windows can then be connected using a Savitzky-Golay filter [33], which smooths out the result. The optimization results of these tests suggest insights equivalent to predicting a non-variant set of parameters. That is why this technique is the preferred choice for optimizing sounds created by humans.

4.1.4. All control parameters in noise

In these experiments, different amounts of Gaussian white noise were added to the original signal. We sought to predict the articulatory parameters that defined the signal. As shown in Figure 9, the optimizers performance deteriorates as more noise is added. Additionally, we observed that the optimizers do not start to exhibit exponentially growing errors until the signal-to-noise ratio reaches 20 decibels. All optimizers act similarly, with the exception of the NN, which does not tolerate noise at its input.

From these experiments, we can conclude that it is possible to optimize signals that are not perfectly generated by a synthesizer but may come from any source, such as a recording from a public database. This finding is significant because it suggests that our approach can be applied in real-world scenarios where the input signals will likely not be perfectly recorded.

4.2. Optimization of real audio files

Results of the tests with real sounds can be seen in Table 2. The results for each perceptual metric are shown for the best-performing combination of the optimizer-representation pair. The columns detail the optimizers and the color shows the best audio representation for each case. We used different perceptual metrics to measure how similar the sounds generated by the synthesizer were to human-generated ones. We also include how the perceptual metrics behaved when predicting PT samples. These set up a benchmark to compare the upper limit that could be reached. Still, we encourage readers to visit our website, where we have published these audio files, and evaluate the quality themselves.

As shown in the table, CMA-ES and GA achieved superior results compared to other optimizers in terms of perceptual similarity in most of the cases. It is important to note that the MOS (Mean Opinion Score) equivalent results can still be considered low compared to the scale, as the sounds are synthesized by a certain vocal tract that may not correspond to the vocal tracts of the people who generated the original sound. Therefore, we do not claim that we can produce an exactly identical sound but an equivalent one.

For all types of signal, we used the strategy of dividing the signal into small windows and smoothing them out into full-length signals. Systematically, PT-generated sounds are predicted with better scores than human-generated sounds. Furthermore, in many of the cases we found that yawns are perceptually recognized as more similar than sustained vowels. This is because the timbre in the sustained vowel has a much greater influence than in the yawn. The PT has vocal characteristics that do not match those of the

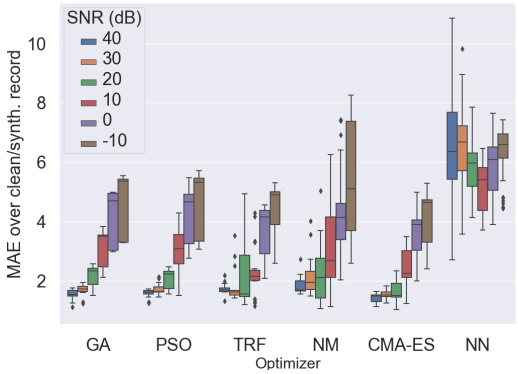


Figure 9: *Performance of the optimizers when Gaussian White Noise was applied at the input. X-axis includes allaudio optimizers. Y-axis represents the MAE between the target and the reconstructed audio file. Each bar is a different Signal-to-Noise ration.*

Table 2: *Perceptual equivalent metrics of the real sounds. Type "PT" stands for "Pink Trombone" generated, "VW" for sustained "Vowel", and "Y" for "Yawn". All metrics are in MOS scale (from 1 to 5) except STOI (from 0 to 1). We used a color code to indicate the best-performing set of acoustic parameters per audio type and optimizer.*

		GA	PSO	TRF	NM	CMA-ES	NN	Best	Result
PESQ	PT	2.2 ± 0.9	2.1 ± 1.2	1.8 ± 1.0	2.2 ± 1.0	2.6 ± 0.9	1.8 ± 0.9	CMA-ES	2.6 ± 0.9
	VW	1.8 ± 0.8	1.5 ± 0.4	1.6 ± 0.6	1.8 ± 0.7	1.5 ± 0.5	1.8 ± 1.2	GA	1.8 ± 0.8
	Y	1.5 ± 0.4	1.3 ± 0.3	1.4 ± 0.2	1.3 ± 0.1	1.3 ± 0.2	1.5 ± 0.7	GA	1.5 ± 0.4
PEAQ	PT	3.0 ± 0.6	3.2 ± 0.8	2.6 ± 0.6	3.0 ± 0.9	3.5 ± 0.7	3.0 ± 0.8	CMA-ES	3.5 ± 0.7
	VW	2.8 ± 0.7	2.2 ± 0.1	2.6 ± 0.7	2.5 ± 0.7	2.5 ± 0.6	2.6 ± 0.7	GA	2.8 ± 0.7
	Y	2.5 ± 0.8	3.0 ± 1.1	2.7 ± 0.7	2.6 ± 0.7	2.7 ± 1.0	2.8 ± 1.0	PSO	3.0 ± 1.1
ViSQOL	PT	3.1 ± 0.9	3.4 ± 0.8	1.7 ± 0.5	3.3 ± 1.2	4.3 ± 0.7	3.0 ± 0.6	CMA-ES	4.3 ± 0.7
	VW	1.9 ± 0.5	2.0 ± 0.2	2 ± 0.3	1.9 ± 0.3	2.1 ± 0.4	1.8 ± 0.4	CMA-ES	2.1 ± 0.4
	Y	2.1 ± 0.1	2.1 ± 0.1	2.3 ± 0.3	2.3 ± 0.4	2.1 ± 0.1	1.9 ± 0.2	TRF	2.3 ± 0.3
STOI	PT	0.3 ± 0.2	0.4 ± 0.3	0.1 ± 0.1	0.5 ± 0.4	0.5 ± 0.3	0.2 ± 0.1	CMA-ES	0.5 ± 0.3
	VW	0.1 ± 0.1	0.1 ± 0.0	0.1 ± 0.0	0.1 ± 0.1	0.1 ± 0.0	0.1 ± 0.0	CMA-ES	0.1 ± 0.0
	Y	0.3 ± 0.1	0.3 ± 0.1	0.3 ± 0.1	0.3 ± 0.1	0.3 ± 0.1	0.1 ± 0.1	-	0.3 ± 0.1

Legend
mel
mfcc
multiscale
stft

person who recorded the sounds. For this reason, it is more difficult to recreate a perfectly harmonic voice like the vowel than a noisy sound like the yawn. This does not imply that our optimizers are malfunctioning, as the goal is to create comparable sounds, not exactly the same. The STOI metric in this regard is very representative of this situation, giving the yawn almost equal score to the PT-generated values, while the vowel is perceived as different.

These experiments also yield two interesting insights. First, one may identify optimizer-representation combinations that perform better than others. In particular, multiscale representation works well for yawns, while for sustained vowels STFT representation can do the job. None performed well using MFCC. Thus, one may need to take into account the type of signal to get good results from the optimizer. Second, there is consistency among the perceptual metrics. Those experiments that are more challenging consistently score worse than simpler ones.

5. CONCLUSION

Optimization techniques effectively predict the parameters of the Pink Trombone to produce human-like vocalisations. The selected algorithms delivered tuned control parameters while operating on different acoustic features and metrics. The resulting audio samples match the selected input sounds regarding the absolute error and perpetual equivalent metrics. A similar trend was observed on sustained vowels and yawnings; as well as under additive noise conditions. Nonetheless, the lower performance levels for the collected audio samples compared to the synthetic inputs, in absolute error and according to the perceptual-equivalent metrics, may be influenced by the limited ability of the Pink Trombone to match sounds out of its standard tract setting.

We comprehensively evaluated some of the most commonly used optimization algorithms in a black-box approach, predicting their control parameters to synthesize non-speech sounds. We tested different audio representations and conducted experiments in different scenarios, ranging from simple single-parameter predictions to complex, time-varying parameters or non-speech human-made sounds. Our results show that the Evolution Strategies (GA and CMA-ES) and Particle Swarm Optimization with multiresolution representation are the most effective for predicting control parameters with minimum error (MAE $< 1\%$) and high quality (ViSQOL 4.3 for PT, PEAQ 3.0 for yawns). Also, PSO achieved the best performance vs. computational cost ratio.

According to our results, GA and PSO algorithms were superior to the other optimization methods in most cases. The NM algorithm struggled with local minima, and the TRF algorithm, although fast, could not optimize the parameters satisfactorily. The NN could predict the control parameters when the input was a sound generated by the PT. However, it failed when confronted with real sounds or noisy inputs. NN strategy can be useful for certain scenarios since it is also very fast once trained, but it can hardly reach the generalization of the GA.

Regarding audio representations, our experiments demonstrate that all those that we tested, including MFCC, STFT, MEL, and multiresolution decomposition, are suitable for optimizing individual control parameters. However, the MFCC representation showed poorer pitch prediction capability than other representations. This is consistent with what is expected from a cepstral representation according to the literature.

Perceptual metrics validate that the optimizers are able to faithfully predict audio samples generated by the synthesizer itself. Taking this benchmark, we can observe that real sounds do not reach such a high performance. The conclusion is that our synthesizer cannot achieve certain characteristics of real voices in the given conditions (e.g. vocal tract size). Nevertheless, comparable sounds have been achieved, which is the goal of our research.

Future research should explore new techniques that enhance the prediction of time-varying signals. Further analysis is needed to investigate the parameter's flow, whether it aligns with the typical structures of a human vocal tract or the chosen optimization strategy. Additionally, hierarchical optimizations can improve the neural network's performance. Predictions should be conducted in two stages by initially narrowing the bounds and then fine-tuning.

Finally, future work should use this framework to benchmark other solutions to the problem, including alternative optimization methods, acoustic features, metrics, and subjective tests.

6. ACKNOWLEDGMENTS

The authors would like to thank David Südholt for the valuable discussions, support and help. Activities described in this contribution were partially funded by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 101003750, the Ministry of Economy and Competitiveness of Spain under grant PID2021-128469OB-I00, and the UPM Research Programme, *Programa Propio de I+D+I 2022*.

7. REFERENCES

[1] Christopher T Kello and David C Plaut, "A neural network model of the articulatory-acoustic forward mapping trained on recordings of articulatory parameters," *The Journal of the Acoustical Society of America*, vol. 116, no. 4, pp. 2354–2364, 2004.

[2] Sandesh Aryal and Ricardo Gutierrez-Osuna, "Data driven articulatory synthesis with deep neural networks," *Computer Speech & Language*, vol. 36, pp. 260–273, 2016.

[3] M Mehdi Afsar et al., "Generating diverse realistic laughter for interactive art," *arXiv preprint arXiv:2111.03146*, 2021.

[4] Chin-Cheng Hsu, "Synthesizing personalized non-speech vocalization from discrete speech representations," *arXiv preprint arXiv:2206.12662*, 2022.

[5] Andrey Anikin, "Soundgen: an open-source tool for synthesizing nonverbal vocalizations," *Behavior research methods*, vol. 51, no. 2, pp. 778–792, 2019.

[6] Aaron van den Oord et al., "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[7] Brad H Story, "A parametric model of the vocal tract area function for vowel and consonant simulation," *The Journal of the Acoustical Society of America*, vol. 117, no. 5, pp. 3231–3254, 2005.

[8] "Freesound technical demo," in *ACM International Conference on Multimedia (MM'13)*, Barcelona, Spain, oct. 2013, ACM, pp. 411–412.

[9] Thomas J Mitchell and David P Creasey, "Evolutionary sound matching: A test methodology and comparative study," in *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*. IEEE, 2007, pp. 229–234.

[10] Yuyo Lai et al., "Automated optimization of parameters for fm sound synthesis with genetic algorithms," in *International Workshop on Computer Music and Audio Technology*. Citeseer, 2006, p. 205.

[11] Jorge Zúñiga and Joshua D Reiss, "Realistic procedural sound synthesis of bird song using particle swarm optimization," in *Audio Engineering Society Convention 147*, 2019.

[12] Matthew John Yee-King et al., "Automatic programming of vst sound synthesizers using deep networks and other techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, 2018.

[13] G Pyz et al., "Lithuanian speech synthesis by computer using additive synthesis," *Elektronika ir Elektrotechnika*, vol. 18, no. 8, pp. 77–80, 2012.

[14] MT Vakil Baghmisheh et al., "Frequency modulation sound parameter identification using novel hybrid evolutionary algorithms," in *2008 International Symposium on Telecommunications*. IEEE, 2008, pp. 67–72.

[15] Oren Barkan and David Tsiris, "Deep synthesizer parameter estimation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2019, pp. 3887–3891.

[16] Jesse Engel et al., "Ddsp: Differentiable digital signal processing," *arXiv preprint arXiv:2001.04643*, 2020.

[17] Naotake Masuda and Daisuke Saito, "Synthesizer sound matching with differentiable dsp.," in *ISMIR*, 2021, pp. 428–434.

[18] David Moffat and Joshua D Reiss, "Perceptual evaluation of synthesized sound effects," *ACM Transactions on Applied Perception (TAP)*, vol. 15, no. 2, pp. 1–19, 2018.

[19] Daniel M Munoz et al., "Opposition-based shuffled pso with passive congregation applied to fm matching synthesis," in *IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 2775–2781.

[20] Janne Riionheimo and Vesa Välimäki, "Parameter estimation of a plucked string synthesis model using a genetic algorithm with perceptual fitness calculation," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, pp. 1–15, 2003.

[21] Katsutoshi Itoyama and Hiroshi G Okuno, "Parameter estimation of virtual musical instrument synthesizers," in *ICMC*, 2014.

[22] John Kelly, "Speech synthesis," *Proc. 4th Int. Congr. Acoustics, 1962*, pp. 1–4, 1962.

[23] Christian J. Steinmetz and Joshua D. Reiss, "auraloss: Audio focused loss functions in PyTorch," in *Digital Music Research Network One-day Workshop (DMRN+15)*, 2020.

[24] Antony W Rix et al., "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, 2001, vol. 2, pp. 749–752.

[25] Thilo Thiede et al., "Peaq - the itu standard for objective measurement of perceived audio quality," *Journal of the Audio Engineering Society*, vol. 48, no. 1/2, pp. 3–29, 2000.

[26] Michael Chinen et al., "Visqol v3: An open source production ready objective speech and audio metric," in *2020 twelfth international conference on quality of multimedia experience (QoMEX)*. IEEE, 2020, pp. 1–6.

[27] Norman R French and John C Steinberg, "Factors governing the intelligibility of speech sounds," *The journal of the Acoustical society of America*, vol. 19, no. 1, pp. 90–119, 1947.

[28] John H Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[29] James Kennedy and Russell Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*. IEEE, 1995, vol. 4, pp. 1942–1948.

[30] Mary Ann Branch, Thomas F Coleman, and Yuying Li, "A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems," *SIAM Journal on Scientific Computing*, vol. 21, no. 1, pp. 1–23, 1999.

[31] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 01 1965.

[32] Nikolaus Hansen and Andreas Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," 06 1996, pp. 312 – 317.

[33] Abraham. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

NONLINEAR STRINGS BASED ON MASSES AND SPRINGS

Silvin Willemsen

SW Audio
The Netherlands
silvinwillemsen@gmail.com

ABSTRACT

Due to advances in computational power, physical modelling for sound synthesis has gained an increased popularity over the past decades. Although much work has been done to accurately simulate existing physical systems, much less work exists on the use of physical modelling simply for the sake of creating sonically interesting sounds. This work presents a mass-spring network, inspired by existing models of the physical string. Masses have 2 translational degrees of freedom (DoF), and the springs have an additional equilibrium separation term, which together result in highly nonlinear effects. The main aim of this work is to create sonically interesting sounds while retaining some of the natural qualities of the physical string, as opposed to accurately simulating it. Although the implementation exhibits chaotic behaviour for certain choices of parameters, the presented system can create sonically interesting timbres, including nonlinear pitch glides and 'wobbles'.

1. INTRODUCTION

Mass-spring networks for sound synthesis have been investigated for over 40 years. Originally introduced in a musical context by Cadoz *et al.* [1, 2, 3], mass-interaction models have seen recent developments by Leonard and Villeneuve in [4, 5]. The modularity of mass-spring networks and their simple formulation make it an attractive physical modelling technique for creating interesting sounds relatively quickly.

As one is restricted to a finite number of nodes in space (i.e., the masses), other physical modelling techniques have been often used to model the musical string. Over the past 50 years, the string has been modelled using various methods, including physically-inspired methods such as the Karplus-Strong algorithm [6] and digital waveguides [7], as well as modal synthesis [8], and finite-difference time-domain (FDTD) methods [9, 10]. For an ideal string, the latter methods have an equivalent mass-spring formulation as described in [11, Sec 6.1.1], but once one wants to add stiffness, FDTD methods are a much more straightforward alternative.

The above models assume low-amplitude string vibration such that the string can be approximated using a linear model. High-amplitude string vibration results in an initial higher pitch of the string after which it decreases due to an effect called tension modulation [12]. To include this effect in the string model, it needs to be extended to be nonlinear instead. One of the most recent works on nonlinear string modelling is due to Ducceschi & Bilbao in [13], who use a mixed FDTD/modal scheme and energy quadratisation techniques, to model the geometrically exact string.

One could imagine that a proper implementation of the nonlinear string requires a very involved mathematical formulation, and quickly loses simplicity in implementation. As opposed to FDTD methods, a mass-spring formulation treats the discrete nodes of the implementation as separate connected entities, rather than as part of a predefined system, which provides additional flexibility in several aspects. One of these is the relatively easy extension to additional degrees of freedom (DoF) per node. In most traditional FDTD schemes, each node only has one degree of freedom, which in the case of the string is the transverse displacement [11]. If we allow for more degrees of freedom, this could potentially lead to interesting nonlinear effects. Furthermore, every mass and spring can be treated as a separate entity of which parameters can be set independently of each other also potentially leading to interesting sonic qualities.

The aim of this work is not to accurately simulate the nonlinear stiff string, but instead to create a flexible model that can produce string-like sounds with interesting nonlinear sonic qualities. To this end, this work uses a mass-spring formulation due to its flexibility and relatively simple formulation. The main additions with respect to an FDTD implementation of the (damped) ideal string are using 2 DoF for each mass and an equilibrium separation for the springs connecting neighbouring masses. These two additions together, result in pitch glides and interesting timbres not feasible with linear models.

The rest of this paper is structured as follows: Section 2 describes the continuous-time model starting with the description of the 2-DoF mass and extending this to a sequentially connected network resembling a string. Section 3 discretises the model and uses analogies to the FDTD formulation to determine the stability and fundamental frequency of the system. Section 4 provides results and discusses several experiments done using the presented schemes, and Section 5 concludes.

2. MODEL

2.1. Single mass

Before considering a network of masses and springs, first consider a single mass $\mathbf{u} = \mathbf{u}(t)$ with time t (in s), defined over two spatial DoF,

$$\mathbf{u} = \begin{bmatrix} u_x & u_y \end{bmatrix}. \tag{1}$$

Here, $u_x = u_x(t)$ and $u_y(t)$ describe the x and y-location of the mass, respectively (in m). Using the ∂_t operator to denote differentiation with respect to time, the ODE describing the dynamics of a 2-DoF mass-spring system connected to the origin is (such as

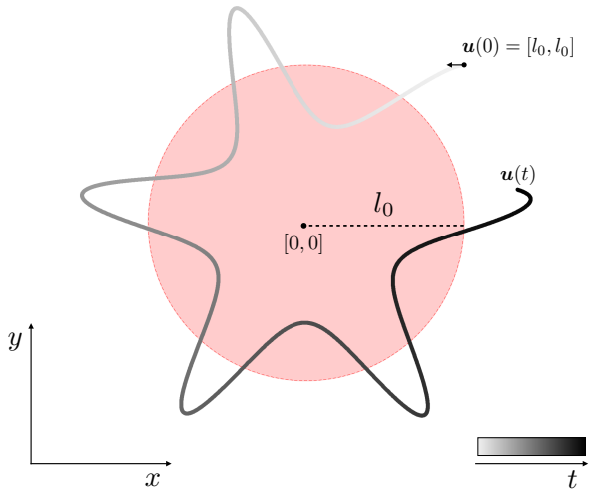


Figure 1: *The effect of l_0 on the trajectory of a 2-DoF mass-spring system. The mass will be pulled towards the origin ($[0, 0]$) if it is outside of the red region. If the mass is in the red region instead, it will be pushed away from the origin. Here, the mass is initialised at $[l_0, l_0]$ and given a small horizontal velocity in the negative x-direction.*

done in [4])

$$M\partial_t^2 \mathbf{u} = -Kf\frac{\mathbf{u}}{\|\mathbf{u}\|}, \quad \text{with} \quad f = \|\mathbf{u}\| - l_0, \tag{2}$$

mass $M > 0$ (in kg), spring constant $K > 0$ (in N/m), equilibrium spring separation $l_0 \geq 0$ (in m) and spring force f (in N). Furthermore, $\|\mathbf{u}\|$ describes the magnitude of $\mathbf{u}$ (in m), which when using two DoF is defined as

$$\|\mathbf{u}\| = \sqrt{u_x^2 + u_y^2}\,. \tag{3}$$

The equilibrium separation l_0, can be seen as introducing a zone where the spring pushes the mass away from the equilibrium rather than pulling it towards it. See Figure 1.

One can observe that if $l_0 = 0$, Eq. (2) reduces to a mass-spring system whose dimensions are uncoupled; in other words, l_0 introduces coupling between the two dimensions.

2.2. Multiple masses

Going towards a string-like mass-spring network, one can create a system of N_{mass} masses connected by N_{spring} springs, which are related as follows:

$$N_{\text{mass}} = N_{\text{spring}} + 1. \tag{4}$$

Subscript $m = \{0, \ldots N_{\text{spring}}\}$ will be used to index the masses, i.e., mass m will be described by the state $\mathbf{u}_m = \mathbf{u}_m(t)$. For a system of length L (in m), the masses are initially placed along the x-axis with no displacement in the y-direction according to

$$\mathbf{u}_m(0) = \begin{bmatrix} m\Delta_0 & 0 \end{bmatrix}, \tag{5}$$

where $\Delta_0 = L/N_{\text{spring}}$ is the initial distance between two consecutive masses. Connecting the masses such that the spring forces between masses m and $m+1$ positively affects mass m and the force between masses m and $m-1$ negatively affects mass m yields

$$\begin{aligned} M\partial_t^2 \mathbf{u}_m = {} & Kf_{m+1/2}\frac{\mathbf{u}_{m+1} - \mathbf{u}_m}{\|\mathbf{u}_{m+1} - \mathbf{u}_m\|} \\ & - Kf_{m-1/2}\frac{\mathbf{u}_m - \mathbf{u}_{m-1}}{\|\mathbf{u}_m - \mathbf{u}_{m-1}\|}, \end{aligned} \tag{6}$$

where the spring force between masses m and $m+1$ is

$$f_{m+1/2} = \|\mathbf{u}_{m+1} - \mathbf{u}_m\| - l_0. \tag{7}$$

Notice that M applies to all masses and K and l_0 apply to all springs in the network.

2.3. Boundaries

Although in a mass-spring context one does not usually speak of boundary conditions, conditions for the first and the last mass still need to be defined. These are set to the following states:

$$\mathbf{u}_0(t) = \begin{bmatrix} 0 & 0 \end{bmatrix}, \quad \text{and} \quad \mathbf{u}_{N_{\text{spring}}}(t) = \begin{bmatrix} L & 0 \end{bmatrix}, \quad \forall t. \tag{8}$$

2.4. Damping

Similar to the damped stiff string (see e.g. [11]), one can add damping terms to Eq. (6) according to

$$\begin{aligned} M\partial_t^2 \mathbf{u}_m = {} & \ldots - 2\sigma M\partial_t \mathbf{u}_m \\ & + 2zM\partial_t\left(\mathbf{u}_{m+1} - \mathbf{u}_m\right) - 2zM\partial_t\left(\mathbf{u}_m - \mathbf{u}_{m-1}\right), \end{aligned} \tag{9}$$

with mass damping $\sigma \geq 0$ and spring damping $z \geq 0$ (both in s^{-1}). The spring damping term acts as a frequency-dependent damping term, analogous to the damped stiff string. As the implementation of this damping term is also analogous to its implementation in the stiff string it can be shown to be passive under similar conditions (see Sec. 3.1.2).

Using the following shorthand notation

$$\boldsymbol{\mathcal{U}}_{m+1/2} \triangleq \frac{\mathbf{u}_{m+1} - \mathbf{u}_m}{\|\mathbf{u}_{m+1} - \mathbf{u}_m\|}, \tag{10}$$

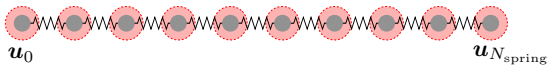
one can simplify Eq. (9) to

$$\begin{aligned} \partial_t^2 \mathbf{u}_m = {} & \frac{K}{M}\left(\mathbf{u}_{m+1} - 2\mathbf{u}_m + \mathbf{u}_{m-1}\right) \\ & - \frac{Kl_0}{M}(\boldsymbol{\mathcal{U}}_{m+1/2} - \boldsymbol{\mathcal{U}}_{m-1/2}) \\ & - 2\sigma\partial_t \mathbf{u}_m + 2z\partial_t\left(\mathbf{u}_{m+1} - 2\mathbf{u}_m + \mathbf{u}_{m-1}\right). \end{aligned} \tag{11}$$

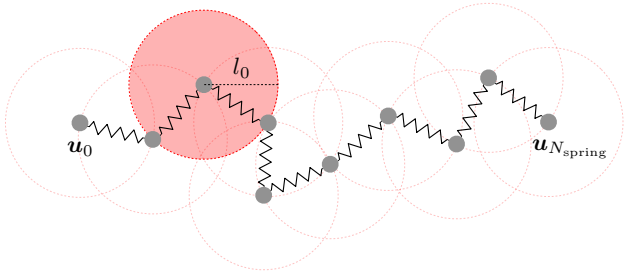
Figure 2 illustrates possible stable states (initially excited and after damping) of the 2-DoF mass-spring network described in this section for low and high values of l_0.

3. DISCRETE TIME

One can discretise continuous time according to $t = nk$, with time index $n = 0, 1, \ldots$, time step $k = 1/f_s$ (in s) and sample rate f_s (in Hz). The position of mass m can then be discretised at



(a) *Small value of* l_0.



(b) *Large value of* l_0.

Figure 2: *Possible stable states for different values of* l_0.

time t according to $\mathbf{u}_m(t) \approx \mathbf{u}_m^n$. To approximate the continuous-time derivatives used in the previous section, the following finite-difference (FD) operators need to be introduced:

$$\partial_t \mathbf{u}_m \approx \begin{cases} \delta_{t+}\mathbf{u}_m^n \triangleq \frac{1}{k}\left(\mathbf{u}_m^{n+1} - \mathbf{u}_m^n\right), & \text{(12a)} \\ \delta_{t-}\mathbf{u}_m^n \triangleq \frac{1}{k}\left(\mathbf{u}_m^n - \mathbf{u}_m^{n-1}\right), & \text{(12b)} \\ \delta_{t\cdot}\mathbf{u}_m^n \triangleq \frac{1}{2k}\left(\mathbf{u}_m^{n+1} - \mathbf{u}_m^{n-1}\right), & \text{(12c)} \end{cases}$$

which are the forward, backward, and centred difference respectively. The former two are first-order accurate, whereas the latter is second-order accurate [11]. A second-order differentiation can be approximated using

$$\partial_t^2 \mathbf{u}_m \approx \delta_{tt}\mathbf{u}_m^n \triangleq \frac{1}{k^2}\left(\mathbf{u}_m^{n+1} - 2\mathbf{u}_m^n + \mathbf{u}_m^{n-1}\right), \tag{13}$$

which is also second-order accurate.

Using these definitions, Eq. (11) can then be discretised to the following scheme:

$$\begin{aligned} \delta_{tt}\mathbf{u}_m^n = {} & \frac{K}{M}\left(\mathbf{u}_{m+1}^n - 2\mathbf{u}_m^n + \mathbf{u}_{m-1}^n\right) \\ & - \frac{Kl_0}{M}\left(\boldsymbol{\mathcal{U}}_{m+1/2}^n - \boldsymbol{\mathcal{U}}_{m-1/2}^n\right) \\ & - 2\sigma\delta_{t\cdot}\mathbf{u}_m^n + 2z\delta_{t-}\left(\mathbf{u}_{m+1}^n - 2\mathbf{u}_m^n + \mathbf{u}_{m-1}^n\right), \end{aligned} \tag{14}$$

where

$$\boldsymbol{\mathcal{U}}_{m+1/2}^n \triangleq \frac{\mathbf{u}_{m+1}^n - \mathbf{u}_m^n}{\|\mathbf{u}_{m+1}^n - \mathbf{u}_m^n\|}, \tag{15}$$

is Eq. (10) discretised. Notice that the first-order FD operators are chosen to yield the highest accuracy, while keeping the scheme explicit. Appendix 7 provides an alternative, implicit discretisation.

To implement scheme (14), it needs to be expanded to an update equation, or recursion:

$$\begin{aligned} (1+\sigma k)\mathbf{u}_m^{n+1} = {} & \left(2 - \frac{2Kk^2}{M} - 4zk\right)\mathbf{u}_m^n \\ & + \left(\frac{Kk^2}{M} + 2zk\right)\left(\mathbf{u}_{m+1}^n + \mathbf{u}_{m-1}^n\right) \\ & - \frac{Kl_0k^2}{M}\left(\boldsymbol{\mathcal{U}}_{m+1/2}^n - \boldsymbol{\mathcal{U}}_{m-1/2}^n\right) \\ & + (\sigma k + 4zk - 1)\mathbf{u}_m^{n-1} - 2zk\left(\mathbf{u}_{m+1}^{n-1} + \mathbf{u}_{m-1}^{n-1}\right), \end{aligned} \tag{16}$$

which, after division by $(1+\sigma k)$, can be solved for $\mathbf{u}_m^{n+1}$.

3.1. Analogies to FDTD schemes

If one is familiar with FDTD methods, it is easy to see the resemblance between scheme (14) and FDTD schemes of the (damped) 1D wave equation. Despite some differences (being the term including l_0 and the possibility for additional DoF), this resemblance can still be used to find definitions for the fundamental frequency and stability, as will be presented in this section. For completeness, the FDTD scheme of the 1D wave equation is given here.

The transverse displacement of an ideal string of length L (in m) can be described by state variable $u = u(x,t)$ (in m), which is defined over space $x \in [0, L]$ (in m) and time t (in s). Space x is subdivided into N_{FD} equally sized intervals according to $x = lh$ with spatial index $l = \{0, \ldots, N_{\text{FD}}\}$, and grid spacing h (in m). Time t is discretised according to the same definitions presented at the beginning of this section. Using these definitions, $u(x,t)$ can be approximated by grid function u_l^n.

Introducing the following FD operator, which approximates a second-order spatial derivative

$$\partial_x^2 u \approx \delta_{xx}u_l^n \triangleq \frac{1}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right), \tag{17}$$

the discrete damped 1D wave equation can be described by the following scheme [11]:

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n. \tag{18}$$

Here, c is the wave speed (in m/s), and σ_0 and σ_1 are the frequency-independent (in s^{-1}) and the frequency-dependent damping coefficients (in m^2/s), respectively.

3.1.1. Fundamental frequency

To obtain the fundamental frequency (in Hz) of a string of length L (in m) fixed at the ends, one uses

$$f_0 = \frac{c}{2L}. \tag{19}$$

Ignoring the damping terms for now (as these do not influence f_0), one can compare the update equation of the 1D wave equation with that of the mass spring network in Eq. (16):

$$\begin{aligned} u_l^{n+1} &= \left(2 - \frac{2c^2k^2}{h^2}\right)u_l^n - u_l^{n-1} + \frac{c^2k^2}{h^2}\left(u_{l+1}^n + u_{l-1}^n\right), \\ \mathbf{u}_m^{n+1} &= \left(2 - \frac{2Kk^2}{M}\right)\mathbf{u}_m - \mathbf{u}_l^{n-1} + \frac{Kk^2}{M}\left(\mathbf{u}_{m+1}^n + \mathbf{u}_{m-1}^n\right). \end{aligned}$$

One can observe that the following combination of variables is analogous to each other:

$$\frac{c^2}{h^2} \iff \frac{K}{M}. \tag{21}$$

As the mass-spring network does not make use of h we can use $h = L/N_{\text{FD}}$ (in the case of the Courant number $\lambda = ck/h = 1$ for the 1D wave equation), and substitute this to yield

$$\frac{c^2 N_{\text{FD}}^2}{L^2} = \frac{K}{M}. \tag{22}$$

As N_{FD} describes the number of intervals (rather than the number of grid points) in the FDTD scheme, this is analogous to the number of springs in the mass-spring network N_{spring}. Using its relation to the number of masses N_{mass} in Eq. (4), rearranging Eq. (22) in terms of c, and substituting this into Eq. (19) yields

$$f_0 = \frac{\sqrt{K/M}}{N_{\text{mass}} - 1}. \tag{23}$$

It is interesting to note that the fundamental frequency is solely determined by the spring constant, the mass, and the number of masses in the system. Therefore, the length L no longer has an influence on the fundamental frequency of the system.

3.1.2. Stability

The stability condition for Eq. (18) can be shown to be [14]

$$\frac{c^2 k^2}{h^2} + \frac{4\sigma_1 k}{h^2} \leq 1. \tag{24}$$

Comparing Eqs. (14) and (18) again, an analogy between the following variables can be made

$$\sigma_1/h^2 \iff z,$$

which, after including Eq. (21), one can rewrite (24) the following stability condition for the mass-spring network

$$\frac{Kk^2}{M} + 4zk \leq 1.$$

As is the case for the 1D wave equation, the closer this condition is to being satisfied with equality, the higher the simulation bandwidth. As we would like to have control over this condition later on, this is rewritten to

$$\frac{Kk^2}{M} + 4zk \leq \Lambda, \tag{25}$$

where $0 < \Lambda \leq 1$ determines the bandwidth limit of the simulation.

3.2. Implementation

Assuming that the fundamental frequency is known, Eq. (23) can be rewritten to

$$N_{\text{mass}} = \frac{\sqrt{K/M}}{2f_0} + 1. \tag{26}$$

Rewriting Eq. (25) in terms of K/M,

$$\frac{K}{M} \leq \frac{(\Lambda - 4zk)}{k^2},$$

and substituting this into Eq. (26), yields

$$N_{\text{mass}} \geq \frac{\sqrt{(\Lambda - 4zk)}}{2f_0 k} + 1. \tag{27}$$

As N_{mass} is an integer, a rounding operation needs to be performed on Eq. (27) that also satisfies the condition. The following can therefore be used to calculate the number of masses

$$N_{\text{mass}} = \left\lfloor \frac{\sqrt{(\Lambda - 4zk)}}{2f_0 k} \right\rfloor + 1, \tag{28}$$

where $\lfloor \cdot \rfloor$ is the flooring operation. Finally, either M or K can be fixed to an arbitrary value, and the other can be calculated by rewriting Eq. (26)

$$K = M(2f_0(N_{\text{mass}} - 1))^2. \tag{29}$$

Here, M is kept fixed and K is changed, analogous to changing the tension of a string and keeping the mass per unit length fixed.

3.3. Output

One can obtain the output of the system by selecting one mass and following its state over time. For an interesting stereo effect, the longitudinal (x) and transverse (y) dimensions, can be mapped to the left and right channel, respectively. As $u^n_{x,m}$ has an initial non-zero location due to Eq. (5) this needs to be corrected for, resulting in

$$o_{\text{left}}(n) = u^n_{x,m_{\text{o}}} - m_{\text{o}}\Delta_0 \quad \text{and} \quad o_{\text{right}}(n) = u^n_{y,m_{\text{o}}}, \tag{30}$$

where $m_{\text{o}} \in \{0, N_{\text{spring}}\}$ is the index of the mass selected for the output.

3.4. Extension to anisotropic systems

Another advantage of using a mass-spring formulation as opposed to a FDTD one, is that it is relatively straightforward to use different parameter values for different parts of the network. Although this subsection will not be further discussed in the next sections, it is interesting to mention a simple extension of Eq. (14) to be anisotropic.

One can rewrite Eq. (9) to allow for different values for M and K along the network.[1] Using M_m to denote the mass of mass m, and $K_{m+1/2}$ to describe the spring force between masses m and $m+1$,

$$\begin{aligned} M_m \partial_t^2 \mathbf{u}_m = {} & K_{m+1/2} f_{m+1/2} \boldsymbol{\mathcal{U}}_{m+1/2} \\ & - K_{m-1/2} f_{m-1/2} \boldsymbol{\mathcal{U}}_{m-1/2} \\ & - 2\sigma M_m \partial_t \mathbf{u}_m \\ & + 2z M_{m+1/2} \partial_t \left(\mathbf{u}_{m+1} - \mathbf{u}_m\right) \\ & - 2z M_{m-1/2} \partial_t \left(\mathbf{u}_m - \mathbf{u}_{m-1}\right). \end{aligned} \tag{31}$$

Here,

$$M_{m+1/2} = \frac{1}{2}\left(M_{m+1} + M_m\right) \tag{32}$$

is the average mass of two neighbouring masses. Please note that the fundamental frequency calculation in Eq. (23) does not hold for an anisotropic system.

[1] In principle, σ, z, and l_0 could also have been chosen to vary along the network, but are left constant for the sake of brevity.

3.4.1. Stability

In order to keep the system stable, the stability condition in Eq. (25) needs to be adapted to challenge the condition most:

$$\frac{K_{\max}k^2}{M_{\min}} + 4zk \leq \Lambda, \tag{33}$$

where

$$K_{\max} = \max_{m\in\{0,\ldots,N_{\text{spring}}\}} K_{m+1/2} \quad \text{and} \quad M_{\min} = \min_{m\in\{0,\ldots,N_{\text{mass}}\}} M_m. \tag{34}$$

4. RESULTS AND DISCUSSION

This section presents the results of several simulations using Eq. (16) with different parameter values and discusses these. Sound examples can be found online [15].

4.1. Simulation setup

As according to Eq. (23) the length of the system L does not change the eventual behaviour of the system, we can set $L = N_{\text{spring}}$ such that $\Delta_0 = 1$ and Eq. (5) simplifies to

$$\mathbf{u}_m(0) = \begin{bmatrix} m & 0 \end{bmatrix}.$$

This way, the value of l_0 can be seen as a ratio of the initial difference between two consecutive masses (i.e., $l_0 = 0.5$ yields a resting length of half the initial distance between two masses). The system is then excited by giving mass $m_{\text{e}} \in \{0, \ldots, N_{\text{spring}}\}$ an initial displacement of e in both x and y directions:

$$\mathbf{u}^0_{m_{\text{e}}} = \mathbf{u}^1_{m_{\text{e}}} = \begin{bmatrix} m_{\text{e}} + e & e \end{bmatrix}. \tag{35}$$

Notice that with this setup, the output in Eq. (30) has to be normalised (divided by e to yield output in the $[-1, 1]$ range. Table 1 shows the other parameters used for the experiments and provides usable ranges for some.

4.2. Chaotic behaviour

The first thing to note, is that for values of Λ close to 1 in Eq. (25), non-zero values of l_0 cause increasingly chaotic behaviour, which causes the system to produce 'buzzing' output. This is in line with what Bilbao mentions in [11, p. 229], stating that

> "*... for a nonlinear system, anomalous behavior may be observed when the grid spacing is chosen close to the stability bound.*"

The behaviour is most likely caused by a numerical integration error of the nonlinear term, which is (most probably) why the implicit implementation in Appendix 7 shows slightly improved behaviour in this regard. It is important to note that the chaotic behaviour does not imply that the implementation is unstable; although the system might never fully decay, it does not exhibit explosive behaviour!

The system has been tested for different values of l_0, z and Λ to see whether it would either exhibit chaotic behaviour or decay instead. These tests have been repeated at $f_{\text{s}} = 44100$ and $f_{\text{s}} = 88200$. A full overview of the results can be found in Figure 3. All generated sounds can be found via [15] and were obtained through Eq. (30).

Table 1: *Parameter values divided into static parameters used to generate the results and parameters that can be tweaked to generate different behaviour.*

Parameter	Symbol (unit)	Value
Static parameters		
Mass	M (kg)	0.01
Fundamental freq.	f_0 (Hz)	100
Spring stiffness	K (N/m)	Eq. (29)
Mass damping	σ (s^{-1})	1
Initial displacement	e (m)	100
Initial inter-mass dist.	Δ_0 (m)	1
Excited mass	m_{e} (-)	$m_{\text{e}} = \lfloor 0.63 N_{\text{spring}} \rfloor$
Output mass	m_{o} (-)	10
Number of masses	N_{mass} (-)	Eq. (28)
Altered parameters		
Stability bound	Λ (-)	$0 < \Lambda \leq 1$
Spring damping	z (s^{-1})	$z \in [0, 5^*]$
Equilibrium sep.	l_0 (m)	$l_0 \in [0, 2^*]$
Sample rate	f_{s} (Hz)	$f_{\text{s}} \in [44100, 88200]^*$

*these numbers are to determine usable ranges, but the parameters are not bounded by these values.

The results indicate that lower values for Λ and l_0 are the main factors for preventing chaotic behaviour. Similar results are obtained for both sample rates (even slightly in favour of $f_{\text{s}} = 44100$). This is because the simulation is not actually oversampled; more masses are added according to Eq. (28) due to a decrease in k. If one instead oversamples without changing any other parameters, this automatically decreases Λ in Eq. (25) resulting in reduced chaotic behaviour. Although this retains a high simulation bandwidth, it does increase the computational cost.

If the eventual goal is to implement this algorithm in real time, a better option would be to reduce Λ manually. Although this decreases the simulation bandwidth, it decreases chaotic behaviour without increasing the computational complexity (even reducing it by reducing the number of masses in springs through Eq. (28)!). Results show that if one chooses $z \geq 3$, values for $\lambda \leq 0.1$ result in non-chaotic behaviour for $l_0 \in [0, 2]$ (and probably higher).

The fact that an increase in the spring damping z decreases chaotic behaviour follows from the fact that the chaotic oscillations cause rapid extensions and contractions of the springs. This will be more damped for higher values of z. Furthermore, a higher value for z also increases the speed that the simulation reaches a stable equilibrium.

4.3. Frequency-domain behaviour

Spectrograms of the simulation with parameters $l_0 = 0.75$, $z = 2$, $\Lambda = 0.1$ can be seen in Figure 4. The results and discussion below assume that Λ and z are chosen such that the system does not behave chaotically.

All non-zero values of l_0 have a tension-reduction effect on the string in the transverse (y) direction. In the longitudinal (x) direction, however, the effect of l_0 on the frequency depends on its value with respect to the initial distance between the masses Δ_0. For $l_0 \leq \Delta_0$, the equilibrium separation in the springs eventually (after damping) cancel each other out, and the system will – after a slight 'wobble' downwards in pitch – return to its original funda-

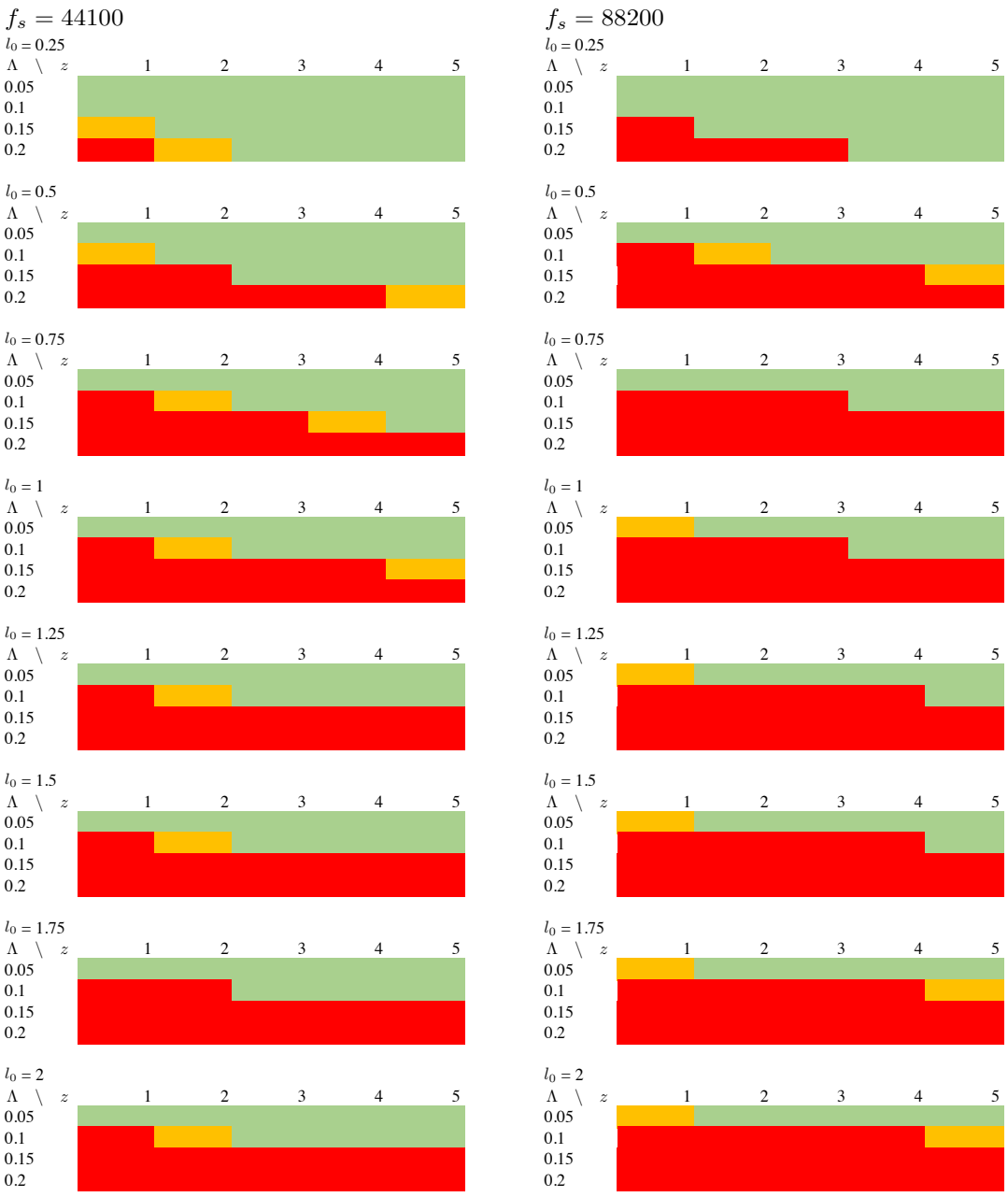


Figure 3: *Simulation results for* $\Lambda = \{0.05, 0.1, 0.15, 0.2\}$, $z = \{1, \ldots 5\}$ *and* $l_0 = \{0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2\}$. *Simulations lasted for 5 seconds and* ***green cells*** *indicate that the output decays within this time.* ***Red cells*** *indicate that the output does not decay within this time, and the system is therefore considered to exhibit chaotic behaviour.* ***Orange cells*** *are boundary cases, where the decay is slightly longer than usual, but the output decays within 5 s.*

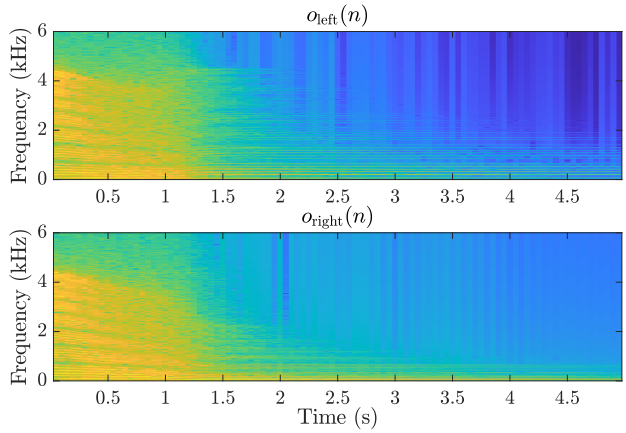


Figure 4: *Spectrograms of the left (longitudinal) and right (transverse) output in Eq.* (30) *with parameters from Table 1 and* $l_0 = 0.75$*,* $z = 2$*,* $\Lambda = 0.1$*, and* $f_s = 44100$*.*

mental frequency. For l_0 values larger than the initial equilibrium separation ($l_0 > \Delta_0$), the string will reach a stable 'loose' state such as depicted in Figure 2b. These results are summarised in Table 2.

Preliminary tests show that $l_0 \approx 0.75\Delta_0$ causes a pitch glide to $f_0/2$ (the subharmonic) and $l_0 \approx 0.94\Delta_0$ to $f_0/4$. More work needs to be done to find the exact relationship between l_0 and f_0. As opposed to the nonlinear schemes presented in e.g. [11, Ch. 8], the pitch glides do not start higher than the original fundamental frequency, but instead move towards 0 Hz, due to the tension-reduction effect of l_0.

Table 2: *Effect of equilibrium separation* l_0 *on output (Eq.* (30)*) if behaviour is not chaotic.*

Value for l_0	Effect on $o_{\text{left}}(n)$	Effect on $o_{\text{right}}(n)$
$l_0 \leq \Delta_0$	Glide down and back up.	Glide to lower f_0
$l_0 > \Delta_0$	Glide to $f_0 = 0$.	Glide to $f_0 = 0$

4.4. Note on only using the longitudinal direction

If one chooses to only excite the longitudinal direction, i.e.

$$\mathbf{u}^0_{m_\text{e}} = \mathbf{u}^1_{m_\text{e}} = \begin{bmatrix} m_\text{e} + e & 0 \end{bmatrix},$$

the following holds:

$$u^n_{y,m} = 0, \quad \forall m, \forall n.$$

In other words, the system behaves as if there was no transverse dimension. For $l_0 \leq \Delta_0$ pitch gliding effects still occur, but chaotic behaviour already occurs for much lower values of l_0. For $l_0 > \Delta_0$, due to the lack of the transversal dimension, the masses effectively have "nowhere to go", resulting in chaotic behaviour at all times.

4.5. Note on computational complexity

Compared to an implementation of the damped 1D wave equation, the update equation in Eq. (16) introduces additional computations in two different ways. First is the obvious extension to 2 DoF, doubling the number of computations with respect to a 1-DoF implementation. The second and more important contribution to computational complexity comes from the calculation of $\boldsymbol{\mathcal{U}}$; more specifically the calculation of the Euclidian distance between two neighbouring masses ($\|\cdot\|$), which adds N_{spring} square-root operations every time step.

However, due to the already computationally inexpensive implementation of the damped 1D wave equation, these additional aspects should definitely not prevent a real-time implementation of the model presented here.

5. CONCLUSION

This paper presents a mass-spring network configured like a string. The masses in the network can move in 2 DoF and the springs connecting the masses have an equilibium separation. These properties cause nonlinear behaviour in the system, such as wobbles and pitch glides.

Although parameters could be chosen that yield chaotic behaviour, results show that one can prevent this by choosing parameters away from the stability bound and including spring damping. Future work includes to find a more precise definition for parameter ranges for which the implementation does not exhibit chaotic behaviour, as well as a relationship for the fundamental frequency and equilibrium spring separation. Finally, it would be interesting to see how this model compares to other already existing models of nonlinear strings or modular mass-spring networks, such as the CORDIS-ANIMA software [3].

6. REFERENCES

[1] C. Cadoz, *Synthèse sonore par simulation de mécanismes vibratoires*, Ph.D. thesis, Thèse de Docteur Ingénieur, I.N.P.G. Grenoble, France, 1979.

[2] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms: the CORDIS system," *Computer Music Journal*, vol. 8, no. 3, pp. 60–73, 1983.

[3] C. Cadoz, A. Luciani, and J.-L. Florens, "CORDIS-ANIMA: a modeling and simulation system for sound and image synthesis: the general formalism," *Computer Music Journal*, vol. 17, no. 1, pp. 19–29, 1993.

[4] J. Villeneuve and J. Leonard, "Mass-interaction physical models for sound and multi-sensory creation: Starting anew," in *Proceedings of the 16th Sound and Music Computing Conference*, 2019.

[5] J. Leonard and J. Villeneuve, "MI-GEN~: An efficient and accessible mass interaction sound synthesis toolbox," *Proceedings of the 16th Sound and Music Computing Conference (SMC)*, 2019.

[6] K. Karplus and A. Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal*, vol. 7, pp. 43–55, 1983.

[7] J. O. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[8] J. D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[9] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," M.S. thesis, University of Illinois, 1969.

[10] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society (JASA)*, vol. 19, no. 6, pp. 462–470, 1971.

[11] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, John Wiley & Sons, 2009.

[12] T. Tolonen, V. Välimäki, and M. Karjalainen, "Modeling of tension modulation nonlinearity in plucked strings," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 3, pp. 300–310, 2000.

[13] M. Ducceschi and S. Bilbao, "Simulation of the geometrically exact nonlinear string via energy quadratisation," *Journal of Sound and Vibration*, vol. 534, 2022.

[14] S. Willemsen, *The Emulated Ensemble: Real-Time Simulation of Musical Instruments using Finite-Difference Time-Domain Methods*, Ph.D. thesis, Aalborg University Copenhagen, Jul. 2021.

[15] S. Willemsen, "Nonlinear Mass Spring DAFx23," Available at https://github.com/SilvinWillemsen/NonlinearMassSpring_DAFx23, accessed April 7, 2023.

7. APPENDIX A: AN ALTERNATIVE DISCRETISATION

Using the centred averaging operator

$$\mu_{t\cdot} u^n = \frac{1}{2}\left(u^{n+1} + u^{n-1}\right), \tag{36}$$

an alternative discretisation of the nonlinear term in Eq. (11) (as opposed to Eq. (14)) can be written according to[2]

$$\begin{aligned}\delta_{tt}\mathbf{u}_m^n = &\frac{K}{M}\left(\mathbf{u}_{m+1}^n - 2\mathbf{u}_m^n + \mathbf{u}_{m-1}^n\right)\\ &- \frac{Kl_0}{M}\left(\mu_{t\cdot}\boldsymbol{\mathcal{U}}_{m+1/2}^n - \mu_{t\cdot}\boldsymbol{\mathcal{U}}_{m-1/2}^n\right)\\ &- 2\sigma\delta_{t\cdot}\mathbf{u}_m^n + 2z\delta_{t\cdot}\left(\mathbf{u}_{m+1}^n - 2\mathbf{u}_m^n + \mathbf{u}_{m-1}^n\right).\end{aligned} \tag{37}$$

Although this makes the system fully implicit, preliminary results show that this can reduce chaotic behaviour in some situations. However, due to its implicit nature, this scheme takes much longer to compute than the scheme in Eq (14).

[2] Notice that the last term in Eq. (37) now also uses a $\delta_{t\cdot}$ operator.

EFFICIENT FINITE-DIFFERENCE ROOM ACOUSTICS SIMULATION INCORPORATING EXTENDED-REACTING ELEMENTS

Jan W. Smits

Independent Researcher
jws@possumstudios.com

ABSTRACT

A method is proposed that allows finite-difference (FD) simulation of room acoustics to incorporate extended-reacting porous elements without adding major computational cost. The porous elements are described by a rigid-frame equivalent fluid model and are incorporated into the time-domain formulation through auxiliary differential equations. By using a local staggered grid scheme for the boundaries of the porous elements, the method allows an efficient second-order scalar approach to be used for the uniform air and porous element interior regions that make up the majority of the computational domain. Both the scalar and staggered schemes are based on a face-centered cubic grid to minimize numerical dispersion. A software implementation running on GPU shows the accuracy of the method compared to a theoretical reference, and demonstrates the method's computational efficiency through a benchmark example.

1. INTRODUCTION

The acoustic simulation of enclosed spaces is important in many applications, from the architectural design of performance halls and recording studios to the production of synthetic audio effects for music, cinema and virtual acoustics. In large rooms, such as performance halls, these simulations are usually performed using a geometrical acoustics approach such as ray or beam tracing [1]. In smaller rooms wave-based methods become necessary to accurately represent diffraction and modal behavior, but unfortunately such methods are computationally intensive. A range of wave-based methods has been applied to this problem, with finite difference (FD) [2] [3] [4] amongst the most popular, thanks in part to its suitability for implementation on GPU [5] [6] [7] [8].

One of the challenges in room acoustics is to accurately model the impact of various absorptive elements and surfaces in a room. This can include furniture and wall and floor coverings, as well as treatment panels or modules that are purposely added to alter the acoustics of the space. The behavior of such elements is often approximated with a locally reacting boundary assumption that simplifies the analysis [9] [10], but which may introduce significant errors. Problems appear for example when absorbing panels are present that have significant air gaps [11] [12] [13]. A more accurate approach is then to compute the 3-D acoustic propagation inside of any porous media as part of the overall simulation.

FD methods have been described that calculate such extended reaction in 2-D [14] and 3-D [15], making use of idealized models for the porous medium and conventional staggered Cartesian grids. More recently, 3-D discontinous Galerkin methods have been described that allow for a more general equivalent-fluid model (EFM) [16] [17], whereas EFM-based FD methods have been shown in up to two dimensions [18] [19]. Meanwhile, non-Cartesian FD approaches have been demonstrated that improve computational efficiency [20] [21] [22], but these schemes have not yet been adapted to allow the modeling of extended reaction.

The aim of this paper is to describe a FD method that extends the non-Cartesian face-centered cubic (FCC) scheme to include the simulation of porous media described by a general frequency-dependent EFM, thus enabling the simulation of extended-reacting elements with higher computational and memory efficiency than previous approaches. To the author's knowledge, this is the first time that a wave-based room acoustics method incorporating extended reaction is demonstrated at full audio bandwidth.

The paper is organized as follows. Section 2 describes the background of compact FD schemes, including a short discussion on dispersion error and computational efficiency. Section 3 then describes the internal porous volume update that is part of the new approach, and this is followed by a description of the porous volume boundary updates in section 4.2. The latter section also details the staggered FCC grid that is used at the boundaries. Section 5 shows results obtained through a GPU-based software implementation, and section 6 provides a short summary of conclusions.

2. FINITE-DIFFERENCE SCHEMES

2.1. Yee scheme

Room acoustics analysis usually starts with the linearized equations of continuity and conservation of momentum

$$\partial_t p = -\rho_0 c^2 \nabla \cdot \boldsymbol{v} \tag{1}$$

$$\rho_0 \partial_t \boldsymbol{v} = -\nabla p \tag{2}$$

where p represents the scalar pressure field, and $\boldsymbol{v}$ the velocity vector field. The constants ρ_0 and c represent the static fluid density ($\mathrm{kg/m^3}$) and propagation speed of sound ($\mathrm{m/s}$) respectively.

To achieve satisfactory results at higher frequencies, viscothermal losses can be incorporated through a post-processing step as described in [23] or [24], which avoids impacting the complexity of the computations through addition of a loss term in (2).

A common approach to solving the first-order system of Eqs.(1) (2) is to discretize them on a so-called Yee grid [25], where the pressure and particle velocities are interlaced in both space and time [4], [2]:

$$\delta_{t+} \underline{p}_{\boldsymbol{i}}^{n} = -\rho_0 c^2 (\delta_{\boldsymbol{x}-})^T \underline{\boldsymbol{v}}_{\boldsymbol{i}+\frac{1}{2}}^{n+\frac{1}{2}} \tag{3}$$

$$\rho_0 \delta_{t-} \underline{\boldsymbol{v}}_{\boldsymbol{i}+\frac{1}{2}}^{n+\frac{1}{2}} = -\delta_{\boldsymbol{x}+} \underline{p}_{\boldsymbol{i}}^{n} \tag{4}$$

The underlined variables in these formulas denote grid functions that approximate their continuous counterparts:

$$\underline{p}_{\boldsymbol{i}}^{n} \approx p(\boldsymbol{i}h, nk) \tag{5}$$

$$\underline{\boldsymbol{v}}_{\boldsymbol{i}+\frac{1}{2}}^{n+\frac{1}{2}} \approx \boldsymbol{v}_{\left((i_x+\frac{1}{2})h,(i_y+\frac{1}{2})h,(i_z+\frac{1}{2})h,nk\right)} \tag{6}$$

where k and h are the temporal and spatial grid steps respectively, and $\boldsymbol{i}$ and n are discrete indices:

$$\boldsymbol{i} := (i_x, i_y, i_z) \in \mathbb{Z}^3, \quad n \in \mathbb{Z}^+ \tag{7}$$

The operators δ_{t+} and δ_{t-} are first-order forward and backward difference operators defined here as

$$\delta_{t+}\underline{p}^{n} := \frac{1}{k}\left(\underline{p}^{n+1} - \underline{p}^{n}\right), \quad \delta_{t-}\underline{p}^{n} := \frac{1}{k}\left(\underline{p}^{n} - \underline{p}^{n-1}\right) \tag{8}$$

and $\delta_{\boldsymbol{x}+}$ and $\delta_{\boldsymbol{x}-}$ stand for spatial difference (vector) operators. Throughout this paper, bold emphasis will be used to distinguish vector quantitites and operators from scalar ones.

2.2. Standard leapfrog scheme

Eqs. (1) and (2) can also be combined to form a scalar wave equation for the pressure:

$$\partial_{tt} p = c^2 \Delta p \tag{9}$$

This second-order equation can then be discretized as

$$\delta_{tt}\underline{p} = c^2 \delta_{\Delta}\underline{p} \tag{10}$$

where δ_{tt} is a second-order time-difference operator defined as

$$\delta_{tt}\underline{p}_{\boldsymbol{i}}^{n} = \frac{1}{k^2}(\underline{p}_{\boldsymbol{i}}^{n+1} - 2\underline{p}_{\boldsymbol{i}}^{n} + \underline{p}_{\boldsymbol{i}}^{n-1}) \tag{11}$$

and δ_{Δ} is a discrete Laplacian operator

$$\delta_{\Delta}\underline{p}_{\boldsymbol{i}}^{n} = \frac{1}{h^2}\left(Q_{\boldsymbol{i}}^{n} - 6\underline{p}_{\boldsymbol{i}}^{n}\right) \tag{12}$$

with $Q_{\boldsymbol{i}}^{n}$ representing the sum of the nearest neighbors of $\underline{p}_{\boldsymbol{i}}^{n}$:

$$Q_{\boldsymbol{i}}^{n} := \underline{p}_{\boldsymbol{i}+\boldsymbol{e}_x}^{n} + \underline{p}_{\boldsymbol{i}-\boldsymbol{e}_x}^{n} + \underline{p}_{\boldsymbol{i}+\boldsymbol{e}_y}^{n} + \underline{p}_{\boldsymbol{i}-\boldsymbol{e}_y}^{n} + \underline{p}_{\boldsymbol{i}+\boldsymbol{e}_z}^{n} + \underline{p}_{\boldsymbol{i}-\boldsymbol{e}_z}^{n} \tag{13}$$

Substituting (11) and (12) in (10) and using the Courant number $\lambda := ck/h$, leads to the scalar update equation

$$\underline{p}_{\boldsymbol{i}}^{n+1} = -\underline{p}_{\boldsymbol{i}}^{n-1} + (2 - 6\lambda^2)\underline{p}_{\boldsymbol{i}}^{n} + \lambda^2 Q_{\boldsymbol{i}}^{n} \tag{14}$$

This formula is often referred to as the "standard leapfrog" scheme [26], abbreviated as SLF. In [27] it is shown that this scheme and the Yee scheme are equivalent, and produce results that are identical to within machine precision. The SLF scheme is usually preferable over the Yee scheme, because it requires less computer memory and fewer computational operations [27].

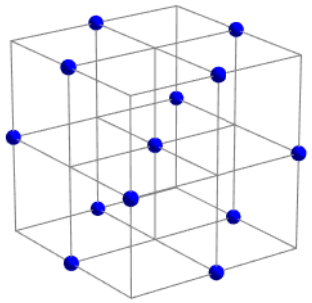


Figure 1: *Pressure field positions of a face-centered cubic (FCC) grid, shown in relation to an associated Cartesian grid.*

2.3. Face-centered cubic scheme

Other schemes for the wave equation can be formulated by considering expanded stencils for the Laplacian operator [20]. A compact stencil of particular interest for room acoustics is the face-centered cubic (FCC) stencil [28] [21], so called because its nodes lie on a non-Cartesian face-centered cubic grid, which is defined by:

$$\boldsymbol{i} := (i_x, i_y, i_z) \in \{\mathbb{Z}^3 : (i_x + i_y + i_z) \pmod 2 = 0\} \tag{15}$$

From the illustration in Figure 1 it can be seen that each node on the FCC grid is surrounded by twelve nearest neighbors, resulting in a 13-point compact Laplacian stencil. If we use the vector $\boldsymbol{l}_m$ to represent the relative coordinates of the nearest neighbors of a given grid node, the Laplacian operator for the FCC scheme can be written as:

$$\delta_{\Delta}\underline{p}_{\boldsymbol{i}}^{n} = \frac{1}{4h^2}\left(\sum_{m=1}^{12} \underline{p}_{\boldsymbol{i}+\boldsymbol{l}_m}^{n} - 12\underline{p}_{\boldsymbol{i}}^{n}\right) \tag{16}$$

By substituting $\underline{P}_{\boldsymbol{i}}^{n} := \sum_{m=1}^{12} \underline{p}_{\boldsymbol{i}+\boldsymbol{l}_m}^{n}$, this can be simplified to:

$$\delta_{\Delta}\underline{p}_{\boldsymbol{i}}^{n} = \frac{1}{4h^2}\left(\underline{P}_{\boldsymbol{i}}^{n} - 12\underline{p}_{\boldsymbol{i}}^{n}\right) \tag{17}$$

Then, using the second-order time operator of (11), the update equation for the FCC scheme follows as:

$$\underline{p}_{\boldsymbol{i}}^{n+1} = \frac{\lambda^2}{4}(\underline{P}_{\boldsymbol{i}}^{n} - 12\underline{p}_{\boldsymbol{i}}^{n}) + 2\underline{p}_{\boldsymbol{i}}^{n} - \underline{p}_{\boldsymbol{i}}^{n-1} \tag{18}$$

The principal benefit of the FCC scheme lies in the fact that it exhibits higher computational efficiency than other compact FD schemes, as shown by Hamilton and Bilbao in [21]. To achieve a fair comparison, the analysis in the reference maximizes the grid steps in each scheme for a given numerical dispersion error, while also satisfying the stability constraint of each method.

A recent study [29] finds that the threshold of human perception lies around 2% dispersion error for spaces with shorter echo times, as typically found in small acoustically treated spaces such as mixing rooms. At this error level, the analysis in [21] finds that the FCC scheme is 11.3 times more efficient than the SLF scheme, and this ratio will be around double when FCC is compared to the Yee scheme [27]. Because all previous FD methods for extended reaction have been based on the Yee scheme [14] [15] [18] [19], this provides clear motivation for the development of an FCC-based approach .

3. EXTENDED-REACTION FD SCHEME

3.1. Equivalent fluid model

For the purpose of room acoustics, the propagation of sound in porous media is commonly described with a rigid-frame equivalent-fluid model (EFM) [11]. In this approximation, the frame of the material is assumed motionless, and the air inside it is replaced macroscopically by an equivalent free fluid with a complex bulk modulus and effective density that both depend on frequency. The acoustic equations for the domain Ω_P of the porous volume, can then be formulated in the frequency domain as

$$j\omega\hat{p} = -\hat{K}(\omega)\nabla\cdot\hat{\boldsymbol{v}}, \quad \boldsymbol{x}\in\Omega_P \tag{19}$$

$$j\omega\hat{\boldsymbol{v}} = -\hat{R}(\omega)\nabla\hat{p}, \quad \boldsymbol{x}\in\Omega_P \tag{20}$$

Here, $\hat{\boldsymbol{v}}$ and $\hat{p}$ denote Fourier-transforms of the corresponding time-domain variables. The complex frequency-dependent quantities $\hat{K}(\omega)$ and $\hat{R}(\omega)$ represent the EFM estimates for the medium's effective bulk modulus in N/m^2 and the inverse of its effective density in m^3/kg respectively. Combining (19) and (20) leads to the Helmholtz equation

$$\omega^2\hat{p} = -\hat{R}(\omega)\hat{K}(\omega)\Delta\hat{p}, \quad \boldsymbol{x}\in\Omega_P \tag{21}$$

A primary condition for the validity of the EFM is that wavelengths are much larger than the characteristic dimensions of the pores [11].

A variety of EFMs have been proposed based on empirical and/or phenomenological justifications; see for example sections 2.5, 5.4 and 5.5 of [11]. For the results in this paper, the Allard-Champoux model was used, described in [30]. With ambient conditions defined by a static pressure $\rho_0 = 1.2\mathrm{kg/m^3}$, a Prandtl number of 0.702, an adiabatic index of 1.40 and a static pressure of 101,320 N/m^2, Eqs. (5) and (6) of [30] provide the model formulas as:

$$\hat{R}(\omega)^{-1} = 1.2 + \left(-0.0364X^{-2} - j0.1144X^{-1}\right)^{1/2} \tag{22}$$

$$\hat{K}(\omega) = 101320\frac{j29.64 + \left(2.82X^{-2} + j24.9X^{-1}\right)^{1/2}}{j21.17 + \left(2.82X^{-2} + j24.9X^{-1}\right)^{1/2}} \tag{23}$$

The intermediate variable X in these formulas is defined as $X := \rho_0 f/\sigma$, with $f = \omega/2\pi$ as the frequency in Hz, and σ representing the flow resistivity of the material in $\mathrm{Nm^{-4}s}$.

3.2. Auxiliary differential equations (ADE) method

In order to solve either (21) or the combination of (19) and (20) in the time domain, the method of auxiliary differential equations (ADE method) will be used, described in [31] and previously applied in [16], [17], [32] and [19], amongst others.

The ADE method consists in approximating the inverse effective density and effective bulk modulus with limited-order rational functions in the frequency domain, then formulating the inverse Fourier-transformed system with a set of additional state variables called accumulators, and determining these accumulators through a set of auxiliary differential equations that are solved alongside the acoustical equation(s). The benefit of this approach is that it avoids computationally expensive convolutions in the time domain, instead performing a limited number of additional state variable computations.

Due to the passive and non-resonant nature of conventional porous materials, it is sufficient to approximate the equivalent fluid properties with only real poles [32], such that the partial fraction expansions can be written as follows

$$\hat{K} \approx \hat{K}_\infty + \sum_{m=1}^{\mathcal{M}_K}\frac{A_{K,m}}{\eta_{K,m} + j\omega} \tag{24}$$

$$\hat{R} \approx \hat{R}_\infty + \sum_{m=1}^{\mathcal{M}_R}\frac{A_{R,m}}{\eta_{R,m} + j\omega} \tag{25}$$

for $\mathcal{M}_K$ and $\mathcal{M}_R$ fractions respectively. The poles $(-\eta)$, residues A, and high-frequency limit values $\hat{K}_\infty$, $\hat{R}_\infty$ in this approximation are found through a fitting procedure such as the method of vector-fitting [33] used here. For stability, the poles should be constrained to be negative or zero ($\eta \geq 0$) [1]

It will also be useful to approximate the product of $\hat{R}$ and $\hat{K}$, in a similar manner:

$$\hat{R}\hat{K} \approx \hat{R}_\infty\hat{K}_\infty + \sum_{m=1}^{\mathcal{M}_{RK}}\frac{A_{RK,m}}{\eta_{RK,m} + j\omega} \tag{26}$$

Using this expression, an approximation for the inverse Fourier transform of (21) can be obtained as

$$\partial_{tt}p(t) \approx \hat{R}_\infty\hat{K}_\infty\Delta p(t) + \sum_{m=1}^{\mathcal{M}_{RK}} A_{RK,m}\phi_{RK,m}(t) \tag{27}$$

where $\phi_{RK,m}$ are accumulator variables that satisfy the auxiliary differential equations defined as:

$$\partial_t\phi_{RK,m} + \eta_{RK,m}\phi_{RK,m} = \Delta p, \forall m \in [1, \mathcal{M}_{RK}] \tag{28}$$

3.3. Porous medium update in FCC scheme

Let us now define three non-overlapping sets of nodes for the finite-difference grid. An interior porous volume set $\mathcal{P}_p$ is defined as the set of grid nodes that are inside of the porous region and for which all the nearest neighbors are also inside the region. The air interior set $\mathcal{P}_a$ similarly contains all nodes that are in air and whose neighbors are in air also. Finally, the boundary set $\mathcal{P}_b$ consists of all nodes which are separated from at least one neighbor by a porous volume boundary.

For the nodes in $\mathcal{P}_p$, a finite-difference update can then be derived in two steps. In the first step, grid functions $\underline{\phi}_{RK,m}$ are computed to approximate the continuous accumulators $\phi_{RK,m}$ by evaluating a discretization of (28) at each time step:

$$\underline{\phi}^{n+\frac{1}{2}}_{RK,m,\boldsymbol{i}} = \frac{(2 - k\eta_{RK,m})}{(2 + k\eta_{RK,m})}\underline{\phi}^{n-\frac{1}{2}}_{RK,m,\boldsymbol{i}} + \frac{2k}{(2 + k\eta_{RK,m})}\delta_\Delta\underline{p}^n_{\boldsymbol{i}}, \quad \forall m \in [1, \mathcal{M}_{RK}] \tag{29}$$

where δ_Δ is the discrete Laplacian operator from (16), and ∂_t has been approximated with the δ_{t+} operator defined in (8).

For the second step, an update formula of the pressure values is derived from (27) by replacing p with the grid function $\underline{p}$, applying

[1] Note that the $(e^{+j\omega t})$ time convention used here is opposite that of the negative time convention used in [31], explaining the different sign appearing in the fraction denominators in Eqs. (24) (25) (26).

the second-order operators defined in (11) and (16), and interpolating between $\underline{\phi}^{n+\frac{1}{2}}_{RK,m,\boldsymbol{i}}$ and $\underline{\phi}^{n-\frac{1}{2}}_{RK,m,\boldsymbol{i}}$ to ensure that the accumulator term is centered at time index n:

$$\underline{p}^{n+1}_{\boldsymbol{i}} = \frac{k^2 \hat{R}_\infty \hat{K}_\infty}{4h^2}(\underline{P}^n_{\boldsymbol{i}} - 12\underline{p}^n_{\boldsymbol{i}}) + 2\underline{p}^n_{\boldsymbol{i}} - \underline{p}^{n-1}_{\boldsymbol{i}} + \frac{k^2}{2}\sum_{m=1}^{\mathcal{M}_{RK}} A_{RK,m}\left(\underline{\phi}^{n+\frac{1}{2}}_{RK,m,\boldsymbol{i}} + \underline{\phi}^{n-\frac{1}{2}}_{RK,m,\boldsymbol{i}}\right) \quad (30)$$

4. POROUS-VOLUME BOUNDARY UPDATE

4.1. Staggered FCC grid

Because the material properties are discontinuous across boundaries of the porous volume, the wave-equation based approach of section 3.3 can not be used for the nodes in $\mathcal{P}_b$. Instead, a staggered scheme will be used that makes it possible to formulate a finite-difference approximation of the first-order system of Eqs. (19) (20). This scheme is part of a family of staggered formulations that can be derived from a finite-volume framework for isohedral cell shapes, described in [34].

For this study, the scheme will be described using a staggered grid that is created by complementing an FCC pressure grid with a set of six velocity subgrids. The velocity subgrids are each created by translating the nodes $\boldsymbol{i}$ of the FCC grid with one of six grid offset vectors $\boldsymbol{j}_l$, defined as

$$\begin{aligned} \boldsymbol{j}_1 &:= \frac{1}{2}\begin{pmatrix}1\\1\\0\end{pmatrix}, \quad \boldsymbol{j}_2 := \frac{1}{2}\begin{pmatrix}0\\1\\1\end{pmatrix}, \quad \boldsymbol{j}_3 := \frac{1}{2}\begin{pmatrix}1\\0\\1\end{pmatrix} \\ \boldsymbol{j}_4 &:= \frac{1}{2}\begin{pmatrix}1\\-1\\0\end{pmatrix}, \quad \boldsymbol{j}_5 := \frac{1}{2}\begin{pmatrix}0\\1\\-1\end{pmatrix}, \quad \boldsymbol{j}_6 := \frac{1}{2}\begin{pmatrix}1\\0\\-1\end{pmatrix} \end{aligned} \quad (31)$$

Scalar velocity grid functions $\underline{v}_{l,\boldsymbol{i}+\boldsymbol{j}_l}$ can be defined on each of these six subgrids to approximate the value of the velocity field in the direction of its associated grid vector:

$$\underline{v}_{l,\boldsymbol{i}+\boldsymbol{j}_l} \approx \boldsymbol{v}(h\boldsymbol{i} + h\boldsymbol{j}_l) \cdot (\boldsymbol{j}_l/\|\boldsymbol{j}_l\|) \quad (32)$$

The staggered grid formed by the combination of these subgrids is depicted in Figure 2. Similarly to the Yee grid, the velocity points are located at the midpoint between adjacent pressure nodes and only a single component is stored for each velocity point.

Using this grid, we can now define a discrete velocity divergence operator as

$$(\delta_{\boldsymbol{x}-} \cdot \underline{v})_{\boldsymbol{i}} := \frac{1}{2\sqrt{2}h}\sum_{l=1}^{6}\left(\underline{v}_{l,\boldsymbol{i}+\boldsymbol{j}_l} - \underline{v}_{l,\boldsymbol{i}-\boldsymbol{j}_l}\right) \quad (33)$$

and a discrete gradient operator for the pressure as

$$\left(\delta_{\boldsymbol{x}+}\underline{p}_{\boldsymbol{i}}\right)_l := \frac{1}{\sqrt{2}h}\left(\underline{p}_{\boldsymbol{i}+2\boldsymbol{j}_l} - \underline{p}_{\boldsymbol{i}}\right) \quad (34)$$

Applying these operators to (1) and (2) results in a staggered FD scheme as follows:

$$\delta_{t+}\underline{p}^{n+1}_{\boldsymbol{i}} = -\frac{\rho_0 c^2}{2\sqrt{2}h}\sum_{l=1}^{6}\left(\underline{v}^{n+\frac{1}{2}}_{l,\boldsymbol{i}+\boldsymbol{j}_l} - \underline{v}^{n+\frac{1}{2}}_{l,\boldsymbol{i}-\boldsymbol{j}_l}\right) \quad (35)$$

$$\delta_{t-}\underline{v}^{n+\frac{1}{2}}_{l,\boldsymbol{i}+\boldsymbol{j}_l} = -\frac{1}{\rho_0\sqrt{2}h}\left(\underline{p}^n_{\boldsymbol{i}+2\boldsymbol{j}_l} - \underline{p}^n_{\boldsymbol{i}}\right) \quad (36)$$

The equivalence derived more generally in [34] can be verified by applying a δ_{t-} operator to (35) and substituting (36). This leads to the following second-order FD expression:

$$\delta_{t-}\delta_{t+}\underline{p}^{n+1}_{\boldsymbol{i}} = \frac{c^2}{4h^2}\sum_{l=1}^{6}\left(\left(\underline{p}^n_{\boldsymbol{i}+2\boldsymbol{j}_l} + \underline{p}^n_{\boldsymbol{i}-2\boldsymbol{j}_l}\right) + 12\underline{p}^n_{\boldsymbol{i}}\right) \quad (37)$$

which, upon substitution of (8) for the difference operators results in the same expression as (18).

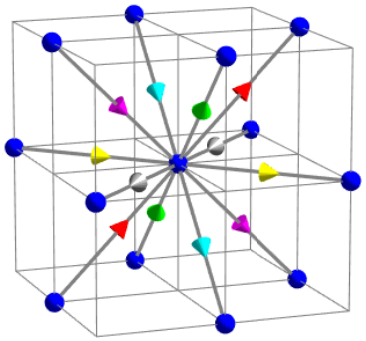


Figure 2: *Depiction of the local staggered face-centered (FCC) grid. The blue spheres denote the pressure subgrid, and different colors are used to distinguish each of the six velocity subgrids. For visual simplicity the plot only shows the twelve velocity nodes that are directly adjacent to the central pressure node.*

4.2. Staggered-scheme boundary update

The system to be solved on the boundary is obtained by applying the ADE method to the inverse Fourier transforms of Eqs. (19) and (20), with use of the rational approximations (24) and (25):

$$\partial_t p = -\hat{K}_\infty \nabla \cdot \boldsymbol{v} - \sum_{m=1}^{\mathcal{M}_K} A_{K,m}\phi_{K,m} \quad (38)$$

$$\partial_t \boldsymbol{v} = -\hat{R}_\infty \nabla p - \sum_{m=1}^{\mathcal{M}_R} A_{R,m}\boldsymbol{\phi}_{R,m} \quad (39)$$

where

$$\partial_t \phi_{K,m} + \eta_{K,m}\phi_{K,m} = \nabla \cdot \boldsymbol{v}, \quad \forall m \in [1, \mathcal{M}_K] \quad (40)$$

and

$$\partial_t \boldsymbol{\phi}_{R,m} + \eta_{R,m}\boldsymbol{\phi}_{R,m} = \nabla p, \quad \forall m \in [1, \mathcal{M}_R] \quad (41)$$

A FD formulation can now be obtained for the pressure nodes in $\mathcal{P}_b$ by applying the divergence and gradient operators from (33) and (34) to Eqs. (38) thru (41), resulting in

$$\underline{p}^{n+1}_{\boldsymbol{i}} = \underline{p}^n_{\boldsymbol{i}} - k\hat{K}_\infty\left(\delta_{\boldsymbol{x}-} \cdot \underline{v}^{n+\frac{1}{2}}\right)_{\boldsymbol{i}} - \frac{k}{2}\sum_{m=1}^{\mathcal{M}_K} A_{K,m}\left(\underline{\phi}^{n+1}_{K,m,\boldsymbol{i}} + \underline{\phi}^n_{K,m,\boldsymbol{i}}\right) \quad (42)$$

$$\underline{v}_{l,\boldsymbol{i}+\boldsymbol{j}_l}^{n+\frac{1}{2}} = \underline{v}_{l,\boldsymbol{i}+\boldsymbol{j}_l}^{n-\frac{1}{2}} - k\hat{R}_\infty \left(\delta_{\boldsymbol{x}+} \underline{p}_{\boldsymbol{i}}^{n}\right)_l - \frac{k}{2} \sum_{m=1}^{\mathcal{M}_R} A_{R,m} \left(\underline{\phi}_{R,m,\boldsymbol{i},l}^{n+\frac{1}{2}} + \underline{\phi}_{R,m,\boldsymbol{i},l}^{n-\frac{1}{2}}\right) \quad (43)$$

and the associated accumulator updates

$$\underline{\phi}_{K,m,\boldsymbol{i}}^{n+1} = \frac{(2-k\eta_{K,m})}{(2+k\eta_{K,m})} \underline{\phi}_{K,m,\boldsymbol{i}}^{n} + \frac{2k}{(2+k\eta_{K,m})} \left(\delta_{\boldsymbol{x}-} \cdot \underline{\boldsymbol{v}}^{n+\frac{1}{2}}\right)_{\boldsymbol{i}}, \quad \forall m \in [1, \mathcal{M}_K] \quad (44)$$

$$\underline{\phi}_{R,m,\boldsymbol{i},l}^{n+\frac{1}{2}} = \frac{(2-k\eta_{R,m})}{(2+k\eta_{R,m})} \underline{\phi}_{R,m,\boldsymbol{i},l}^{n-\frac{1}{2}} + \frac{2k}{(2+k\eta_{R,m})} \left(\delta_{\boldsymbol{x}+} \underline{p}_{\boldsymbol{i}}^{n}\right)_l, \quad \forall m \in [1, \mathcal{M}_R] \quad (45)$$

In these equations, the accumulator variables for the continuity and momentum equations are shifted by $k/2$ from each other in time in order to ensure that both updates remain centered.

Figure 3 depicts a 2-D cross section of the grid, illustrating the transition between the regular FCC grids used in the air and porous volume sections, and the staggered FCC grid used at the boundaries. The local (staggered) grid needs to include at a minimum all pressure nodes that are separated from one or more of their nearest neighbors by a porous-volume boundary. For each of these pressure nodes, the evaluation of (42) requires that all the adjacent velocity values are computed and stored as well.

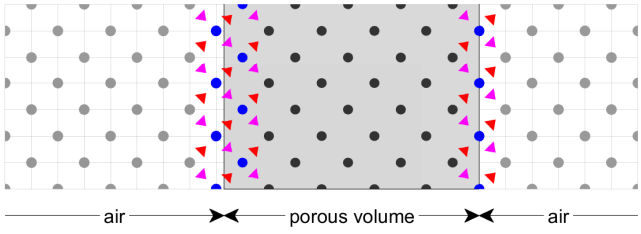


Figure 3: *2-D cross section showing the local staggered grid at the boundaries of a porous volume. The blue spheres indicate the pressure nodes of the local grid ($\mathcal{P}_b$), and the red and violet triangles indicate the associated velocity nodes in the viewing plane. The internal pressure nodes of the air ($\mathcal{P}_a$) and porous volumes ($\mathcal{P}_p$) are shown in light and dark grey respectively. A raster in the background shows how the nodes are positioned relative to the Cartesian grid spacing (h).*

4.3. Rigid boundary conditions

At the edges of the domain, rigid boundary conditions are applied that enforce zero particle velocity normal to the boundary: $v_\perp = 0$. In the domain covered by the staggered grid this is implemented simply by setting the corresponding velocity values resulting from (43) to zero, prior to the evaluation of (42). In the air and porous volume internal domains, the boundary condition is implemented by modifying the Laplacian operator: any "ghost points" that are located across from a boundary are removed from the summation in (16), and for each removal, the constant factor 12 on the right is reduced by 1.

5. PRACTICAL RESULTS

5.1. Implementation

The method described in this paper was implemented as a C++ computer program, making use of the CUDA programming interface to access GPU compute capabilities. New CUDA Kernels were developed for the updates (30) (29) on the porous volume internal domain and updates (35) (36) (44) (45) along with updates for the rigid boundary conditions adjacent to porous volumes. Kernels for the air update (18) and for rigid boundary conditions next to the air domain were borrowed from Hamilton's open-source FD program named PFFDTD to save development effort [35].

In order to facilitate efficient GPU data access on the staggered grid, linked data structures were constructed that avoid expensive spatial search operations. The preparation of this data from input geometry was implemented making use of the OpenVDB library of sparse volumetric data structures and tools [36].

5.2. Comparison against theory

An absorber configuration is considered at normal and 45° angles of incidence θ, as shown in Fig. 4. The configuration consists of a 10cm thick porous panel that is separated by a 20cm thick air gap from a rigid boundary. The height and width of the domain are both 16m, and its length is 5.3m for the case of normal incidence, and 18.3m for the oblique incidence case. Initial conditions are set up to generate a plane wave with a 4cm wide raised cosine shape, starting at 10cm to the left of the left-most absorber edge. The pressure is computed at the midpoint on the surface of the porous panel and is truncated in time to avoid contamination by spurious reflections due to the finite domain and absorber sizes. The grid step (h) is 1 cm. The porous medium properties are defined by Eqs. (22) (23) with $\rho_0 = 1.2\text{kg/m}^3$ and a flow resistivity value of $\sigma = 10,000\text{Nm}^{-4}\text{s}$, which is situated in the typical range for commonly used reticulated foams.

The coefficients of Eqs. (24) (25) (26) were found by using the method of vector fitting [33], in which two poles were used to fit each of $\hat{R}$ and $\hat{K}$, and three poles were used to fit the product $\hat{R}\hat{K}$. It was found that for the above parameter values, these fit orders were sufficient to fit the Allard-Champoux model within 0.32% maximum relative error over a frequency range of 20 to 4,000 Hz.

In order to compute the surface impedance of the absorber, an incident-wave pressure p_i is first computed by removing the absorber from the simulation. This response is than subtracted from the response with the absorber to yield the reflected-wave pressure p_r. By taking the FFT of both p_r and p_i and then dividing the two, a frequency-domain reflection coefficient $\hat{R}$ is found, from which the surface impedance $\hat{Z}$ and absorption coefficient α can be computed through [11]:

$$\hat{Z} = \frac{\rho_0 c \left(1+\hat{R}\right)}{\cos\theta \left(1-\hat{R}\right)} \quad (46)$$

and

$$\alpha = 1 - ||\hat{R}||^2 \quad (47)$$

For comparison of the simulation against theory, an exact analytical formulation is used for the surface impedance of multilayered fluids at oblique incidence, provided in section 3.4 of [11]. The results along with the theoretical reference are shown in Figs.

5 and 6 for the incidence angles of 0° and 45° respectively. In both cases the absorption shows a close match to the theory.

A brief comment should be made here about errors due to the non-conforming grid, also referred to as staircase errors. Generally, an issue arises due to staircasing when absorbing local-reacting boundary conditions are applied, resulting in significant angle-dependent errors in the energy absorption, that do not reduce when the grid step is refined [10]. These errors are due to incorrect effective surface-area of the applied boundary condition. Because the extended-reaction absorber modeling described here is volumetric in nature, it does not exhibit this type of error, as supported by the fact that both modeled angles yield absorptions that are close to the theoretical values. On the other hand, a new type of error may appear when modeling thin structures, because the non-conforming grid can cause errors in the effective absorber thickness. Further analysis of this limitation should be conducted as part of follow-up work.

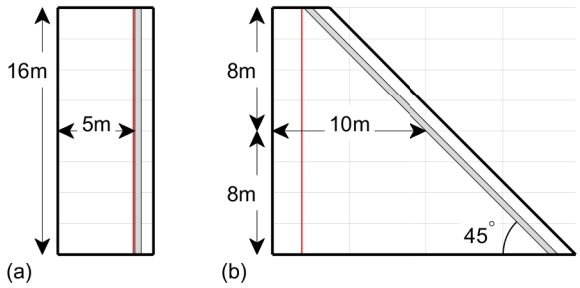


Figure 4: *Geometry of the validation test with gapped absorber. (a) Test case for normal incidence. (b) Test case for* 45° *incidence. The porous absorber thickness (10cm) and air gap size (20cm) are shown exaggerated four times for visual clarity. The thick lines around the outside represent rigid boundary conditions. The red vertical lines mark the position of the plane wave initialization, 10cm to the left of the left-most absorber dimension.*

5.3. Efficiency results

As a benchmark case for the computational and memory efficiency of the method, a configuration was used consisting of a rectangular room with a 10 cm thick porous panel hanging 20 cm below the ceiling, as well as a vertical free-standing porous panel with dimensions of 1.8 m height by 1.2 m width and 20 cm thickness. Dimensions of the room are $5\mathrm{m} \times 5\mathrm{m} \times 4\mathrm{m}$ yielding a total volume of 100 m^3, and the simulation was run at resolutions of $h = 10.0\mathrm{mm}$ and $h = 3.07\mathrm{mm}$ in terms of the Cartesian grid step, giving a maximum of 2% numerical dispersion for frequencies up to 6.2 kHz and up to 20 kHz respectively.

Figure 7 shows an early-time visualization of the wavefront propagation, where a point-source is used with a 20 cm wide single-cycle sine excitation. Partial transmission, reflection, and refraction of the pressure signal are visible at the porous barriers as expected.

The performance results for this benchmark model are summarized in Table 1 for impulse-response simulations with a duration of $1s$, executed on a RTX A5000 GPU with 24GB video RAM. The timing data shows that the addition of the extended-reaction modeling results in a moderate additional cost in compute,

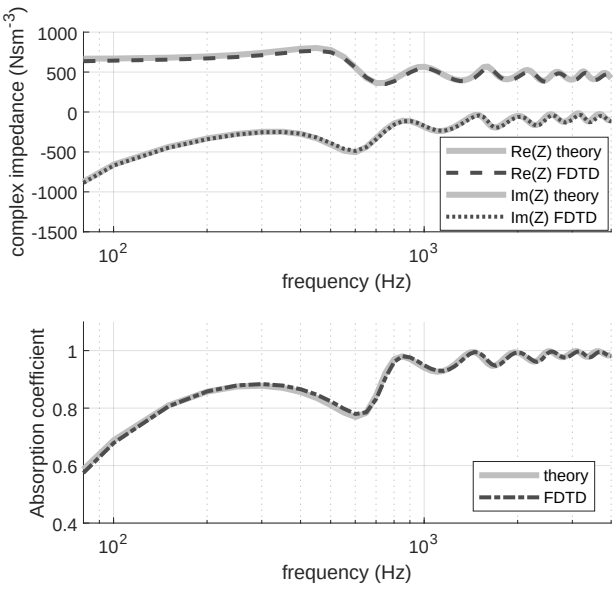


Figure 5: *Simulated impedance and absorption coefficient at interface of absorber for a normal incidence plane wave, with comparison to theory.*

with 27% of the total GPU time being spent on porous medium updates in the full-bandwidth simulation, and 45% in the 6.2 kHz case. The results imply that, for smaller-sized rooms at least, it is feasible with this method to perform simulations up to full audio frequency range on a single workstation GPU.

Experimentally, the porous volume updates were found to behave in a stable manner at least up to the stability limit $ck/h \leq 1$ of the FCC scheme in air. This result aligns with the fact that the wave propagation velocity in porous volumes is lower than in air. A formal analysis of the stability conditions should be undertaken as follow-up work.

6. CONCLUSIONS AND FUTURE WORK

In this paper a new finite-difference (FD) method was described for the computation of acoustical impulse responses in rooms that include rigid-frame porous media. By combining second-order face-centered cubic (FCC) updates in the uniform subdomains with a staggered grid formulation on the boundaries, the method makes it possible to include extended-reaction effects at only a limited extra computational cost.

An experimental GPU-based software implementation was described, and results were presented. A theoretical test case was shown to provide validation for the accuracy of the method and a benchmark case was used to quantify the computational and memory performance. The benchmark results demonstrate that it is possible to compute extended reaction in a small room up to full audio-bandwidth on a single GPU.

The method behaved in a stable manner during the experimental tests, however, a formal analysis of the stability conditions should be performed as part of follow-up work. Future work should also quantify the impact of stairstep error when modeling thin absorber structures.

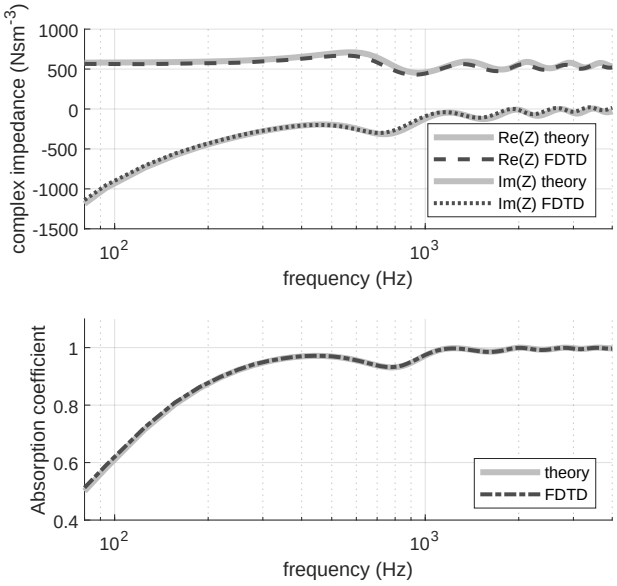


Figure 6: *Simulated impedance and absorption coefficient at interface of absorber for a plane wave at* 45° *incidence angle, with comparison to theory.*

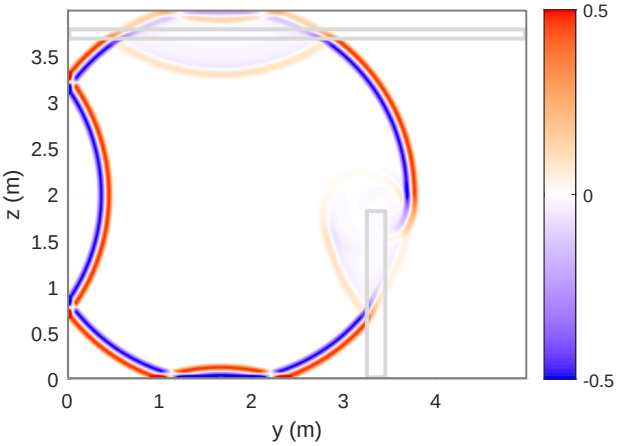


Figure 7: *Snapshot of acoustic wave propagation in a* $5\text{m} \times 5\text{m} \times 4\text{m}$ *(*100m^3*) benchmark room with ceiling absorber and free-standing porous panel, taken at* $t = 6\text{ms}$*. The colorscale is normalized to 50% of the maximum pressure values within the cross-section, giving some compression to improve visibility of the response around the porous volumes*

7. REFERENCES

[1] Lauri Savioja and Peter Svensson, "Overview of geometrical room acoustic modeling techniques," *J. Acoustical Society of America*, vol. 138, pp. 708–730, 08 2015.

[2] Osamu Chiba, Tatsuta Kashiwa, Hidemaro Shimoda, Shin Kagami, and Ichiro Fukai, "Analysis of sound fields in three dimensional space by the time-dependent finite-difference method based on the leap frog algorithm," *J. Acoustical Society of Japan*, vol. 49, no. 8, pp. 551–562, 1993.

[3] Lauri Savioja, Timo J. Rinne, and Tapio Takala, "Simulation of room acoustics with a 3-D finite difference mesh," in *Proc. Int. Computer Music Conf. (ICMC)*, Danish Institute of Electroacoustic Music, Denmark, 1994, pp. 463–466.

[4] Dick Botteldooren, "Finite-difference time-domain simulation of low-frequency room acoustic problems," *J. Acoustical Society of America*, vol. 98, no. 6, pp. 3302–3308, 1995.

[5] Lauri Savioja, "Real-time 3D finite-difference time-domain simulation of low- and mid-frequency room acoustics," in *Proc. Digital Audio Effects (DAFx)*, Graz, Austria, 2010, vol. 1, p. 75.

[6] Craig Webb and Stefan Bilbao, "Computing room acoustics with CUDA - 3D FDTD schemes with boundary losses and viscosity," in *Proc. IEEE ICASSP*, Prague, Czech Republic, 05 2011, pp. 317–320.

[7] Brian Hamilton, Stefan Bilbao, and Craig Webb, "Revisiting implicit finite difference schemes for 3-D room acoustics simulations on GPU," in *Proc. Digital Audio Effects (DAFx)*, Erlangen, Germany, 09 2014.

[8] Brian Hamilton, Craig Webb, Nathaniel Fletcher, and Stefan Bilbao, "Finite difference room acoustics simulation with general impedance boundaries and viscothermal losses in air: Parallel implementation on multiple GPUs," in *Proc. Int. Symp. on Music and Room Acoustics (ISMRA)*, Buenos Aires, Argentina, 09 2016.

[9] Konrad Kowalczyk and Maarten Van Walstijn, "Formulation of locally reacting surfaces in FDTD/K-DWM modelling of acoustic spaces," *Acta Acustica united with Acustica*, vol. 94, no. 6, pp. 891–906, 2008.

[10] Stefan Bilbao, Brian Hamilton, Jonathan Botts, and Lauri Savioja, "Finite volume time domain room acoustics simulation under general impedance boundary conditions," *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 24, no. 1, pp. 161–173, 2015.

[11] Jean Allard and Noureddine Atalla, *Propagation of sound in porous media: modelling sound absorbing materials*, John Wiley & Sons, second edition, 2009.

[12] Cheol-Ho Jeong, "Guideline for adopting the local reaction assumption for porous absorbers in terms of random incidence absorption coefficients," *Acta Acustica united with Acustica*, vol. 97, pp. 779–790, 09 2011.

[13] Kristrún Gunnarsdóttir, Cheol-Ho Jeong, and Gerd Marbjerg, "Acoustic behavior of porous ceiling absorbers based on local and extended reaction," *J. Acoustical Society of America*, vol. 137, no. 1, pp. 509–512, 2015.

[14] Hisaharu Suzuki, Akira Omorto, and Kyoji Fujiwara, "Treatment of boundary conditions by finite difference time domain method," *Acoustical Science and Technology*, vol. 28, pp. 16–26, 01 2007.

[15] Nuno Ferreira and Carl Hopkins, "Using finite-difference time-domain methods with a Rayleigh approach to model low-frequency sound fields in small spaces subdivided by porous materials," *Acoustical Science and Technology*, vol. 34, no. 5, pp. 332–341, 2013.

Table 1: *Timing results for tests run on a RTX A5000 GPU at 32-bit floating-point precision. The room volume is 100m^3 and the simulation length is 1 second. The quoted frequency ranges correspond to a maximum numeric dispersion error of 2%.*

frequency range	6.2 kHz	20 kHz	
Cartesian grid spacing, h	10.0	3.07	mm
time step size, k	29.2	8.94	µs
total number of time steps	34.3	111.8	$\times 10^3$
domain sizes:			
total number of nodes	50.0	1725	$\times 10^6$
rigid boundary nodes	1.90	20.5	$\times 10^6$
porous volume nodes	1.25	46.1	$\times 10^6$
porous volume boundary nodes	0.55	6.0	$\times 10^6$
video RAM used	0.6	16.2	GB
timing results:			
air update	31.6	4456	secs.
rigid boundaries update	11.0	408	secs.
other (overhead)	3.3	93	secs.
internal porous volume update	7.8	711	secs.
porous boundary update	30.3	1168	secs.
total time for updates	83.9	6837	secs.

[16] Finnur Pind, Cheol-Ho Jeong, Allan P Engsig-Karup, Jan S Hesthaven, and Jakob Strømann-Andersen, "Time-domain room acoustic simulations with extended-reacting porous absorbers using the discontinuous Galerkin method," *J. Acoustical Society of America*, vol. 148, no. 5, pp. 2851–2863, 2020.

[17] Huiqing Wang and Maarten Hornikx, "General extended-reacting porous materials modeling with the time-discontinuous Galerkin method for room acoustic simulations," in *Proc. of the 24th Int. Congress on Acoustics*, Gyeongju, Korea, 2022.

[18] Jing Zhao, Zhifei Chen, Ming Bao, Hyojin Lee, and Shinichi Sakamoto, "Two-dimensional finite-difference time-domain analysis of sound propagation in rigid-frame porous material based on equivalent fluid model," *Applied Acoustics*, vol. 146, pp. 204–212, 2019.

[19] Antoni Alomar, Didier Dragna, and Marie-Annick Galland, "Time-domain simulations of sound propagation in a flow duct with extended-reacting liners," *Journal of Sound and Vibration*, vol. 507, pp. 116137, 2021.

[20] Konrad Kowalczyk and Maarten Van Walstijn, "Room acoustics simulation using 3-D compact explicit FDTD schemes," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 34–46, 2010.

[21] Brian Hamilton and Stefan Bilbao, "On finite difference schemes for the 3-D wave equation using non-Cartesian grids," in *Proc. Sound and Music Computing Conf. (SMC)*, Stockholm, Sweden, 01 2013, pp. 592–599.

[22] Brian Hamilton and Craig J Webb, "Room acoustics modelling using GPU-accelerated finite difference and finite volume methods on a face-centered cubic grid," in *Proc. Digital Audio Effects (DAFx)*, Maynooth, Ireland, 2013, pp. 336–343.

[23] James Kates and Eugene Brandewie, "Adding air absorption to simulated room acoustic models," *J. Acoustical Society of America*, vol. 148, pp. EL408–EL413, 11 2020.

[24] Brian Hamilton, "Air absorption filtering method based on approximate Green's function for Stokes' equation," in *Proc. Digital Audio Effects (DAFx)*, 2021, pp. 160–167.

[25] Kane Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Trans. on Antennas and Propagation*, vol. 14, no. 3, pp. 302–307, 1966.

[26] John C Strikwerda, *Finite difference schemes and partial differential equations*, SIAM, Philadelphia, PA, second edition, 2004.

[27] Jonathan Botts and Lauri Savioja, "Integrating finite difference schemes for scalar and vector wave equations," in *Proc. IEEE ICASSP*. IEEE, 2013, pp. 171–175.

[28] M.E. Potter, M. Lamoureux, and M.D. Nauta, "An FDTD scheme on a face-centered-cubic (FCC) grid for the solution of the wave equation," *J. Computational Physics*, vol. 230, no. 15, pp. 6169–6183, 2011.

[29] Julie Meyer, Tapio Lokki, and Jens Ahrens, "Perceptual detection thresholds for numerical dispersion in binaural auralizations of two acoustically different rooms," *J. Acoustical Society of America*, vol. 152, pp. 2266–2276, 10 2022.

[30] Jean-F. Allard and Yvan Champoux, "New empirical equations for sound propagation in rigid frame fibrous materials," *J. Acoustical Society of America*, vol. 91, no. 6, pp. 3346–3353, 1992.

[31] Didier Dragna, Pierre Pineau, and Philippe Blanc-Benon, "A generalized recursive convolution method for time-domain propagation in porous media," *J. Acoustical Society of America*, vol. 138, no. 2, pp. 1030–1042, 2015.

[32] Antoni Alomar, Didier Dragna, and Marie-Annick Galland, "Pole-based identification method to extract the equivalent fluid characteristics of general sound-absorbing materials," *Applied Acoustics*, vol. 174, 12 2020.

[33] Bjørn Gustavsen and A. Semlyen, "Rational approximation of frequency-domain responses by vector fitting," *IEEE Trans. on Power Delivery*, vol. 14, pp. 1052 – 1061, 08 1999.

[34] Brian Hamilton, "Finite volume perspectives on finite difference schemes and boundary formulations for wave simulation," in *Proc. Digital Audio Effects (DAFx)*, 2014, pp. 295–302.

[35] Brian Hamilton, "PFFDTD software," 2021, https://github.com/bsxfun/pffdtd.

[36] Ken Museth, "VDB high-resolution sparse volumes with dynamic topology," *ACM Trans. on Graphics*, vol. 32, pp. 27:1–27:22, 07 2013.

REAL-TIME MODAL SYNTHESIS OF NONLINEARLY INTERCONNECTED NETWORKS

Michele Ducceschi

Department of Industrial Engineering
University of Bologna
Bologna, Italy
michele.ducceschi@unibo.it

Stefan Bilbao

Acoustics and Audio Group
University of Edinburgh
Edinburgh, United Kingdom
sbilbao@ed.ac.uk

Craig J. Webb

Physical Audio Ltd.
London, United Kingdom
craig@physicalaudio.co.uk

ABSTRACT

Modal methods are a long-established approach to physical modeling sound synthesis. Projecting the equation of motion of a linear, time-invariant system onto a basis of eigenfunctions yields a set of independent forced, lossy oscillators, which may be simulated efficiently and accurately by means of standard time-stepping methods. Extensions of modal techniques to nonlinear problems are possible, though often requiring the solution of densely coupled nonlinear time-dependent equations. Here, an application of recent results in numerical simulation design is employed, in which the nonlinear energy is first quadratised via a convenient auxiliary variable. The resulting equations may be updated in time explicitly, thus avoiding the need for expensive iterative solvers, dense linear system solutions, or matrix inversions. The case of a network of interconnected distributed elements is detailed, along with a real-time implementation as an audio plugin.

1. INTRODUCTION

Modal methods are a well-known approach to physical modeling synthesis. In this framework, akin to spectral-like techniques [1], a distributed linear, time-invariant system is described as a superposition of spatial eigenfunctions called the "modes" of the system [2]. These may be computed offline, along with a set of modal weights depending on static parameters describing the input and output (I/O) locations for the system. The resulting equations take the form of a set of independent forced, damped oscillators. Setting appropriate natural frequencies and frequency-dependent damping ratios is simple in this framework. This structure lends itself naturally to fast time-stepping implementations [3], which can be made virtually dispersion-free by employing exact integrators [4, 5]. These desirable numerical properties and the relative ease of implementation made modal methods attractive in early approaches to physical modeling sound synthesis, when frameworks such as Mosaic and Modalys emerged [6, 7]. Further simulation methods, such as the Functional Transformation Method, share some common features with modal synthesis [8, 9], though the equations are here updated in time via inverse Laplace transforms.

In spite of many desirable properties, modal methods possess some limitations. Most notably, the modes can be obtained analytically only in a few cases of interest in musical acoustics (though the eigenvalue problem may still be solved numerically, using e.g. finite differences or finite elements [10]). Furthermore, efficiency quickly deteriorates when the I/O's need to be rendered dynamically. Finally, modal methods yield a rather cumbersome structure of densely nonlinearly coupled equations when applied to nonlinear systems [11], the topic of this work. With regard to this latter issue, exceptions exist, such as the Kirchhoff-Carrier string model [4] (Chapter 8), or the case of strings colliding against a barrier with a linear restoring force [12]. Perceptually-motivated assumptions may reduce the computational burden of nonlinear modal systems, such as the simplified nonlinear model adopted in [13] for the simulation of piano strings. More efficient direct numerical methods, primarily finite differences, are often preferred in such cases[4].

This work presents a fast implementation of modal techniques in the case of nonlinearly coupled distributed elements. While modularity in physical modelling has long roots (see e.g. [14, 15, 16, 17]), this work presents an efficient, energy-stable algorithm for large, nonlinearly coupled modal systems. To this end, a numerical method analogous to the Scalar Auxiliary Variable (SAV) approach is employed [18]. Recently, this method was applied to Hamiltonian systems with non-negative potential energy, yielding computation times on par with simple explicit designs such as the Störmer-Verlet method while conserving energy to machine accuracy and allowing the extraction of sufficient conditions for numerical stability [19]. This is enabled by an appropriate "quadratisation" of the nonlinear potential energy, via an auxiliary state function to be updated independently. The resulting equations of motion may be updated explicitly, through fast linear system solution techniques that exploit matrix structure. The numerical methods described here form the basis for an efficient real-time audio plugin.

The article is structured as follows: Section 2 introduces continuous models of the distributed canonical elements to be used to construct the network. Useful identities for modal projections are given alongside energy considerations. A model for the connections between elements is also detailed. Section 3 introduces the equations for the network, as well as a suitable "quadratised" form of the resulting modal equations, in state-space form. An energy-stable, efficient time discretisation of the same system is described in detail in Section 4, followed by several numerical examples. Finally, a discussion of the architecture of a real-time plugin, written in C++ and suitable for use in commercial digital audio workstations is detailed in Section 5.

2. MODELS

In this section, the basic components of a modular network are described: these are the resonators, including bars and plates, and connections.

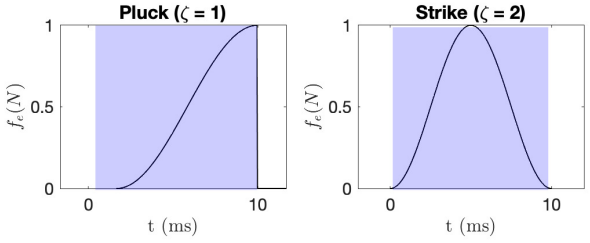


Figure 1: *Excitation functions of the form given in* (4)*. Here,* $T_e = 10$ *ms,* $f_{e,max} = 1$ *N.*

2.1. Resonators

The resonating elements considered here are α-dimensional distributed objects, with $\alpha \in \{1, 2\}$ (that is, $\alpha = 1$ for a string or bar and $\alpha = 2$ for a membrane or plate). A model for such distributed elements is given as [4]:

$$\mathcal{G}u = \delta_e f_e(t), \quad \text{where} \quad \mathcal{G} := \rho\,\partial_t^2 - \mathcal{L} + \rho\sigma\,\partial_t. \tag{1}$$

Here, $u(\mathbf{x}, t) : \mathcal{V} \times \mathbb{R}_0^+ \to \mathbb{R}$ is the displacement of the element, in m, measured transversely from the rest position. u is a function of coordinates $\mathbf{x} = [x_1, \ldots, x_\alpha] \in \mathcal{V} \subset \mathbb{R}^\alpha$ as well as time $t \geq 0$. For simplicity, it is assumed that $\mathcal{V}$ represents an α-dimensional rectangular domain such that $0 \leq x_\kappa \leq L_\kappa$, with $1 \leq \kappa \leq \alpha$. The dynamics of the element are encapsulated by the differential operator $\mathcal{G}$, where ∂_t represents partial differentiation with respect to time t. ρ is the α-dimensional density, in kg·m$^{-\alpha}$, and $\sigma \geq 0$ is a loss parameter, with units of s^{-1}. Furthermore,

$$\mathcal{L} = T\Delta - D\Delta^2, \tag{2}$$

where Δ is the α-dimensional Laplace operator. Here, T, in N·m$^{1-\alpha}$ is a tension constant, and D, in N·m$^{3-\alpha}$ is a rigidity constant. Boundary conditions of simply-supported type are employed:

$$u = \partial_n^2 u = 0, \quad \text{for} \quad t \geq 0, \mathbf{x} \in \mathcal{B}. \tag{3}$$

Here, $\mathcal{B}$ is the boundary of $\mathcal{V}$, and ∂_n represents partial spatial differentiation in a direction perpendicular to the boundary. Finally, f_e represents the excitation function, given here as a raised (or half-raised) squared sine distribution, modeling either a strike or pluck:

$$f_e(t) = f_{e,max} \sin^2\left(\frac{\zeta\pi\,(t - t_e)}{2T_e}\right) \tag{4}$$

for $t_e \leq t \leq t_e + T_e$, and is 0 otherwise. Here t_e in s is the starting time of the excitation, T_e in s is the duration, and $f_{e,max}$ is the maximum force, in N, and where $\zeta = 1$ for a pluck and $\zeta = 2$ for a strike, see Figure 1. Other forms for f_e will be considered subsequently here, including pure sinusoids or sawtooth waves.

The force is assumed to act over a distribution δ_e, here idealized as an α-dimensional Dirac delta function acting at the excitation location $\mathbf{x} = \mathbf{x}_e$:

$$\delta_e := \delta(\mathbf{x} - \mathbf{x}_e). \tag{5}$$

2.1.1. Energy balance of the isolated resonator

Under this choice of boundary conditions, model (1) satisfies the following energy balance:

$$\frac{d(\int_\mathcal{V} \mathcal{H}\,\mathrm{d}\mathbf{x})}{dt} = \underbrace{-\rho\sigma \int_\mathcal{V} (\partial_t u)^2 \mathrm{d}\mathbf{x}}_{\mathcal{Q}} + \underbrace{\partial_t u(\mathbf{x}_e, t) f_e(t)}_{\mathcal{P}}, \tag{6}$$

where the energy density $\mathcal{H}$ is given as [3]:

$$\mathcal{H} = \frac{\rho}{2}(\partial_t u)^2 + \frac{T}{2}|\nabla u|^2 + \frac{D}{2}(\Delta u)^2. \tag{7}$$

Under unforced conditions, the injected power $\mathcal{P}$ is zero and the system is strictly dissipative (since $\mathcal{Q} > 0$), leading to boundedness of the solutions.

2.1.2. Modal expansion and identities

Consider now a modal expansion for the displacement, where each mode is represented by a product of one spatial and one time component:

$$u(\mathbf{x}, t) = \boldsymbol{\chi}^\intercal(\mathbf{x})\,\mathbf{q}(t). \tag{8}$$

Here, $\boldsymbol{\chi}$, $\mathbf{q}$ are column vectors of length M. This is a finite integer, to be specified in terms of the stability requirements of the associated time-stepping scheme, as will be shown below. Let the modal index be $m \in [1, ..., M]$. With the mth mode, there is associated a set of integers $\{\mu_1^m \ldots, \mu_\alpha^m\}$ (the modal numbers). Since simply-supported boundary conditions are assumed, one has

$$\chi_m(\mathbf{x}) := \prod\nolimits_{\kappa=1}^{\alpha} \sqrt{\frac{2}{L_\kappa}} \sin\frac{\mu_\kappa^m \pi x_\kappa}{L_\kappa}, \;\; \mu_\kappa^m \in \mathbb{N}. \tag{9}$$

From (9), it immediately follows that

$$\int_\mathcal{V} \boldsymbol{\chi}\boldsymbol{\chi}^\intercal\,\mathrm{d}\mathbf{x} = \mathbf{I}, \quad \int_\mathcal{V} \boldsymbol{\chi}\,\mathcal{L}(\boldsymbol{\chi}^\intercal)\,\mathrm{d}\mathbf{x} = -\rho\boldsymbol{\Omega}^2, \tag{10}$$

where $\mathbf{I}$ is the $M \times M$ identity matrix, and where $\boldsymbol{\Omega}$ is a diagonal $M \times M$ matrix whose diagonal elements $[\boldsymbol{\Omega}]_{mm}$ are the natural radian frequencies ω_m, defined as:

$$\omega_m = \sqrt{\frac{T}{\rho}\sum_{\kappa=1}^{\alpha}\left(\frac{\mu_\kappa^m \pi}{L_\kappa}\right)^2 + \frac{D}{\rho}\left(\sum_{\kappa=1}^{\alpha}\left(\frac{\mu_\kappa^m \pi}{L_\kappa}\right)^2\right)^2}. \tag{11}$$

In the above, the modal indices μ_κ^m are found by sorting the eigenfrequencies, such that $\omega_1 \leq \omega_2 \leq ... \leq \omega_M$. In the one-dimensional case, $\mu_1^m = m$. In the two-dimensional case, the modal indices μ_1^m, μ_2^m associated with each mode m depend on the aspect ratio, see Figure 2 for an example of this.

2.2. Connections

Connections here take the form of nonlinear springs transferring vibrations across the network. These are energy-storing devices, for which the resulting force may be given as the gradient of a potential, as

$$f(\eta) = -\frac{d\phi}{d\eta}, \tag{12}$$

where η is the elongation of the spring. In this work, intermittent contact is permitted, such that

$$\phi(\eta) = \frac{K}{\gamma + 1}\,[|\eta| - \beta]_+^{\gamma+1} + \frac{\epsilon_0}{2} \geq 0. \tag{13}$$

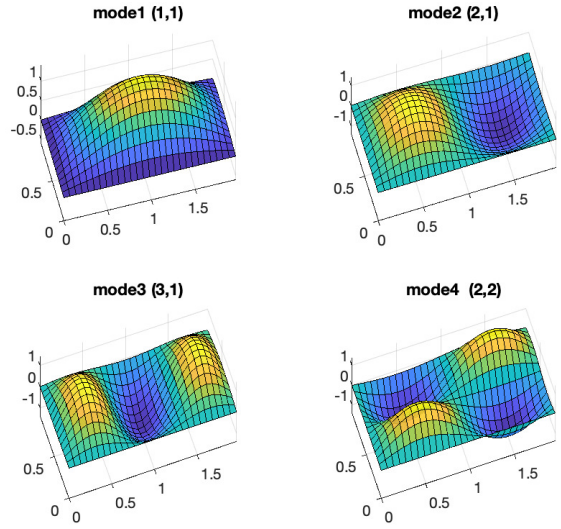


Figure 2: *First four modes of the membrane / plate, with aspect ratio $L_1/L_2 = 2$. The corresponding modal indices μ_κ^m are given in brackets.*

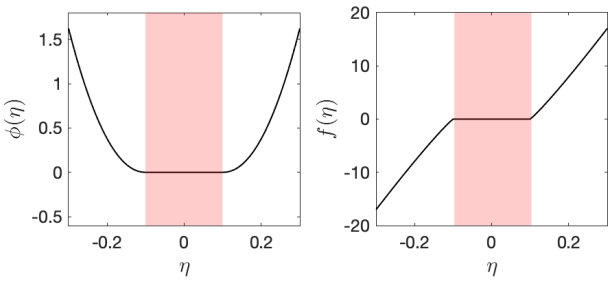


Figure 3: *Example of nonlinear potential* (13) *and corresponding force* (12)*. Here, $K = 10^2$, $\gamma = 1.1$, $\beta = 0.1$, $\epsilon_0 = 0$. The shaded area, whose width is given by 2β, represents a dead zone (no force exerted).*

Here, $K \geq 0$ is a stiffness constant, $\gamma \geq 1$ is a nonlinear exponent and $\beta \geq 0$ is a gap. The gauge constant $\epsilon_0 \geq 0$ is introduced to shift the zero-point of the potential without affecting the force f in (12) (the role of such constant will be briefly discussed below). The potential and the resulting force are represented in Figure 3. Note that no force is exerted by the spring when $|\eta| < \beta$, resulting in a rattling-type force [12, 16]. When $\beta = 0$, linear and cubic springs are recovered by setting $\gamma = \{1, 3\}$, respectively.

3. MODULAR NETWORKS

The resonators described above can now be interconnected in a modular fashion, using groups of connections. Assume N_u such distributed elements, with displacement $u^{(j)}(\mathbf{x}^{(j)}, t)$ and described by the linear operator $\mathcal{G}^{(j)}$, defined over $\mathcal{V}^{(j)}$, $j \in [0, \dots, N_u]$. The excitation of the jth element, of the form (4) is here denoted as $f_e^{(j)}$, and is distributed according to $\delta_e^{(j)}$. Assume furthermore to have a number N_c of connections, indexed by $\nu = 1, ..., N_c$. The νth connection connects element j_ν to j'_ν, for $j_\nu, j'_\nu \in [0, \dots, N_u]$

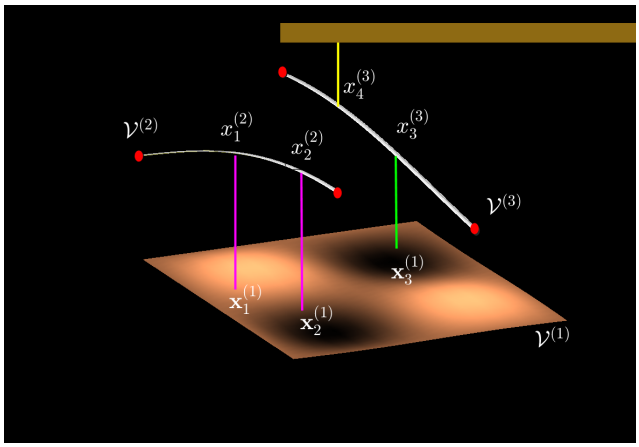


Figure 4: *Example of a network comprising $N_u = 3$ distributed elements (here, one plate and two strings), occupying the domains $\mathcal{V}^{(i)}, i = \{1, 2, 3\}$. The total number of connections is $N_c = 4$, including a connection to a fixed reference frame.*

(the special case in either one of j_ν, j'_ν is zero will be treated shortly). The connection locations are $\mathbf{x}_\nu \in \mathcal{V}^{(j_\nu)}$ and $\mathbf{x}'_\nu \in \mathcal{V}^{(j'_\nu)}$. The force experienced by the distributed elements j_ν, j'_ν due to connection ν is then of the form (12), so that:

$$f^{(\nu)}(\eta^{(\nu)}) = -\frac{d\phi^{(\nu)}}{d\eta^{(\nu)}}, \tag{14}$$

with

$$\eta^{(\nu)} := u^{(j_\nu)}(\mathbf{x}_\nu, t) - u^{(j'_\nu)}(\mathbf{x}'_\nu, t). \tag{15}$$

Furthermore, define the index set:

$$\mathbb{I}_c^{(j)} = \{\nu \in \{1, \dots, N_c\} \,|\, j_\nu = j\} \tag{16a}$$

and let

$$\delta^{(\nu)} := \delta(\mathbf{x}^{(j_\nu)} - \mathbf{x}_\nu), \ \nu \in \mathbb{I}_c^{(j)}. \tag{17}$$

and Given these, the equation of motion of the j^{th} element can be given as

$$\mathcal{G}^{(j)}(u^{(j)}) = \sum_{\nu \in \mathbb{I}_c^{(j)}} \delta^{(\nu)} f^{(\nu)} + \delta_e^{(j)} f_e^{(j)}, \tag{18}$$

Note that, in this framework, the the special case $j'_\nu = 0$ refers to a connection to a fixed reference frame, as seen in Figure 4, for which

$$\eta^{(\nu, j'_\nu = 0)} := u^{(j_\nu)}(\mathbf{x}_\nu, t). \tag{19}$$

It is convenient to define a single scalar potential $\phi = \phi(\boldsymbol{\eta}) : \mathbb{R}^{N_c} \to \mathbb{R}_0^+$, incorporating the potentials from all the connections in the network. This is

$$\phi = \sum_{\nu=1}^{N_c} \phi^{(\nu)}. \tag{20}$$

An energy balance for the network is obtained after multiplying (18) by $\partial_t u^{(j)}$, integrating, and summing all the equations in the network. This yields:

$$\frac{d(\sum_{j=1}^{N_u} \int_{\mathcal{V}^{(j)}} \mathcal{H}^{(j)} \mathrm{d}\mathbf{x}^{(j)} + \phi)}{dt} = \sum_{j=1}^{N_u} \left(-\mathcal{Q}^{(j)} + \mathcal{P}^{(j)}\right), \tag{21}$$

where $\mathcal{Q}^{(j)}$, $\mathcal{P}^{(j)}$ have expressions analogous to those in (6) for the element in isolation. When $\mathcal{P}^{(i)} = 0\ \forall i$, the energy is non-increasing, leading to boundedness of the solutions.

3.1. Energy Quadratisation

Central to the time-stepping scheme presented below is the idea of "quadratisation" of the potential energy. This techinque is often referred to as the Scalar Auxiliary Variable (SAV) method, proposed originally for dissipative phase-field models [18], and later applied to Hamiltonian systems [19]. For that, define

$$\psi = \sqrt{2\phi}. \tag{22}$$

Under such definition, an application of the chain rule allows one to write the forces in terms of ψ, as

$$f^{(\nu)} = -\psi \frac{\partial \psi}{\partial \eta^{(\nu)}} := -\psi\, g^{(\nu)}, \tag{23}$$

and note that ψ, like ϕ is a *scalar* function of $\boldsymbol{\eta} := [\eta^{(1)}, ..., \eta^{(N_c)}]$. When $\phi = \psi^2/2$ is substituted in (21), the energy includes quadratic terms only. Note as well that the auxiliary variable ψ evolves in time according to

$$\dot{\psi} = \mathbf{g}^\intercal\, \dot{\boldsymbol{\eta}}. \tag{24}$$

3.2. Modal Equations

A set of time-dependent modal equations is obtained from (18), after left-multiplying by $\boldsymbol{\chi}^{(j)}$ and integrating over $\mathcal{V}^{(j)}$. By virtue of (10), one gets

$$\Gamma^{(j)}(\mathbf{q}^{(j)}) = \sum_{\nu \in \mathbb{I}_c^{(j)}} \mathbf{Y}^{(\nu)} g^{(\nu)} \psi + \boldsymbol{\chi}^{(j)}(\mathbf{x}_e^{(j)}) f_e^{(j)}, \tag{25}$$

where:

$$\Gamma^{(j)}(\mathbf{q}^{(j)}) = \rho^{(j)} \left(\ddot{\mathbf{q}}^{(j)} + \mathbf{C}^{(j)} \dot{\mathbf{q}}^{(j)} + (\boldsymbol{\Omega}^{(j)})^2 \mathbf{q}^{(j)} \right). \tag{26}$$

In the above, $\mathbf{q}^{(j)}$ is the vector of modal coordinates of the jth distributed element, of length $M^{(j)}$. Furthermore:

$$\mathbf{Y}^{(\nu)} := \int_{\mathcal{V}^{(j_\nu)}} \boldsymbol{\chi}^{(j_\nu)} \delta^{(\nu)}\, \mathrm{d}\mathbf{x}^{(j_\nu)},\ \nu \in \mathbb{I}_c^{(j)}. \tag{27}$$

Above, the matrix $\mathbf{C}$ is a positive diagonal matrix whose diagonal elements include the modal damping coefficients, thus generalising the simple (i.e. frequency independent) loss profile given in (1). The modal equations of all the N_u distributed elements can now be consolidated into a single system:

$$\mathbf{M} \left(\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \boldsymbol{\Omega}^2 \mathbf{q} \right) = -\mathbf{Y}\mathbf{g}\, \psi + \mathbf{f}_e(t), \tag{28}$$

where $\mathbf{M}$ is a fully diagonal mass matrix including the $\rho^{(j)}$'s; $\mathbf{C}$ and $\boldsymbol{\Omega}$ are fully diagonal matrices including the losses and radian frequencies, sorted accordingly. The vector $\mathbf{f}_e$ is obtained by concatenating $\boldsymbol{\chi}^{(j)}(\mathbf{x}_e^{(j)}) f_e^{(j)}$, $j = 1, ..., N_u$. Finally, $\mathbf{Y}$ is a (generally dense) $M \times N_c$ matrix, where $M = \sum_{j=1}^{N_u} M^{(j)}$. Note as well that, in modal form, (24) becomes

$$\dot{\psi} = (\mathbf{Y}\mathbf{g})^\intercal\, \dot{\mathbf{q}}. \tag{29}$$

3.2.1. State-space form

In view of the numerical application presented below, it is worth recasting (28) and (29) in state-space form. This is:

$$\dot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{p}, \tag{30a}$$

$$\dot{\mathbf{p}} = -\mathbf{C}\mathbf{p} - \mathbf{K}\mathbf{q} - \mathbf{Y}\mathbf{g}\, \psi + \mathbf{f}_e(t), \tag{30b}$$

$$\dot{\psi} = (\mathbf{Y}\mathbf{g})^\intercal \mathbf{M}^{-1}\mathbf{p}, \tag{30c}$$

where the stiffness matrix was conveniently defined as $\mathbf{K} := \mathbf{M}\,\boldsymbol{\Omega}^2$. In this configuration, to a vector of inputs $\mathbf{f}_e(t)$ corresponds a vector $\mathbf{y}^{(j)}(t)$ of $N_o^{(j)}$ outputs of the j^{th} element, defined as

$$y_i^{(j)}(t) = (\boldsymbol{\chi}^{(j)}(\mathbf{x}_i^{(j)}))^\intercal \mathbf{q}^{(j)}(t). \tag{31}$$

System (30) is linear in the state variables $\mathbf{q}, \mathbf{p}, \psi$. The nonlinearities are now expressed solely via the vector $\mathbf{g}$. Such linearity is the key feature of the underlying numerical scheme, in that if $\mathbf{g}$ is assumed known at all times, then the system may be solved by a simple matrix inversion. Since here $\mathbf{M}$ is diagonal, the scheme overall becomes explicit, as will be seen shortly.

An energy balance for the modal system may be obtained immediately, after left-multiplying (30b) by $\left(\mathbf{M}^{-1}\,\mathbf{p}\right)^\intercal$. Applying simple identities, one gets:

$$\frac{d}{dt}\left(\frac{1}{2}\mathbf{p}^\intercal \mathbf{M}^{-\intercal}\mathbf{p} + \frac{1}{2}\mathbf{q}^\intercal \mathbf{K}\mathbf{q} + \frac{\psi^2}{2} \right) = -\mathcal{Q} + \mathcal{P}, \tag{32}$$

where $\mathcal{Q} = \mathbf{p}^\intercal \mathbf{M}^{-\intercal}\mathbf{C}\mathbf{p}$, $\mathcal{P} = \mathbf{p}^\intercal \mathbf{M}^{-\intercal}\mathbf{f}_e$. One may of course obtain the same results by carrying out the integrals directly in (21), when the integration over all the $\mathcal{V}^{(i)}$'s is performed using the eigenfunctions (9), and after quadratisation of the nonlinear energy as per (22).

4. DISCRETE-TIME EQUATIONS

A discretisation of system (30) follows as a direct application of the time-stepping procedure detailed in [19]. This method allows us to update the discrete equations in time explicitly, while guaranteeing a passive form of the discrete energy balance. Time is now discretised according to a sample rate $f_s := 1/k$, where k is the sampling period, in s. Then, $n \in \mathbb{N}$ defines the time index, such that solutions are evaluated at the time $t_n := kn$. An interleaved time grid will also be employed, at half-integer steps, as $t_{n+1/2} := k(n + 1/2)$. Time difference operators are introduced as follows, for a time series $\mathbf{u}^n$ defined on the integer time grid:

$$\mathfrak{d}_+ \mathbf{u}^n = \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{k}. \tag{33}$$

An analogous definition holds for an interleaved time series $\mathbf{v}^{n-1/2}$, such that $\mathfrak{d}_+ \mathbf{v}^{n-1/2} = (\mathbf{v}^{n+1/2} - \mathbf{v}^{n-1/2})/k$. An averaging operator for an interleaved time series is also given as

$$\mathfrak{m}_+ \mathbf{v}^{n-\frac{1}{2}} = \frac{\mathbf{v}^{n+\frac{1}{2}} + \mathbf{v}^{n-\frac{1}{2}}}{2}. \tag{34}$$

Given these definitions, a discretisation of (30) is given as:

$$\mathfrak{d}_+ \mathbf{q}^n = \mathbf{M}^{-1}\mathbf{p}^{n+\frac{1}{2}}, \tag{35a}$$

$$\mathfrak{d}_+ \mathbf{p}^{n-\frac{1}{2}} = -\mathbf{C}\,\mathfrak{m}_+ \mathbf{p}^{n-\frac{1}{2}} - \mathbf{K}\mathbf{q}^n - \mathbf{Y}\mathbf{g}^n\, \mathfrak{m}_+ \psi^{n-\frac{1}{2}} + \mathbf{f}_e^n, \tag{35b}$$

$$\mathfrak{d}_+ \psi^{n-\frac{1}{2}} = (\mathbf{Y}\mathbf{g}^n)^\intercal\, \mathbf{M}^{-1} \mathfrak{m}_+ \mathbf{p}^{n-\frac{1}{2}}. \tag{35c}$$

This discretisation is energy-passive, in that left-multiplying (35c) by $(\mathbf{M}^{-1}\mathfrak{m}_+\mathbf{p}^{n-\frac{1}{2}})^\intercal$ leads to [19]:

$$\mathfrak{d}_+\left(\frac{1}{2}(\mathbf{p}^{n-\frac{1}{2}})^\intercal\mathbf{M}^{-\intercal}\mathbf{p}^{n-\frac{1}{2}}+\frac{1}{2}\mathbf{q}^{n-1}\mathbf{K}\mathbf{q}^{n}+\frac{(\psi^{n-\frac{1}{2}})^2}{2}\right)= \\ -(\mathfrak{m}_+\mathbf{p}^{n-\frac{1}{2}})^\intercal\mathbf{M}^{-\intercal}\mathbf{C}\,\mathfrak{m}_+\mathbf{p}^{n-\frac{1}{2}}+(\mathfrak{m}_+\mathbf{p}^{n-\frac{1}{2}})^\intercal\mathbf{M}^{-\intercal}\mathbf{f}_e^n, \tag{36}$$

which discretises (32). While the discrete energy comprises quadratic terms only, numerical stability is guaranteed only under non-negativity of the energy overall. Using (35a) to express the momenta $\mathbf{p}$ in terms of $\mathbf{q}$ in the expression for the energy above, one may derive the following bound on the time step [19]:

$$k<\frac{2}{\lambda_{\max}(\mathbf{M}^{-\frac{1}{2}}\mathbf{K}\mathbf{M}^{-\frac{\intercal}{2}})}=\frac{2}{\omega_M}, \tag{37}$$

where $\lambda_{\max}$ denotes the largest eigenvalue, and ω_M is the largest eigenfrequency across all the N_u distributed elements, taking the form (11). This condition arises solely as the particular discretisation adopted here for the linear part, and it guarantees non-negativity of the discrete energy in (36).

The key feature of scheme (35) is the explicit form of $\mathbf{g}$, defined here as:

$$\mathbf{g}^n:=\nabla_{\boldsymbol{\eta}}\psi\Big|_{t=t_n}=\frac{1}{\sqrt{2\phi}}\nabla_{\boldsymbol{\eta}}\phi\Big|_{t=t_n}, \tag{38}$$

where ϕ is as per (20). Note that the division by $\sqrt{2\phi}$ may be ill-defined when the gauge constant $\epsilon_0=0$ in (13). Thus, it may be useful to shift the potential upward. It is known that the gauge constant has an influence on the convergence properties of the quadratised schemes [20], though a thorough discussion on the role of such constant is out-of-scope here. Using (35a) and (35c) in (35b), one arrives at the following:

$$\mathbf{A}^n\mathbf{q}^{n+1}=\mathbf{d}\left(\mathbf{q}^n,\mathbf{q}^{n-1},\mathbf{g}^n,\psi^{n-\frac{1}{2}}\right),\quad\text{with} \tag{39}$$
$$\mathbf{A}^n:=\mathbf{I}+\mathbf{a}\mathbf{b}^\intercal,\ \mathbf{a}:=\frac{k}{2}\mathbf{M}^{-1}\mathbf{Y}\mathbf{g}^n,\ \mathbf{b}:=\frac{k}{2}\mathbf{Y}\mathbf{g}^n.$$

Above, the vector $\mathbf{d}$ contains values of the state vector from previous time steps. As noted above, the update is in the form of a linear system (a rank-1 perturbation of the identity matrix). However, the inversion of $\mathbf{A}$ may be performed in $\mathcal{O}(M)$ operations, using the Sherman-Morrison formula [21, 19], thus yielding a very efficient algorithm, as shown below.

4.1. Numerical Examples

Numerical experiments are given in Figures 5 and 6. There, scheme (35) is compared to the simple Störmer-Verlet method [22], applied directly to the second-order-in-time system (28). This is an explicit method which, in the linear case, has the same stability condition as (37). However, stability is not guaranteed under nonlinear conditions, and unpredictable numerical behaviour ensues for sufficiently large stiffness values, as seen in Figure 6.

In spite of this, the compute times for method (35) are very much on par with the Störmer-Verlet method, since these are both explicit schemes. Remarkably, the computational bottleneck here

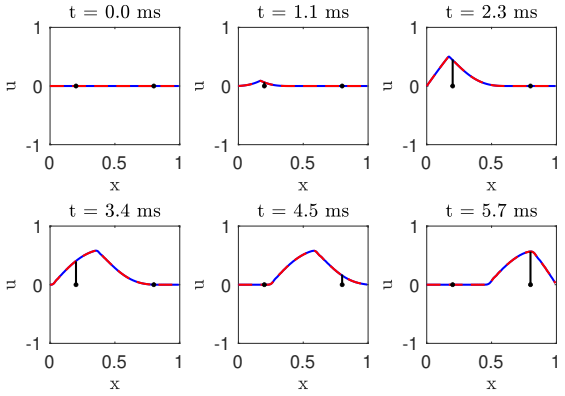


Figure 5: *Time evolution of a string under the influence of $N_s=2$ nonlinear springs (represented as black lines). Solid blue: scheme (35); dashed red: Störmer-Verlet. Here, the string has a fundamental frequency of 100 Hz, a mass of 1 g, and presents a total $M=128$ modes between the fundamental and the limit of stability. The string is activated using a strike with $T_e=5$ ms, $f_{e,max}=200$ N. The springs are located at $0.2L$, $0.8L$, and have $K=7.3\cdot10^{-4}$, $\gamma=1.1$, $\beta=0$. For this simulation, $\epsilon_0=10^9$.*

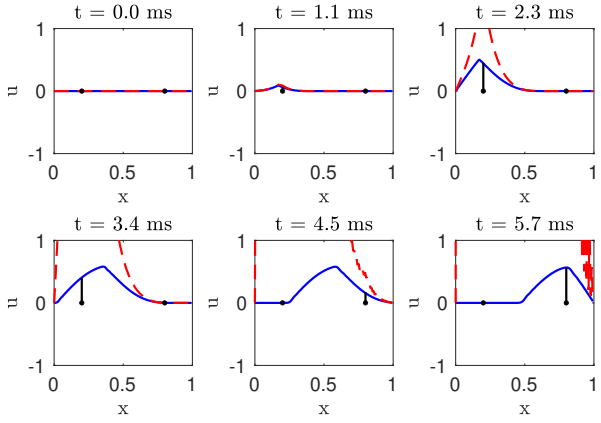


Figure 6: *Same as Figure 5, but here $K=730$.*

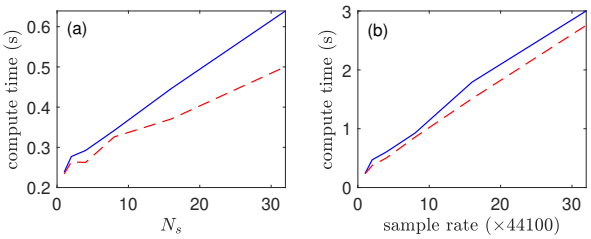


Figure 7: *Matlab compute times for the string of Figure 5. (a): $f_s = 44100$, N_s as indicated. (b): $N_s = 1$, f_s as indicated. Solid blue: scheme* (35)*; dashed red: Störmer-Verlet.*

is not in the the structure of the scheme itself, but rather in the algebraic operations involving the (dense) product $\mathbf{Y}\mathbf{g}^n$, as can be seen from Figure 7. In the figure, it is seen that the slope of the two schemes is identical (and the compute times are very similar) when the number of connections is fixed, and the sample rate is changed. However, when the sample rate is fixed, and the number of connections is varied, scheme (35) presents a somewhat steeper slope than Störmer-Verlet. In practice, though, only a handful of such nonlinear connections is needed for expressive sound synthesis, as will be shown in the examples below.

5. REAL-TIME IMPLEMENTATION

There are a number of considerations that have to be addressed in order to create a real-time instrument in the form of an audio plug-in. Computational expense, polyphonic behaviour and parameter control are all key aspects in developing a usable system. Three different configurations of models were tested: connected strings (SS), strings connected to a plate (SP), and two connected plates (PP), as shown in Figures 8 - 10. In the SS set-up the excitation pluck is applied to string 1 (that is $f_e^{(j)}$ in (18) is identically zero for $j > 1$), with the remaining strings acting as resonators. These can be tuned to different fundamental frequencies as required, with either semitone or fine tuning offsets (this is a trivial procedure in the modal framework). In the SP configuration both strings are plucked and the plate acts as a resonating device. In the final set-up PP an excitation signal is applied to the top plate with the second plate acting as the resonator. This excitation can be in the form of a raised cosine as described above, or indeed some arbitrary signal such as a sawtooth or other waveform.

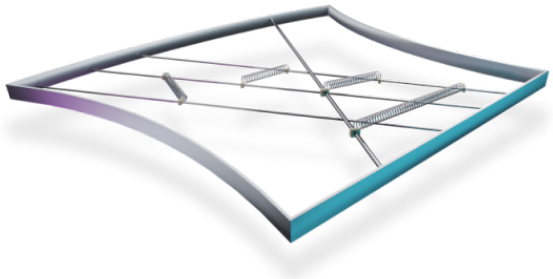


Figure 8: *System of four connected strings.*

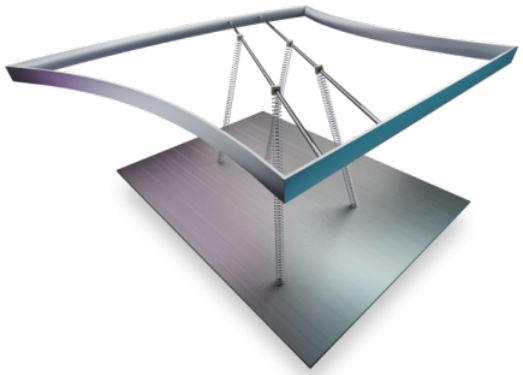


Figure 9: *System of two strings connected to a plate.*

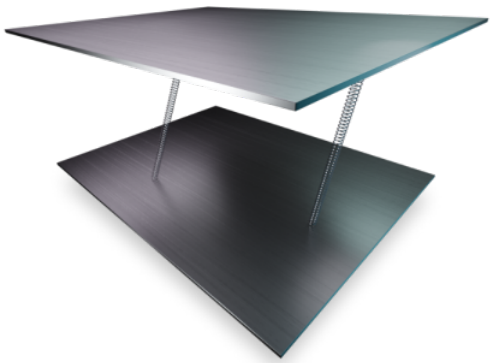


Figure 10: *System of two connected plates.*

Combining all 3 configurations in a single unified instrument provides a suitably wide palette of timbres, from harmonic string tones and pads, percussive strikes and evolving soundscapes. An example of the plugin capabilities is given in the companion page[1]. Each of the configurations requires a different approach to polyphonic behaviour, as described in Section 5.2 below.

5.1. Computational Performance

The core elements of the system which have to be computed at each time-step are mainly simple operations over one-dimensional arrays such as cumulative sum reductions, and also dense matrix-by-vector multiplication ($\mathbf{Y}\mathbf{g}^n$ in (35)). The compute performance varies linearly according to the number of modes used in the system and the number of connecting elements. Ensuring peak performance in C++ requires all aspects of the computation to be fully vectorized. Whilst the majority of the operations can be vectorized by the compiler with a suitable optimisation level (i.e. -Ofast in Clang), some elements do require manual application of vector intrinsics. These are either SSE/AVX intrinsics for Intel builds or NEON intrinsics for native use on Apple Silicon machines. Note

[1] `https://mdphys.org/DAFx23.html`

that double-precision was used throughout due to the high level of nonlinearity in the system.

Initial testing was performed off-line to gauge general CPU performance. The strings model was tested with all 4 strings tuned to either C0 (which used 800 modes) or C3 (which used 184 modes), and with 4 connecting elements. The string-plate model was tested using 1,100 modes and 4 connections, and finally the dual-plate model using 1,300 modes and 2 connections. Test machines were a Mac Pro with a Xeon E5 processor, a MacBook Pro with a Core i7, and Mac mini with Apple Silicon M1 processor. Table 1 shows the resulting computation times.

Table 1: *Computation times for optimised C++ simulations over 44100 time-steps.*

Configuration	Xeon E5	Core i7	Apple M1
Strings C0	0.20s	0.18s	0.11s
Strings C3	0.06s	0.05s	0.04s
Strings-Plate	0.27s	0.26s	0.15s
Plates	0.29s	0.26s	0.17s

These timings are all well within the bounds required for use in a real-time environment, even considering the lower overall clock frequencies which are obtained when running a plug-in inside a Digital Audio Workstation.

5.2. Polyphonic Behaviour

Having tested the models in their basic off-line states, the next stage is to decide how to use them to create a playable instrument. The CPU-usage of the SS model decreases significantly as the fundamental frequency increases. Only the bottom 2 octaves use the full 200 modes, and higher registers require much less. Therefore using a standard voice-based approach is viable in this case, and allows 5 to 6 note polyphony (holding one voice as being ready for reset). At a Note-On event a damped voice is recalculated to the given fundamental and tuning offsets of the connected strings, along with the other parameters that define the state of the string. A re-triggering system is also employed, so that a string may be plucked multiple times without having to set up a new voice.

The SP model, however, requires a very different mechanism. Due to the number of modes used in the plate the system will always use over 1,000 modes. Applying a voice-based system would very quickly consume an entire CPU core, even with just 2 or 3 note polyphony. Instead, a hybrid mechanism was used consisting of a single instance of the SP. In order to achieve multiple octaves of playable notes the strings are disconnected from the plate and retuned on-the-fly. So at a Note-On event the required state of a string is calculated whilst it is disconnected from the system, and it is then reconnected whilst applying the excitation pluck. At the same time the opposing string is damped and disconnected ready for the next Note-On, in a monophonic mode, or left to continue sounding in duo-phonic mode.

Finally the behaviour of the PP configuration depends on the type of excitation. Here again a single instance of the 2 plates is used. Note-On events can trigger strikes, or multiple strikes, for percussive sounds. Separately, a choice of sawtooth or sine wave signals can be used as the forcing excitation, thus allowing multiple octaves of polyphonic signals at very little computational cost using a voice-based setup. Traditional elements such as VCF and distortion units can be employed at this point in the signal chain to give a more flexible soundscape.

5.3. Parameter Control and Modulation

Parameter control of physical modelling systems is always a complicated process. In an optimal case all parameters would continuously affect the sound as the engine is being computed. However, this is often made difficult due to parameters that change the setup of the system state, and for example the mass of a plate or the stiffness of a string. During prototyping 3 different types of parameter control were used; real-time continuous, Note-On, and system-reset.

The core principle of these models is the ability to connect strings and plates together to form systems of sympathetic resonance. The parameters for the connecting elements, such as their strength and rattle gap, can be directly manipulated in real-time. The only requirement is some level of smoothing of the parameter movement to avoid unwanted noise.

Parameters that define the state of a string, such as stiffness (i.e., harmonicity), are defined as Note-On controls. Their value is picked-up from the control at a new Note-On event and the string state is calculated appropriately. This system works well for both the connected strings configuration where it fits naturally into the voice-based mechanism, and for the string-plate where we are disconnecting and reconnecting strings on-the-fly.

A further aspect of parameter control is modulation. One essential example is being able to perform pitchbend and vibrato on the strings. In a modal system this is straightforward as one has access to the modal frequencies and their weights directly in the sound engine. By computing a vector of frequencies at pitchbend up, and another at pitchbend down, a simple linear interpolation can be used to obtain the correct frequency during run-time. Further modulation effects can be obtained by varying the modal weights over time.

Figure 11: *Prototype of a connected strings model running as a VST3 plug-in.*

Parameters that control the state of a plate are more complicated as the objects are acting as continuous resonators in both configurations. One possible approach would be to store multiple tables of modal frequencies and weights for various control settings and interpolate between them. This approach has been used in our previous work on plate reverberation [3] to allow the plate

size to be dynamically adjusted without resetting the plate and disrupting the audio signal. A simpler method is to perform a fast damp and reset of the system when these parameters change, and then re-exciting the model with the new settings. This was used in prototyping due the number of plate controls which are available.

6. CONCLUDING REMARKS

This work has illustrated an application of newly devised schemes for the fast simulation of mechanical systems with non-negative potential energy. The schemes were applied to the modal equations of a nonlinearly coupled network of distributed elements, forming the basis of an advanced physical modelling synthesizer. The proposed schemes yield compute times close to those of simpler, though numerically unstable designs such as Störmer-Verlet. A working real-time plugin, suitable for use in most current digital audio workstations, has also been illustrated. It has been shown that this fast mathematical model allows the simulation of thousands of nonlinearly coupled modes, while keeping the CPU usage low enough for real-time performance. The wide sonic palette of the synthesizer, in conjunction with the flexibility of the modal approach, makes the plugin an attractive choice for musicians and sound designers alike.

7. ACKNOWLEDGMENTS

Michele Ducceschi received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 950084-NEMUS.

8. REFERENCES

[1] John P Boyd, *Chebyshev and Fourier spectral methods*, Dover, Mineola, New York, USA, 2001.

[2] L. Meirovitch, *Fundamentals of vibrations*, Waveland Press, Long Grove, Illinois, USA, 2010.

[3] M. Ducceschi and C.J. Webb, "Plate reverberation: Towards the development of a real-time plug-in for the working musician," in *Proc of Int Conf Acoust (ICA 2016)*, Buenos Aires, Argentina, September 2016.

[4] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, Wiley, Chichester, UK, 2009.

[5] J. Cieslinski, "On the exact discretization of the classical harmonic oscillator equation," *J Differ Equ Appl*, vol. 17, no. 11, pp. 1673–1694, 2009.

[6] J.M. Adrien, "The missing link: Modal synthesis," in *Representations of Musical Signals*, p. 269–298. MIT Press, Cambridge, MA, USA, 1991.

[7] G. Eckel, "Sound synthesis by physical modelling with modalys," *Proc. ISMA'95*, pp. 478–482, 1995.

[8] R. Rabenstein and L. Trautmann, "Digital sound synthesis of string instruments with the functional transformation method," *Signal Proc*, vol. 83, no. 8, pp. 1673–1688, 2003.

[9] S. Schlecht, J. Parker, M. Schäfer, and R. Rabenstein, "Physical modeling using recurrent neural networks with fast convolutional layers," in *Proc Digital Audio Effects (DAFx-22)*, 2022, pp. 138–145.

[10] J. McQuillan and M. van Walstijn, "Modal spring reverb based on discretisation of the thin helical spring model," in *Proc Digital Audio Effects (DAFx-21)*, 2021, pp. 191–198.

[11] M. Ducceschi and C. Touzé, "Modal approach for nonlinear vibrations of damped impacted plates: application to sound synthesis of gongs and cymbals," *J Sound Vib*, vol. 334, pp. 313–331, 2015.

[12] M. van Walstijn, J. Bridges, and S. Mehes, "A real-time synthesis oriented tanpura model," in *Proc Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, September 2016.

[13] B. Bank and L. Sujbert, "Generation of longitudinal vibrations in piano strings: From physics to sound synthesis," *J Acoust Soc Am*, vol. 117, no. 4, pp. 2268–2278, 2005.

[14] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms," *Comp. Music J.*, vol. 8, no. 3, pp. 60–73, 1983.

[15] M. Van Walstijn and M. Sandor, "An explorative string-bridge-plate model with tunable parameters," in *Proc Digital Audio Effects (DAFx-17)*, Edinburgh, UK, September 2017.

[16] S. Bilbao, M. Ducceschi, and C. Webb, "Large-scale real-time modular physical modeling sound synthesis," in *Proc Digital Audio Effects (DAFx-19)*, Birmingham, UK, September 2019.

[17] S. Willemsen, *The Emulated Ensemble: Real-Time Simulation of Musical Instruments using Finite-Difference Time-Domain Methods*, Ph.D. thesis, Aalborg University, Denmark, 2021.

[18] J. Shen, J. Xu, and J. Yang, "The scalar auxiliary variable (sav) approach for gradient flows," *J Comput Phys*, vol. 353, pp. 407–416, 2018.

[19] S. Bilbao, M. Ducceschi, and F. Zama, "Explicit exactly energy-conserving methods for hamiltonian systems," *J Comput Phys*, vol. 472, pp. 111697, 2023.

[20] Z. Liu and X. Li, "The exponential scalar auxiliary variable (e-sav) approach for phase field models and its explicit computing," *SIAM J Sci Comput*, vol. 42, no. 3, pp. B630–B655, 2020.

[21] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Ann Math Stat*, vol. 21, pp. 124–127, 1950.

[22] E. Hairer, C. Lubich, and G. Wanner, "Geometric numerical integration illustrated by the Störmer–Verlet method," *Acta Numerica*, vol. 12, pp. 399–450, 2003.

TUNABLE COLLISIONS: HAMMER-STRING SIMULATION WITH TIME-VARIANT PARAMETERS

Maarten van Walstijn and Abhiram Bhanuprakash

SARC
Queen's University Belfast
Belfast, UK
{m.vanwalstijn|abhanuprakash01}@qub.ac.uk

Vasileios Chatziioannou

Department of Music Acoustics (IWK)
University of Music and Performing Arts
Vienna, Austria
chatziioannou@mdw.ac.at

ABSTRACT

In physical modelling synthesis, articulation and tuning are effected via time-variation in one or more parameters. Adopting hammered strings as a test case, this paper develops extended forms of such control, proposing a numerical formulation that affords on-line adjustment of each of its scaled-form parameters, including those featuring in the one-sided power law for modelling hammer-string collisions. Starting from a modally-expanded representation of the string, an explicit scheme is constructed based on quadratising the contact energy. Compared to the case of time-invariant contact parameters, updating the scheme's state variables relies on the evaluation of two additional analytic partial derivatives of the auxiliary variable. A numerical energy balance is derived and the numerical contact force is shown to be strictly non-adhesive. Example results with time-variant tension and time-variant contact stiffness are detailed, and real-time viability is demonstrated.

1. INTRODUCTION

The manipulation of variables is intrinsic to musical instrument performance. For example, to produce a specific sound with a violin the musician controls the speed, normal force, and angle of the bow as well as the position of fingers that press the string to the fingerboard. In hammered string instruments, which is the target of the current study, such articulation is normally restricted to the acceleration of keys that drive the hammer motion and the adjustment of tension during the tuning process, although in certain instrument families, such as dulcimers, the striking position on the string can also be varied.

In musical instrument modelling, articulation and tuning are accomplished through variation over time of the relevant physical parameters. This is exploited in physics-based synthesis for the exploration of the sound of acoustic instruments (of both existing and modular design) across their parameter spaces [1]. Recently, specific attention has been given to on-line tuning of parameters that are normally considered to remain constant [2, 3]. To contribute towards facilitating such extended synthesis control, this paper sets out to numerically model the interaction between a hammer and a stiff string under time-variance of a non-redundant set of model parameters. For the resulting algorithm to be of practical use, it should be computationally efficient, numerically stable, and free of audible artefacts. In addition, the response to driving forces and parameter manipulations should ideally be similar to what can be expected in that regard from the underlying physical laws, and parameter time-variance should not necessarily lead to large amplitude swings in the chosen output signal.

Solutions to various similar and related problems can be found across the literature. Most notably, simulation of string vibrations under time-varying tension (or an equivalent string length adjustment) has been reported using digital waveguides [4], mass-spring-damper systems [5], finite-difference methods [3], and modal synthesis [6]. The challenge increases when nonlinearities are introduced, perhaps most tellingly so when one-sided forces are involved. For example, models in which a finger or other object can be dynamically brought in contact with a string while also its position along the string axis can be varied over time (e.g. [7, 8]) typically rely on an iterative solver to update the state variables, which severely reduces the scope for parallelisation and real-time implementation [9]. Similarly, in [2] all 29 parameters of a modal-form string-bridge-plate model with nonlinear spring connections were made tunable, but the use of an iterative solver meant that for real-time audio rendering the parameter space and the rate of change in parameters had to be empirically constrained to avoid instability issues and artefacts. In [10] this issue was side-stepped by casting the update equations in analytic form, but so far this has been made to work only for a small subset of cases in which a unity contact power law exponent applies.

Originating separately in Port-Hamiltonian form [11], the recent emergence of energy quadratisation approaches, including the Invariant Energy Quadratisation (IEQ) method [12] and the Scalar Auxiliary Variable (SAV) method [13], has paved the way for numerical simulation of nonlinear musical instruments vibrations without the use of iterative solvers [14, 15], with specific scheme variants introduced for modelling collisions [16, 17]. The current paper extends energy quadratisation to modelling lumped conservative nonlinearities under parameter time-variance, taking hammer-string interaction as a case study. For completeness, additional innovations that improve the handling of tension time-variance in modal-form algorithms are introduced.

The paper is structured as follows. The hammer-string system equations are outlined in Section 2, including a scaled form that is modally expanded. The discretisation of the resulting equations is presented in Section 3, with the update equations provided and also featuring analyses of the key properties of the resulting algorithm. The proposed formulation is then explored, exemplified, and tested in Section 4 via a number of numerical experiments, followed by concluding remarks in Section 5. Sound examples are available on the accompanying github page[1].

[1] https://github.com/mvanwalstijn/Tunable-Collisions

2. HAMMER-STRING MODEL

In the following, $\partial_t y$ and $\partial_x y$ denote the partial derivatives with respect to time t and string axis position x, respectively, and the total time derivative is written as $d_t y$. Considering a string of length L, mass density ρ, cross section A, Young's modulus E, moment of inertia I interacting with a hammer of mass m_h striking from above, the equations governing the transversal string displacement $u = u(x,t)$ and hammer position $u_\mathrm{h} = u_\mathrm{h}(t)$ can be written as [18]:

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u + \theta(x) F_\mathrm{c}, \tag{1}$$

$$d_t \big\{ m_\mathrm{h} d_t u_\mathrm{h} \big\} = F_\mathrm{e} - F_\mathrm{c}. \tag{2}$$

The driving of the hammer is represented here with the excitation force $F_\mathrm{e} = F_\mathrm{e}(t)$, the specific form of which depends on the type of instrument. For example, a simplified form of modelling N_p successive piano hammer strikes at time instants $t = \tau_l$ is:

$$F_\mathrm{e}(t) = \sum_{l=1}^{N_\mathrm{p}} \underbrace{m_\mathrm{h} V_{\mathrm{e},l}}_{p_{e,l}} \delta(t - \tau_l) + F_{\mathrm{stop},l}(t), \tag{3}$$

where $V_{\mathrm{e},l} < 0$ is an externally supplied hammer velocity parameter. In the absence of gravity, we emulate the hammer coming to rest at $u_\mathrm{h} = u_\mathrm{max}$ after bouncing back from the string, which is represented in (3) by the forces $F_{\mathrm{stop},l}(t) < 0$; in practice, one may achieve this by simply capping the hammer displacement at $u_\mathrm{h} = u_\mathrm{max}$. Modelling the driving of the hammer in dulcimers or clavichords would require different formulations of $F_\mathrm{e}(t)$.

The contact force in (1,2) is assumed to be non-hysteretic under parameter constancy, and is defined with a power-law:

$$F_\mathrm{c} = -\kappa \left[y \right]_+^\alpha \leq 0, \tag{4}$$

where $[y(t)]_+ = \max(0, u_\mathrm{s}(t) - u_\mathrm{h}(t))$ is the effective inter-object compression. For simplicity, the contact force is applied at a single point, using $\theta(x) = \delta(x - x_\mathrm{h})$. Correspondingly, the displacement of the string as 'seen' by the hammer is $u_\mathrm{s}(t) = u(x_\mathrm{h}, t)$. Simply supported boundary conditions are assumed:

$$u(0,t) = \partial_x^2 u(0,t) = 0, \quad u(L,t) = \partial_x^2 u(L,t) = 0, \tag{5}$$

and initial conditions are set as

$$u_\mathrm{h}(0) = u_\mathrm{max}, \quad d_t u_\mathrm{h}(0) = 0, \quad u(x,0) = 0, \quad \partial_t u(0,x) = 0. \tag{6}$$

As in previous studies (e.g. [10]), the force at the end of the string is chosen as an appropriate output variable:

$$F_\mathrm{s}(t) = -T \partial_x u(L,t) + EI \partial_x^3 u(L,t). \tag{7}$$

2.1. Scaled Form

The parameters ρ, A, and L are considered to remain constant over time. To obtain a form of the system equations with fewer parameters, the following non-dimensional variables are introduced:

$$\bar{x} = \frac{x}{L}, \qquad \bar{u} = \frac{u}{L}, \qquad \bar{u}_\mathrm{h} = \frac{u_\mathrm{h}}{L}. \tag{8}$$

Then, after substitution, the system can be written as

$$\partial_t^2 \bar{u} = \bar{T} \left[\partial_{\bar{x}}^2 \bar{u} - \pi^{-2} \mathcal{B} \partial_{\bar{x}}^4 \bar{u} \right] + \bar{\theta}(\bar{x}) \bar{F}_\mathrm{c}, \tag{9}$$

$$d_t \big\{ \bar{m}_\mathrm{h} d_t \bar{u}_\mathrm{h} \big\} = \bar{F}_\mathrm{e} - \bar{F}_\mathrm{c}, \tag{10}$$

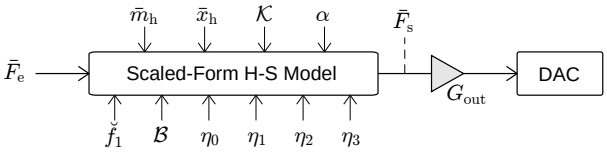


Figure 1: *Schematic diagram of the scaled-form hammer-string model.*

where

$$\bar{F}_\mathrm{c} = \bar{\kappa} \big[\overbrace{\bar{u}_s - \bar{u}_\mathrm{h}}^{\bar{y}} \big]_+^\alpha, \quad \bar{F}_\mathrm{e} = \sum_l^{N_\mathrm{p}} \overbrace{\frac{p_{\mathrm{e},l}}{\rho A L^2}}^{\bar{p}_{e,l}} \delta(t - \tau_l), \tag{11}$$

$$\bar{m}_\mathrm{h} = \frac{m_\mathrm{h}}{\rho A L}, \quad \bar{T} = \frac{T}{\rho A L^2}, \quad \mathcal{B} = \frac{EI\pi^2}{TL^2}, \tag{12}$$

$$\bar{\kappa} = \frac{\kappa L^{\alpha-2}}{\rho A}, \quad \bar{x}_\mathrm{h} = \frac{x_\mathrm{h}}{L}, \quad \bar{\theta}(\bar{x}) = \delta(\bar{x} - \bar{x}_\mathrm{h}). \tag{13}$$

The new parameter $\mathcal{B}$ is the inharmonicity factor [19], $\bar{m}_\mathrm{h}$ is the hammer/string mass ratio, and $\bar{u}_\mathrm{s}(t) = \bar{u}(\bar{x}_\mathrm{h}, t)$. Analogously, we may define

$$\bar{F}_\mathrm{s}(t) = \frac{F_\mathrm{s}}{\rho A L^2} = \bar{T} \left[\pi^{-2} \mathcal{B} \partial_{\bar{x}}^3 - \partial_{\bar{x}} \right] \bar{u}(1,t). \tag{14}$$

2.2. Energy Quadratisation

The scaled-form contact force in (11) can be expressed as:

$$\bar{F}_\mathrm{c} = -\partial_{\bar{y}} \bar{\Phi}, \qquad \bar{\Phi}(\bar{y}) = \frac{\bar{\kappa}}{\alpha + 1} \left[\bar{y} \right]_+^{\alpha+1}, \tag{15}$$

where the actual contact potential Φ (in Joules) relates to its scaled-form counterpart as $\Phi = \rho A L^3 \bar{\Phi}$. Taking a split-potential energy quadratisation approach [15], the scaled-form contact potential is written in quadratic form with $\bar{\Phi} = \frac{1}{2}\psi^2$. Making use of the chain rule, this allows writing the contact force as

$$\bar{F}_\mathrm{c} = -\psi \overbrace{\partial_{\bar{y}} \psi}^{g_{\bar{y}}} = -\psi \frac{d_t \psi - d_t \bar{\kappa} \overbrace{\partial_{\bar{\kappa}} \psi}^{g_{\bar{\kappa}}} - d_t \alpha \overbrace{\partial_\alpha \psi}^{g_\alpha}}{d_t \bar{y}}. \tag{16}$$

Defining the auxiliary variable ψ as the positive square root of $2\bar{\Phi}$, the gradient variables $g_{\bar{y}}$, g_α, and $g_{\bar{\kappa}}$ can be expressed directly as functions of $\bar{y}$, α, and $\bar{\kappa}$ as follows:

$$g_{\bar{y}} = \sqrt{\tfrac{1}{2}\bar{\kappa}(\alpha+1) \left[\bar{y} \right]_+^{\alpha-1}}, \quad g_{\bar{\kappa}} = \sqrt{\frac{\left[y \right]_+^{\alpha+1}}{2\bar{\kappa}\,(\alpha+1)}} \tag{17}$$

$$g_\alpha = \sqrt{\frac{\frac{1}{2}\bar{\kappa} \left[\bar{y} \right]_+^{\alpha+1}}{\alpha+1}} \left[\log\left(\left[\bar{y} \right]_+ + \varepsilon \right) - \frac{1}{\alpha+1} \right], \tag{18}$$

where a positive constant of the size of the machine epsilon has been included within the log term in (18) for handling the case where $[\bar{y}]_+$ approaches zero. Following similar principles as applied in IEQ and SAV methods, the numerical scheme will be constructed in explicit form by directly discretising the equations contained within (16) and making use of the analytic expressions in (17,18) to evaluate the gradient variables. A novel aspect is the emergence of the additional gradient terms g_α and $g_{\bar{\kappa}}$ due to time variance in the power law parameters. It is worth noting that if ψ is defined as $-\sqrt{2\bar{\Phi}}$, for consistency the terms $g_{\bar{y}}$, $g_{\bar{\kappa}}$ and g_α would also have to have a minus sign in front of the square root symbol.

2.3. Parameter Control

Figure 1 shows the inputs and outputs of the scaled-form model. The output gain G_{out} is needed to scale $\bar{F}_{\text{s}}$ to the $[-1, 1]$ input range of the digital-to-analog converter. Each of the ten control parameters in Figure 1 is considered to be adjustable on the fly, and as such is treated as time-dependent. Four of those parameters ($\mathcal{B}$, $\bar{m}_{\text{h}}$, $\bar{x}_{\text{h}}$, and α) readily appear in the scaled-form model equations (9-13). The parameters $\eta_{0,1,2,3}$ are damping coefficients that will be introduced in Section 2.4. This subsection explains how the remaining two parameters ($\breve{f}_1$, $\mathcal{K}$) are related to the scaled-form model parameters. The tension parameter ($\bar{T}$) can be calculated directly from the string's fundamental frequency $\breve{f}_1$ (in Hz) in the absence of stiffness as $\bar{T} = 4\breve{f}_1^2$. To enable independent control of the effective stiffness (through $\mathcal{K}$) and the 'contact nonlinearity' (through α), $\bar{\kappa}$ has been re-parameterised as follows:

$$\bar{\kappa} = (\alpha + 1)\mathcal{K}\left(\frac{\mathcal{K}}{\bar{\Phi}_{\text{r}}}\right)^{\alpha}. \tag{19}$$

where $\bar{\Phi}_{\text{r}}$ denotes a (constant-over-time) scaled-form reference potential that represents the amount of contact energy that can approximately be expected[2]. This is exemplified in Figure 2. Example values for the control parameters, which were transcribed from [20], are listed in Table 1. Where needed, the string constants ρA and L are used for un-scaling displacements, forces, or energies, but they do not otherwise feature within the scaled-form model that forms the basis for numerical simulation.

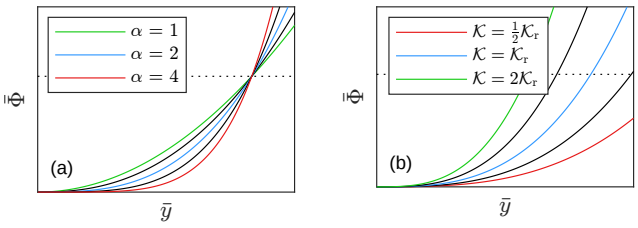


Figure 2: *Contact potential curves for (a): a range of α values with $\mathcal{K} = \mathcal{K}_{\text{r}}$ and (b): a range of $\mathcal{K}$ values and $\alpha = 2$. In both subfigures, the horizontal dashed line indicates the scaled-form reference potential ($\bar{\Phi}_{\text{r}}$).*

2.4. Modal Expansion

For the boundary conditions in (5), we may expand the string displacement as

$$\bar{u}(\bar{x}, t) = \sum_{i=1}^{M} \underbrace{\sqrt{2}\sin(i\pi\bar{x}(t))}_{v_i(\bar{x}(t))} \widetilde{u}_i(t) = [\mathbf{v}(\bar{x})]^{\text{T}}\mathbf{u}, \tag{20}$$

where $v_i(\bar{x}(t))$ and $\widetilde{u}_i(t)$ are the modal shape function and the modal displacement for the ith mode, and $\mathbf{v}(\bar{x})$ and $\mathbf{u}$ are the respective column vector representations (with time dependence dropped in the notation). Substituting (20) into (9), multiplying with the basis functions, and spatially integrating from $\bar{x} = 0$ to $\bar{x} = 1$ then leads to a set of coupled ordinary differential equations which may be expressed in vector form as

$$d_t^2\mathbf{u} + \mathbf{R}d_t\mathbf{u} + \mathbf{K}\mathbf{u} = \mathbf{h}\bar{F}_{\text{c}}, \tag{21}$$

where $\mathbf{h} = \mathbf{v}(\bar{x}_{\text{h}})$ and $\mathbf{K}$ is an $M \times M$ diagonal matrix with the non-zero elements

$$K_{i,i} = i^2\pi^2\bar{T}\left(1 + \mathcal{B}i^2\right). \tag{22}$$

Initally, since there is no string damping in (9), the $M \times M$ damping matrix $\mathbf{R}$ contains only zeros. String damping can be introduced in polynomial form by setting the diagonal elements of $\mathbf{R}$ to:

$$R_{i,i} = 2\left(\eta_0 + \eta_1 i\pi + \eta_2 i^2\pi^2 + \eta_3 i^3\pi^3\right). \tag{23}$$

Of particular relevance is the case where $\eta_2 = 0$, which can be shown (see [2]) to align well with the experimentally validated damping formulation by Woodhouse [21]. The ith mode frequency for free vibration (i.e. $\widetilde{F}_i = 0$) then is

$$\omega_i = \sqrt{K_{i,i} - \tfrac{1}{4}R_{i,i}^2}, \tag{24}$$

which takes on an imaginary value in case of overdamping. In modal form, the string displacement at the contact point can be written as $u_{\text{s}} = \mathbf{h}^{\text{T}}\mathbf{u}$, and the string force in (14) becomes

$$\bar{F}_{\text{s}}(t) = -\sum_{i=1}^{M} \underbrace{i\pi\bar{T}(t)\left[1 + \mathcal{B}(t)i^2\right](-1)^i}_{w_i(t)} \widetilde{u}_i(t) = -\mathbf{w}^{\text{T}}\mathbf{u}. \tag{25}$$

The direct dependence of w_i on $\bar{T}$ can lead to large swings in the output amplitude when time-varying the string tension. The need for output gain adjustments can be significantly reduced by replacing $w_i(t)$ with the adjusted output weights $\dot{w}_i(t) = \sqrt{\frac{\bar{T}(0)}{\bar{T}(t)}}w_i(t)$.

2.5. Energy Balance and Conserved Quantities

An energy balance equation can be obtained by pre-multiplying (21) with $(d_t\mathbf{u})^{\text{T}}$, multipling (10) with $d_t\bar{u}_{\text{h}}$, and adding the resulting equations, yielding

$$\begin{aligned} d_t\bar{H} = {} & \tfrac{1}{2}\mathbf{u}^{\text{T}}(d_t\mathbf{K})\mathbf{u} - (d_t\mathbf{u})^{\text{T}}\mathbf{R}\,d_t\mathbf{u} - (d_t\mathbf{h})^{\text{T}}\mathbf{u}\bar{F}_{\text{c}} + d_t\bar{u}_{\text{h}}\bar{F}_{\text{e}} \\ & - (d_t\bar{u}_{\text{h}})^2 d_t\bar{m}_{\text{h}} + \psi\left(g_{\bar{\kappa}}d_t\bar{\kappa} + g_\alpha d_t\alpha\right), \end{aligned} \tag{26}$$

in which the scaled-form Hamiltonian $\bar{H}$ takes the form :

$$\bar{H} = \frac{(d_t\mathbf{u})^{\text{T}}d_t\mathbf{u}}{2} + \frac{\mathbf{u}^{\text{T}}\mathbf{K}\mathbf{u}}{2} + \frac{\bar{m}_{\text{h}}(d_t\bar{u}_{\text{h}})^2}{2} + \frac{\psi^2}{2} \geq 0. \tag{27}$$

Table 1: *Scaled-form model parameter values (transcribed from [20]).*

		piano C2	piano C4	piano C7
$\breve{f}_1$	[Hz]	65.4	262	2093
$\mathcal{B}$		7.4×10^{-5}	3.77×10^{-4}	8.6×10^{-3}
η_0	[s^{-1}]	0.5	0.5	0.5
η_1	[s^{-1}]	0.01	0.01	0.1
η_2	[s^{-1}]	0.0	0.0	0.0
η_3	[s^{-1}]	10^{-6}	10^{-6}	10^{-4}
$\bar{m}_{\text{h}}$		0.14	0.75	4.71
$\mathcal{K}$	[s^{-2}]	335	2560	4.3×10^4
α		2.3	2.5	3.0
$\bar{x}_{\text{h}}$		0.12	0.12	0.0625
ρA	[kg m^{-1}]	18.4×10^{-3}	6.3×10^{-3}	5.2×10^{-3}
L	[m]	1.90	0.62	0.09

[2]The $\mathcal{K}$ values in Table 1 have been transcribed using $\bar{\Phi}_{\text{r}} \approx 0.75/L^2$, which is a representative value of the average piano key, derived from setting the unscaled version Φ_{r} equal to the kinetic energy of a hammer with $\bar{m}_{\text{h}} = 0.75$ and $d_t u_{\text{h}} = -1.41$ m/s.

From (10) and (26), it is immediately clear that the scaled-form hammer momentum $\bar{p}_\mathrm{h} = \bar{m}_\mathrm{h} d_t \bar{u}_\mathrm{h}$ and system Hamiltonian $\bar{H}$ are conserved under specific conditions:

$$d_t \bar{p}_\mathrm{h} = 0 \text{ if } \big(\bar{F}_\mathrm{e}, \bar{F}_\mathrm{c} = 0\big), \tag{28}$$

$$d_t \bar{H} = 0 \text{ if } \big(\bar{F}_\mathrm{e}, \eta_0, \eta_1, \eta_2, \eta_3 = 0, \partial_t \bar{T}, \partial_t \mathcal{B}, \partial_t \bar{m}_\mathrm{h}, \partial_t \bar{\kappa}, \partial_t \alpha, \partial_t \bar{x}_\mathrm{h} = 0\big). \tag{29}$$

The numerical scheme will be constructed such that the discrete counterparts of $\bar{H}$ and $\bar{p}_\mathrm{h}$ are conserved under the same conditions.

3. NUMERICAL FORMULATION

3.1. Difference and Averaging Operators

The numerical model will evaluate variables and parameters at discrete-time instants $t_n = n\Delta_t$. The usual form u^n is employed to denote the approximation to u at time $t = n\Delta_t$. The following shift operators are defined:

$$\epsilon_{t+} u^n = u^{n+\frac{1}{2}}, \qquad \epsilon_{t-} u^n = u^{n-\frac{1}{2}}. \tag{30}$$

Elemental temporal difference and averaging operators can then be constructed as

$$\delta_t = \frac{\epsilon_{t+} - \epsilon_{t-}}{\Delta_t}, \quad \mu_t = \frac{\epsilon_{t+} + \epsilon_{t-}}{2}, \quad \delta_{t\cdot} = \frac{\epsilon_{t+}^2 - \epsilon_{t-}^2}{2\Delta_t}, \tag{31}$$

$$\delta_{t+} = \frac{\epsilon_{t+}^2 - 1}{\Delta_t}, \quad \delta_{t-} = \frac{1 - \epsilon_{t-}^2}{\Delta_t} \quad \mu_{t\cdot} = \frac{\epsilon_{t+}^2 + \epsilon_{t-}^2}{2}. \tag{32}$$

where we can identify several equivalences (e.g. $\delta_{t\cdot} = \delta_t \mu_t$). Finite-difference approximations are achieved by either combining or directly applying these elemental operators, e.g.

$$\delta_t^2 u^n = \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta_t^2} = \partial_t^2 u(n\Delta_t) + O(\Delta_t^2), \tag{33}$$

$$\mu_t^2 u^n = \frac{u^{n+1} + 2u^n + u^{n-1}}{4} = u(n\Delta_t) + O(\Delta_t^2), \tag{34}$$

$$\delta_{t\cdot} u^n = \frac{u^{n+1} - u^{n-1}}{2\Delta_t} = \partial_t u(n\Delta_t) + O(\Delta_t^2). \tag{35}$$

The following product identities can be derived for arbitrary grid functions u^n, q^n:

$$\delta_t\big\{(\delta_t u^n)^2\big\} = 2\delta_{t\cdot} u^n \delta_t^2 u^n, \quad \delta_t\big\{(\mu_t u^n)^2\big\} = 2\delta_{t\cdot} u^n \mu_t^2 u^n, \tag{36}$$

$$\delta_t\{\mu_t q^n \delta_t u^n\}\, \delta_t u^n = \tfrac{1}{2}\delta_t\big\{\mu_t q^n (\delta_t u^n)^2\big\} + \delta_{t\cdot} q^n \delta_{t+} u^n \delta_{t-} u^n, \tag{37}$$

which is useful for the purposes of numerical energy analysis.

3.2. Discretisation

Let $\mathbf{u}^n$ be a column vector holding the modal displacements $\widetilde{u}_i^n$, $i = 1, 2, \cdots M$. The size of this vector (i.e. M) will not be varied on-line. Taking into account the need to avoid both numerical dispersion and mode aliasing, the mode dynamics in (21) with time-varying parameters can be discretised in vector form as follows:

$$\delta_t^2 \mathbf{u}^n + \widehat{\mathbf{R}}^n \delta_{t\cdot} \mathbf{u}^n + \mu_t^2 \widehat{\mathbf{K}}^n \mu_t^2 \mathbf{u}^n = \bar{F}_\mathrm{c}^n \mu_{t\cdot} \mathbf{h}^n, \tag{38}$$

where the non-zero elements of the adjusted diagonal matrices $\widehat{\mathbf{R}}^n$ and $\widehat{\mathbf{K}}^n$ are

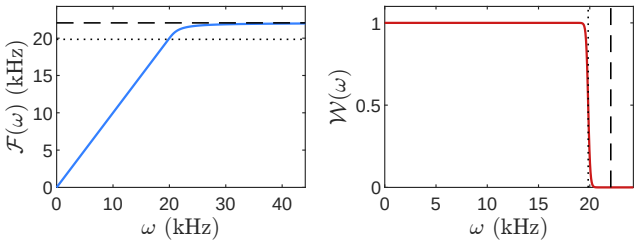


Figure 3: *The mode frequency soft-clipping function $\mathcal{F}(\omega)$ (left) and the suppression weight function $\mathcal{W}(\omega)$ (right) for $\Delta_t = 1/44100$. The dashed line indicates the Nyquist frequency (π/Δ_t), and the dotted line represents the cut-off frequency (ω_a).*

$$\widehat{R}_{i,i}^n = \frac{2b_i^n}{\Delta_t}, \qquad \widehat{K}_{i,i}^n = \frac{4}{\Delta_t^2}\, a_i^n. \tag{39}$$

The real-valued coefficients in (39) are:

$$a_i^n = \frac{1 - \left(\Upsilon_{+,i}^n + \Upsilon_{-,i}^n\right) + \Upsilon_{+,i}^n \Upsilon_{-,i}^n}{1 + \left(\Upsilon_{+,i}^n + \Upsilon_{-,i}^n\right) + \Upsilon_{+,i}^n \Upsilon_{-,i}^n}, \tag{40}$$

$$b_i^n = \frac{2 - 2\Upsilon_{+,i}^n \Upsilon_{-,i}^n}{1 + \left(\Upsilon_{+,i}^n + \Upsilon_{-,i}^n\right) + \Upsilon_{+,i}^n \Upsilon_{-,i}^n}, \tag{41}$$

featuring the complex-conjugate pair

$$\Upsilon_{\pm,i}^n = \exp\left\{\pm \mathrm{j}\mathcal{F}(\omega_i^n)\Delta_t - \tfrac{1}{2} R_{i,i}^n \Delta_t\right\}, \tag{42}$$

where $\mathrm{j} = \sqrt{-1}$ and

$$\mathcal{F}(\omega) = \begin{cases} \omega & : \omega^2 < \omega_\mathrm{a}^2 \\ \zeta \arctan[\zeta^{-1}(\omega - \omega_\mathrm{a})] + \omega_\mathrm{a} & : \text{otherwise} \end{cases} \tag{43}$$

is a function that 'soft-clips' the mode frequencies (see the plot on the left-hand side of Figure 3), as such preventing aliased mode frequencies. Here ω_a is an appropriate 'cut-off frequency' chosen below the Nyquist frequency ($\omega_\mathrm{a} = 0.9\pi/\Delta_t$ is used throughout the paper), and $\zeta = 2/\Delta_t - 2\omega_\mathrm{a}/\pi$. Since the 'out-of-range' modes will have incorrect resonance frequencies, they need to be suppressed in the calculation of both $\bar{u}_\mathrm{s}$ and $\bar{F}_\mathrm{s}$. This can be achieved by calculating the elements of the vectors $\mathbf{h}^n$ and $\mathbf{w}^n$ as $h_i^n = \mathcal{W}(\omega_i^n) \cdot v_i(\bar{x}_\mathrm{h}^n)$ and $w_i^n = \mathcal{W}(\omega_i^n) \cdot w_i(n\Delta_t)$, respectively, where $\mathcal{W}(\omega)$ is a smooth weight function

$$\mathcal{W}(\omega) = \frac{1}{1 + [\mathrm{Re}(\omega)/\omega_\mathrm{a}]^{200}}, \tag{44}$$

which strongly suppresses frequencies larger than ω_a (see the plot on the right-hand side of Figure 3). Using (24), it can be shown that $|\Upsilon_{\pm,i}^n| \leq 1$ for both under- and over-damped modes. It follows from (40,41) that $a_i^n \geq 0$ and $b_i^n \geq 0$, meaning that the diagonal elements of $\widehat{\mathbf{R}}^n$ and $\widehat{\mathbf{K}}^n$ are guaranteed non-negative. For constancy in the mode frequencies ω_i^n, the above discretisation results into a scheme free of numerical dispersion and attenuation, similarly to the modal-form schemes proposed in previous studies [22, 2]. Expanding (38), one may derive the update form

$$\mathbf{u}^{n+1} = \mathbf{u}^{n-1} + \mathbf{C}^n \mu_{t\cdot} \mathbf{h}^n \bar{F}_\mathrm{c}^n - 2\mathbf{z}^n, \tag{45}$$

with

$$\mathbf{z}^n = \tfrac{1}{2}\Big[\mathbf{B}^n \mathbf{u}^{n-1} - \mathbf{A}^n \mathbf{u}^n\Big], \tag{46}$$

where the elements of the diagonal matrices $\mathbf{A}^n$, $\mathbf{B}^n$, and $\mathbf{C}^n$ are:

$$A^n_{i,i} = \frac{2 - 2\mu_t^2 a_i^n}{1 + \mu_t^2 a_i^n + \mu_t^2 b_i^n}, \tag{47}$$

$$B^n_{i,i} = \frac{2 + 2\mu_t^2 a_i^n}{1 + \mu_t^2 a_i^n + \mu_t^2 b_i^n}, \tag{48}$$

$$C^n_{i,i} = \frac{\Delta_t^2}{1 + \mu_t^2 a_i^n + \mu_t^2 b_i^n}. \tag{49}$$

The hammer dynamics in (10) are discretised with:

$$\delta_t \{ \mu_t \bar{m}_\mathrm{h}^n \delta_t \bar{u}_\mathrm{h}^n \} = \bar{F}_\mathrm{e}^n - \bar{F}_\mathrm{c}^n. \tag{50}$$

Setting $\xi_\mathrm{h}^n = \Delta_t^2 / \mu_t \bar{m}_\mathrm{h}^{n+\frac{1}{2}}$ and $\gamma^n = \mu_t \bar{m}_\mathrm{h}^{n-\frac{1}{2}} / \mu_t \bar{m}_\mathrm{h}^{n+\frac{1}{2}}$, this can be written as

$$\bar{u}_\mathrm{h}^{n+1} = \bar{u}_\mathrm{h}^{n-1} - \xi_\mathrm{h}^n \bar{F}_\mathrm{c}^n - 2 z_\mathrm{h}^n, \tag{51}$$

where

$$z_\mathrm{h}^n = \tfrac{1}{2}(1 + \gamma^n)\left(\bar{u}_\mathrm{h}^{n-1} - \bar{u}_\mathrm{h}^n\right) - \tfrac{1}{2}\xi_\mathrm{h}^n \bar{F}_\mathrm{e}^n. \tag{52}$$

By pre-multiplying (45) with $(\mathbf{h}^n)^\mathrm{T}$ and subtracting (50), the following scalar equation is obtained:

$$\underbrace{\bar{y}^{n+1} - \bar{y}^{n-1}}_{s^n} = \underbrace{\left[(\mathbf{h}^n)^\mathrm{T} \mathbf{C}^n \mu_t . \mathbf{h}^n + \xi_\mathrm{h}^n \right]}_{\xi^n} \bar{F}_\mathrm{c}^n - 2 \underbrace{\left[(\mathbf{h}^n)^\mathrm{T} \mathbf{z}^n - z_\mathrm{h}^n \right]}_{z^n}. \tag{53}$$

The contact force $\bar{F}_\mathrm{c}^n$ as written in quadratised form in (16) is discretised with:

$$\bar{F}_\mathrm{c}^n = -(\mu_t \psi^n) \cdot g_{\bar{y}}^n, \quad \frac{\delta_t \psi^n - g_{\bar{\kappa}}^n \delta_t . \bar{\kappa}^n - g_\alpha^n \delta_t . \alpha^n}{\delta_t . \bar{y}^n} = g_{\bar{y}}^n, \tag{54}$$

where we can substitute $2\Delta_t \delta_t . \bar{y}^n = s^n$. From the second equation, a separate update of the auxiliary variable is found as

$$\psi^{n+\frac{1}{2}} = \psi^{n-\frac{1}{2}} + \tfrac{1}{2} g_{\bar{y}}^n s^n + \underbrace{\tfrac{1}{2}\left[g_{\bar{\kappa}}^n (\bar{\kappa}^{n+1} - \bar{\kappa}^{n-1}) + g_\alpha^n (\alpha^{n+1} - \alpha^{n-1}) \right]}_{\chi^n}. \tag{55}$$

Substituting the first equation in (54) into (53) we then can, making use of (55), obtain the explicit solution

$$s^n = -\frac{2 z^n + \xi^n \left(\psi^{n-\frac{1}{2}} + \frac{1}{2}\chi^n \right) g_{\bar{y}}^n}{1 + \frac{1}{4}\xi^n (g_{\bar{y}}^n)^2}, \tag{56}$$

where it is seen that the denominator in (56) is guaranteed positive, hence solution existence is ensured. Under the assumption that the auxiliary variable remains non-negative during simulation, the gradient variables $g_{\bar{\kappa}}^n$ and g_α^n can be calculated directly as per equations (17,18). The remaining gradient variable $g_{\bar{y}}^n$ then has to be constrained such that $\psi^{n+\frac{1}{2}} \geq 0$, which in the time-variant case translates to satisfying the quadratic inequality

$$\tfrac{1}{4}\psi^{n-\frac{1}{2}} \xi^n (g_{\bar{y}}^n)^2 + z^n g_{\bar{y}}^n - \left(\psi^{n-\frac{1}{2}} + \chi^n \right) \leq 0. \tag{57}$$

This leads to the evaluation of $g_{\bar{y}}^n$ in branched form as given in the Appendix. Once s^n has been calculated, the auxiliary variable is updated with (55). The $[.]_+$ operator is subsequently applied to ensure that the value of $\psi^{n+\frac{1}{2}}$ does not become ever so slightly negative due to finite-precision errors. Next, the contact force $\bar{F}_\mathrm{c}^n$ is calculated with (54), after which the state variables $\mathbf{u}$ and u_h can be updated with (45) and (50), respectively. Finally, the output signal is calculated with $\bar{F}_\mathrm{s}^n = (\mathbf{w}^n)^\mathrm{T} \mathbf{u}^n$. Note that the matrices $\hat{\mathbf{K}}^n$ and $\hat{\mathbf{R}}^n$ are not calculated within the algorithm; this is needed only in instances where one wants to track the evolution of the system energy. The update of the modal displacements requires only the elements expressed in (47-49).

3.3. Non-Adhesive Contact Force

For the explicit scheme presented above, guaranteed non-adhesion can be shown starting from the inequality $\psi^{n+\frac{1}{2}} \geq 0$ for all n, from which it follows that $\mu_t \psi^n \geq 0$. From (61) we have that $\dot{g}_{\bar{y}}^n \geq 0$ by definition. Further, given that the numerator of (62) is non-negative, it follows that $\dot{g}_{\bar{y}}^n \geq 0$. Seen together with (60) this means that $g_{\bar{y}}^n \geq 0$, and therefore that $\bar{F}_\mathrm{c}^n = -\mu_t \psi^n \; g_{\bar{y}}^n \leq 0$.

3.4. Energy Balance and Conserved Quantities

From (50) it is immediately clear that the numerical hammer momentum $\bar{p}_\mathrm{h}^{n+\frac{1}{2}} = \mu_t \bar{m}_\mathrm{h}^{n+\frac{1}{2}} \delta_t \bar{u}_\mathrm{h}^{n+\frac{1}{2}}$ is conserved when no forces act upon the hammer. A discrete energy balance can be derived by pre-multiplying (38) with $(\delta_t . \mathbf{u}^n)^\mathrm{T}$, (50) with $\delta_t . u_\mathrm{h}^n$ and adding the resulting equations:

$$\begin{aligned} \delta_t \bar{H}^n = {} & \tfrac{1}{4}(\mu_t \mathbf{u}^{n+\frac{1}{2}})^\mathrm{T} \delta_t . \widehat{\mathbf{K}}^n \mu_t \mathbf{u}^{n+\frac{1}{2}} + \tfrac{1}{4}(\mu_t \mathbf{u}^{n-\frac{1}{2}})^\mathrm{T} \delta_t . \widehat{\mathbf{K}}^n \mu_t \mathbf{u}^{n-\frac{1}{2}} \\ & - (\delta_t . \mathbf{u}^n)^\mathrm{T} \widehat{\mathbf{R}}^n \delta_t . \mathbf{u}^n + \delta_t . \bar{u}_\mathrm{h}^n \bar{F}_\mathrm{e}^n - \delta_t . \bar{m}_\mathrm{h}^n \delta_{t+} \bar{u}_\mathrm{h}^n \delta_{t-} \bar{u}_\mathrm{h}^n \\ & - \bar{F}_\mathrm{c}^n (\delta_t . \mathbf{h}^n)^\mathrm{T} \mu_t . \mathbf{u}^n + \mu_t \psi^n (g_{\bar{\kappa}}^n \delta_t . \bar{\kappa}^n + g_\alpha^n \delta_t . \alpha^n), \end{aligned} \tag{58}$$

$$\begin{aligned} \bar{H}^{n+\frac{1}{2}} = {} & \tfrac{1}{2}(\delta_t \mathbf{u}^{n+\frac{1}{2}})^\mathrm{T} \delta_t \mathbf{u}^{n+\frac{1}{2}} + \tfrac{1}{2}(\mu_t \mathbf{u}^{n+\frac{1}{2}})^\mathrm{T} \mu_t \widehat{\mathbf{K}}^{n+\frac{1}{2}} \mu_t \mathbf{u}^{n+\frac{1}{2}} \\ & + \tfrac{1}{2}\mu_t \bar{m}_\mathrm{h}^{n+\frac{1}{2}} (\delta_t \bar{u}_\mathrm{h}^{n+\frac{1}{2}})^2 + \tfrac{1}{2}(\psi^{n+\frac{1}{2}})^2 \geq 0. \end{aligned} \tag{59}$$

Here we made use of the product identities in (36-37). It follows directly that the numerical energy $\bar{H}^{n+\frac{1}{2}}$ is conserved for constant parameters and no damping or external force. Further, the terms on both sides of (58) are consistent approximations to the corresponding terms in (26), so we can expect the simulation to exhibit energy behaviour under parameter time-variance that approximates that of the underlying continuous-domain model (see Figure 6 for a numerical verification).

3.5. Blockwise Parameter Updates & Linear Interpolation

Under the assumption that the control parameters vary over time relatively slowly, they can be updated every N_b samples, such that the control rate is N_b times lower than the audio sampling rate, in which case the parameter signals are assumed to be bandlimited in the sense of containing no frequency components above $f = 1/(2N_\mathrm{b}\Delta_t)$. Given that one round of updating of the model parameters is computationally more expensive than one time step of updating the state variables, such a blockwise parameter update form yields significant computational savings. To alleviate audible artefacts, the following parameters and coefficients are linearly interpolated over each block, at very low cost: $\bar{m}_\mathrm{h}$, $\bar{\kappa}$, α, h_i, a_i, b_i. The coefficients ξ_h and γ_h as well as the diagonals of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are updated at each time step using the aforementioned linearly-interpolated values.

4. NUMERICAL EXPERIMENTS

4.1. Contact Force Signals with Static Parameters

Before exploring parameter time-variance, we first verify the correctness of the algorithm and its implementation for the case of constant parameters. Figure 4 shows the contact force signal for three hammer striking velocities, using a standard audio rate (i.e. $\Delta_t = 1/44100$ s). The resulting waveforms are similar to those obtained in previous studies (e.g. [20]).

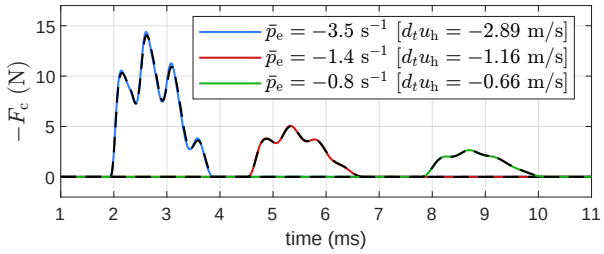


Figure 4: *Contact force signals for a C4 string with static parameters. For each hammer velocity, the black dashed line indicates the exact solution as approximated with 24 times oversampling.*

4.2. String Tuning

Of special interest is the behaviour of the algorithm under time-varying tension, because this involves string modes moving out of and into the normal frequency range of interest. Figure 5 shows the amplitude normalised spectrograms of four simulation output signals. In each simulation, the string was excited with a hard hammer ($\mathcal{K} = 100000$ s^{-2}, $\alpha = 1.2$) at $t = 0.2$ s, and the parameter $\breve{f}_1$ was subsequently increased upwards in a linear fashion by a factor of 1.5, and then linearly decreased back to its original value. The time step is $\Delta_t = 1/(OF \cdot 44100)$ s, where OF is the oversampling factor. The top left plot, obtained with two times oversampling, can be considered as the nominally correct result. For wider comparison we also include the result obtained with the dynamic grid model [3], for which $N_b = 1$. Visible in the top right plot ($N_b = 1$) is the suppression of the out-of-range modes according to the weight function $\mathcal{W}(\omega)$. High-frequency artefacts appear with $N_b = 128$ (bottom left plot), but these are generally more than 60 dB below the level of the partial tones, and as such barely or not audible. A noticeable difference with the FD dynamic grid result is that the mode frequencies evolve in a more regular fashion, and as such the proposed methodology does not rely on frequency-dependent damping and/or oversampling to mask audible artefacts. Secondly, with the modal-form algorithm the modes that drift out of range when the tension increases are 'pulled back' into range once the tension is reduced.

4.3. Energy and Output Amplitude

A similar experiment is conducted here, this time with no string damping, to investigate the behavour of the output amplitude and the numerical energy under time-varying tension. Figure 6(b) shows the output signal as calculated with w_i and $\dot{w}_i$, respectively, confirming that the use of adjusted output weights helps reducing output amplitude swings. Figure 6(c) compares the corresponding numerical energy evolution as calculated with oversampling factors

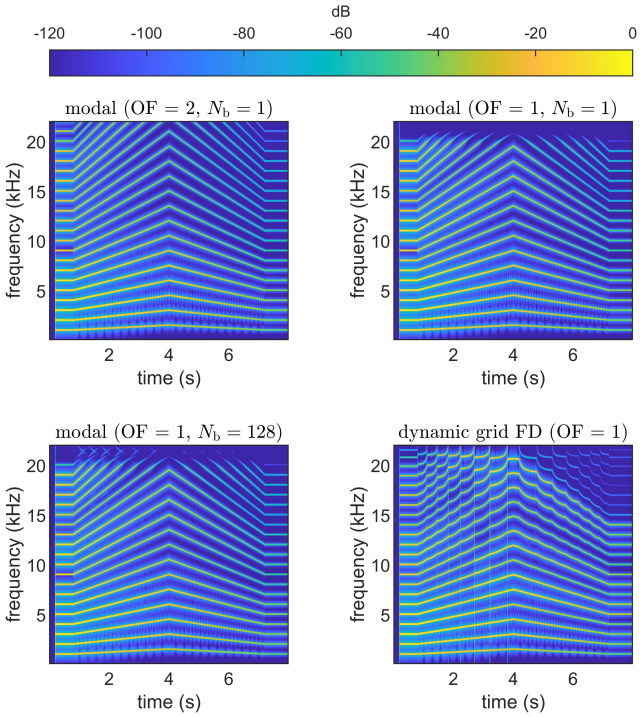


Figure 5: *Spectral evolution of the string force signal under time-varying tension. The model parameters are* $\breve{f}_1 = 1000$ *Hz,* $\mathcal{B} = 2.55 \times 10^{-6}$, $\eta_0 = 0.5\, s^{-1}$, $\eta_1 = 0\, s^{-1}$, $\eta_2 = 0.0001\, s^{-1}$, $\eta_3 = 0\ s^{-1}$, $\bar{m}_h = 0.3$, $\bar{x}_h = 1/23$.

$OF = 1$ and $OF = 24$, in both cases using $M = 60$. The closeness of these two curves demonstrates that the numerical energy balance remains approximately correct in the presence of mode-frequency soft-clipping (which occurs only for $OF = 1$ here) and other approximation errors.

4.4. Time-Varying Contact Parameters

Among less conventional forms of parameter time-variance, the terrain of on-the-fly adjustment of the contact parameters seems particularly uncharted. To investigate how the algorithm handles such time-variance, single hammer-string collisions were simulated in which the effective stiffness parameter $\mathcal{K}$ was set to either rapidly increase (see the left-hand side plots of Figure 7) or rapidly decrease (see the right-hand side plots of Figure 7) during contact. As can be seen in the bottom plots, time-variance in $\mathcal{K}$ leads to hysteresis in the force-vs-compression curve. For increasing $\mathcal{K}$, the hysteresis is 'inverted' (i.e. in the opposite direction to what is normally observed with hysteresis in piano hammers due to loss mechanisms). The main take-away from these results is that fast time-variance in $\mathcal{K}$ – and indeed in α, which was also tested but not shown here – can be simulated without artefacts.

4.5. Computational Efficiency

To asses the viability of real-time implementation, the real-time factor (RTF), which is defined here as a measure of how much real time passes with the computation of one second of audio output, was recorded for a range of N_b and M values. The piano C2 pa-

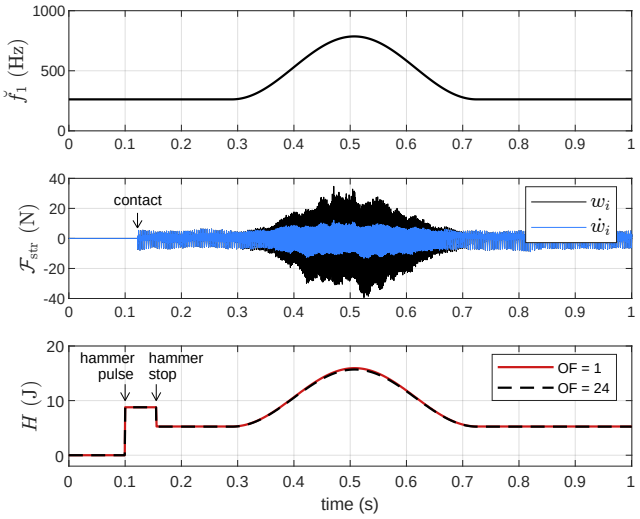


Figure 6: *Simulation with time-varying tension and no damping. Top: fundamental frequency profile. Middle: output waveforms for OF = 1. Bottom: system energy. The parameters are those for a piano C4 string, with* $\Delta_t = 1/(OF \cdot 44100)$ *s and* $N_\mathrm{b} = 32$.

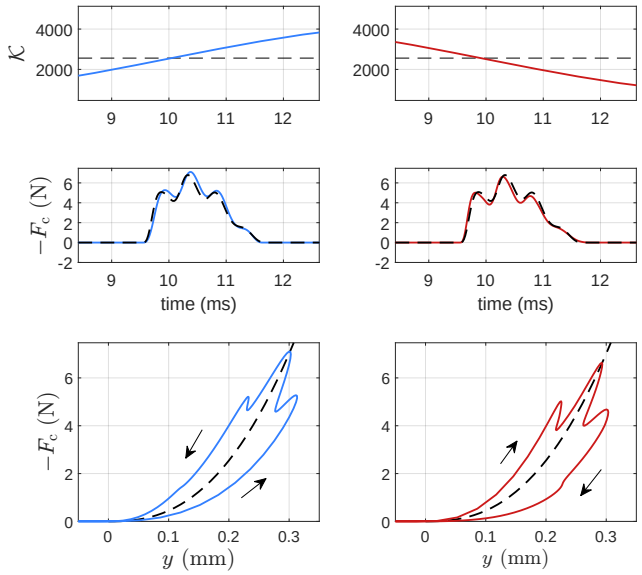


Figure 7: *Simulation with time-variation in the contact stiffness parameter (*$\mathcal{K}$*). Left: increasing stiffness. Right: decreasing stiffness. In each subplot, the dashed line indicates the simulation with constant* $\mathcal{K}$*. The parameters are those for a piano C4 string, with* $\Delta_t = 1/44100$ *s and* $N_\mathrm{b} = 32$.

rameters listed in Table 1 were used in the simulations, applying regular hammering with randomised hammer velocities across 0.5 s simulation time. For each set (N_b, M), the RTF was calculated as an average over 50 simulations. The computations were performed in Matlab on an Intel i7-6700 CPU. As can be seen from the results presented in Figure 8, the RTF remains below 0.75 for block sizes of 32 and above with up to 1000 string modes. For reference, the number of modes needed to cover the audio range for an A0, which is the lowest key on standard pianos, is about 300. Because of the uncoupled structure of the modal update form, significant efficiency gains compared to Matlab implementations can be made utilising parallel processing methods in optimised C++ implementations. Examples include the use of Advanced Vector Extensions (see, e.g. [2]).

5. CONCLUSION

A numerical scheme for simulating hammer-string interaction with time-varying parameters has been formulated in modal form. As part of adapting the energy quadratisation approach to modelling collisions under parameter time variance, two new gradient variables were introduced. The physical correctness of the algorithm is underpinned by the numerical contact force being provably non-adhesive and by the existence of a numerical energy balance, the form of which directly mirrors that of the continuous-domain model.

With respect to handling time-varying string tension, modes with frequencies that exceed the available frequency bandwidth for a given time step remain active within the numerical model, with their frequencies adjusted to fall into a narrow frequency band just below the Nyquist frequency. These out-of-range modes continue to contribute to the overall system energy (hence a numerical energy balance exists) but are suppressed in the calculation of the output signal and of the string displacement at the hammer position and as such do not interfere with the sound synthesis process while they are out of range. In addition, a pragmatic form of re-scaling of the output force signal under time-variance in the string tension was introduced to avoid large amplitude fluctuations, meaning a much reduced need to make adjustments to the output gain.

The off-line numerical experiments conducted within this work indicate that the algorithm's computational load is sufficiently small for real-time implementation and that no audible artefacts arise under parameter time-variance. More exhaustive testing on the latter point is still required though, and this is perhaps best done through on-line control. A real-time controlled implementation will also provide better opportunities to explore the possibilities that the proposed model can offer as the sound engine of a (live-performable) virtual-acoustic musical instrument.

6. ACKNOWLEDGMENTS

This work was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 812719.

7. REFERENCES

[1] S. Mehes, M. van Walstijn, and P. Stapleton, "Virtual-Acoustic Instrument Design: Exploring the Parameter Space of a String-Plate Model," in *New Interfaces for Musical Expression*, Copenhagen, 2017, pp. 399–403.

[2] M. van Walstijn and S. Mehes, "An explorative string-bridge-plate model with tunable parameters," in *20th Int. Conf. on Digital Audio Effects (DAFx-17)*, 2017, pp. 291–298.

[3] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "The dynamic grid: time-varying parameters for musical instrument simulations based on finite-difference time-domain schemes," *J. Aud. Eng. Soc.*, vol. 70, no. 9, pp. 650–660, september 2022.

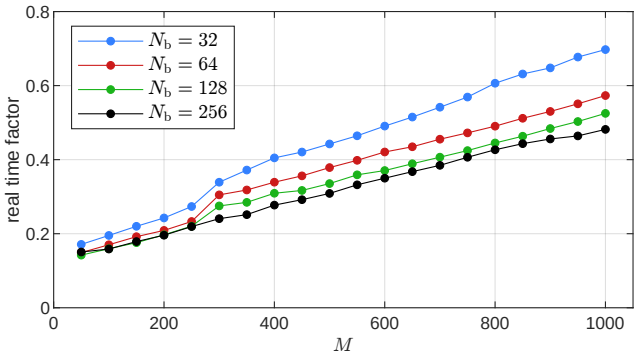


Figure 8: *Matlab real-time factor for specific combinations of number of modes (M) and block size (N_b).*

[4] J. Pakarinen, T. Puputti, and V. Välimäki, "Virtual Slide Guitar," *Computer Music Journal*, vol. 32, no. 3, pp. 42–54, 09 2008.

[5] C. Cadoz, A. Luciani, and J. Florens, "Cordis-anima: A modeling and simulation system for sound and image synthesis: The general formalism," *Computer Music Journal*, vol. 17, no. 1, pp. 19–29, 1993.

[6] J. Morrison and J. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[7] S. Bilbao and A. Torin, "Numerical simulation of string/barrier collisions: The fretboard," in *Proc. of the 17th Int. Conf. on Digital Audio Effects (DAFx-14), Erlangen, Germany, September 1-5, 2014*, 2014, pp. 137–144.

[8] C. Desvages and S. Bilbao, "Two-polarisation physical model of bowed strings with nonlinear contact and friction forces, and application to gesture-based sound synthesis," *Applied Sciences*, vol. 6, no. 5, 2016.

[9] S. Bilbao, J. Perry, P. Graham A. Gray, K. Kavoussanakis, G. Delap, T. Mudd, G. Sassoon T. Wishart, and S. Young, "Large-Scale Physical Modeling Synthesis, Parallel Computing, and Musical Experimentation: The NESS Project in Practice," *Computer Music Journal*, vol. 43, no. 2-3, pp. 31–47, 2019.

[10] A. Bhanuprakash, M. van Walstijn, and P. Stapleton, "A finite difference model for articulated slide-string simulation," in *Proc. Int. Conf. Digital Audio Effects (DAFx-20),*. Sept. 2020, pp. 195–202, University of Music and Performing Arts Vienna.

[11] N. Lopes, T. Hélie, and A. Falaize, "Explicit second-order accurate method for the passive guaranteed simulation of port-Hamiltonian systems," in *5th IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control LHMNC 2015*, Lyon, France, July 2015, IFAC, vol. 48 of *IFAC-PapersOnLine*, pp. 223–228.

[12] J. Zhao, Q. Wang, and Z. Yang, "Numerical approximations for a phase field dendritic crystal growth model based on the invariant energy quadratization approach," *Int. J. Num. Meth. Eng.*, vol. 110, no. 3, pp. 279–300, 2017.

[13] J. Shen, J. Xu, and J. Yang, "The scalar auxiliary variable (SAV) approach for gradient flows," *J. Computational Physics*, vol. 353, pp. 407–416, 2018.

[14] M. Ducceschi and S. Bilbao, "Simulation of the geometrically exact nonlinear string via energy quadratisation," *J. Sound and Vibr.*, vol. 534, pp. 117021, 2022.

[15] S. Bilbao, M. Ducceschi, and F. Zama, "Explicit exactly energy-conserving methods for hamiltonian systems," *J. Computational Physics*, vol. 472, pp. 111697, 2023.

[16] M. Ducceschi and S. Bilbao, "Non-iterative solvers for nonlinear problems: the case of collisions," in *Proc. of the 22nd Conf. of Digital Audio Effects (DAFx-19)*, Birmingham, UK, September 2019, pp. 17–24.

[17] M. Ducceschi, S. Bilbao, S. Willemsen, and S. Serafin, "Linearly-implicit schemes for collisions in musical acoustics based on energy quadratisation," *J. Acoust. Soc. Am.*, vol. 149, no. 5, pp. 3502–3516, 2021.

[18] A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. I. A physical model for a struck string using finite difference methods," *J. Acoust. Soc. Am.*, vol. 95, pp. 1112–1118, 1994.

[19] N. H. Fletcher and T. D Rossing, *The Physics of Musical Instruments*, Springer-Verlag, New York, 1991, Second Edition: 1998.

[20] A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. II. Comparisons with measurements and systematic exploration of some hammer-string parameters," *J. Acoust. Soc. Am.*, vol. 95, no. 3, pp. 1631–1640, 1994.

[21] J. Woodhouse, "Plucked guitar transients: Comparison of measurements and synthesis," *Acta Acustica united with Acustica*, vol. 90, no. 5, pp. 945–965, 2004.

[22] M. van Walstijn, J. Bridges, and S. Mehes, "A Real-Time Synthesis Oriented Tanpura Model," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, 2016, pp. 175–182.

8. APPENDIX: BRANCHED EVALUATION OF $g_{\bar{y}}^n$

For the update equations (55,56), the gradient variable $g_{\bar{y}}^n$ is calculated as

$$g_{\bar{y}}^n = \begin{cases} \min(\dot{\gamma}_{\bar{y}}^n, \dot{g}_{\bar{y}+}^n) & : \quad \bar{y}^n > 0 \\ \dot{g}_{\bar{y}+}^n & : \quad \bar{y}^n < 0 \quad \& \quad \bar{y}^{n-1} > 0 \\ 0 & : \quad \text{otherwise} \end{cases}, \tag{60}$$

where $\dot{\gamma}_{\bar{y}}^n$ denotes the nominal value according to (17):

$$\dot{\gamma}_{\bar{y}}^n = g_{\bar{y}}(\bar{y}^n) = \sqrt{\tfrac{1}{2}\bar{\kappa}^n(\alpha^n+1)\,[\bar{y}^n]_+^{\alpha^n-1}}. \tag{61}$$

The term $\dot{g}_{\bar{y}+}^n$ is the positive root of the quadratic term on the left-hand side of (57) that needs to remain non-positive to ensure that $\psi^{n+\frac{1}{2}} \geq 0$:

$$\dot{g}_{\bar{y}+}^n = 2\frac{-z^n + \sqrt{(z^n)^2 + \xi^n \psi^{n-\frac{1}{2}}\left[\psi^{n-\frac{1}{2}} + \chi^n\right]_+}}{\sqrt{\varepsilon\xi^n(z^n)^2 + (\xi^n\psi^{n-\frac{1}{2}})^2}}. \tag{62}$$

The inclusion of the term $\varepsilon\xi^n(z^n)^2$ in the denominator, where $\varepsilon > 0$ is of the order of the machine epsilon, helps ensure that the correct solution is found for $\psi^{n-\frac{1}{2}} \to 0$. The operator $[.]_+$ is applied within the square root to ensure a real root and that $\dot{g}_{\bar{y}}^n \geq 0$. The middle branch in (60) effectively sets $\psi^{n+\frac{1}{2}}$ to zero at the end of contact, as such altogether avoiding any spurious non-zero contact force values at time instances where there is no contact.

EFFICIENT SIMULATION OF THE YAYBAHAR USING A MODAL APPROACH

Riccardo Russo, Michele Ducceschi

Department of Industrial Engineering
University of Bologna
Viale Risorgimento 2, Bologna, Italy
riccardo.russo19@unibo.it
michele.ducceschi@unibo.it

Stefan Bilbao

Acoustics & Audio Group
University of Edinburgh
Edinburgh, UK
sbilbao@ed.ac.uk

ABSTRACT

This work presents a physical model of the yaybahar, a recently invented acoustic instrument. Here, output from a bowed string is passed through a long spring, before being amplified and propagated in air via a membrane. The highly dispersive character of the spring is responsible for the typical synthetic tonal quality of this instrument. Building on previous literature, this work presents a modal discretisation of the full system, with fine control over frequency-dependent decay times, modal amplitudes and frequencies, all essential for an accurate simulation of the dispersive characteristics of reverberation. The string-bow-bridge system is also solved in the modal domain, using recently developed non-iterative numerical methods allowing for efficient simulation.

1. INTRODUCTION

The yaybahar is an acoustic musical instrument, recently invented by Turkish artist Görkem Şen [1]. It consists of a neck, with two strings and a fretboard, to which two long springs are attached; each spring is in turn connected, on its opposite end, to a tensioned membrane. The instrument, depicted in Figure 1, is played by either bowing and plucking the strings, or by hitting the springs and the membranes with a mallet. The yaybahar can be described as a cello-like instrument, where amplification is provided by springs and membranes, and not by a resonant body. This structure provides a distinctive reverberant sound, mainly due to the characteristic sound transmission of springs [1]. Given its modular design, the yaybahar lends itself well to physical modeling simulation: in fact, all its components are widely studied systems in physical modeling literature [1, 2, 3]; therefore, a model can be implemented by first simulating the different modules, and by then connecting them together appropriately. A first physical model of the yaybahar was recently proposed by Christensen et al. [4]. There, the strings and membranes are described by the Kirchhoff-Carrier and the Berger models respectively [2, Chapters 8, 13], thus incorporating mild nonlinear effects, while the spring is modeled by a linear stiff bar, following [5]. The components are then coupled by lumped, spring-like connections, and the full model is simulated by using finite-difference-time-domain (FDTD) methods.

[1] https://www.gorkemsen.com/gorkem-sen-s-yaybahar

[2] https://commons.wikimedia.org/wiki/File:Yaybahar.jpg

Figure 1: *The yaybahar (Source [2])*

In this work, a different approach is proposed, based on a modal decomposition of the subsystems. The bowed string, in particular, is simulated in the modal domain following the non-iterative procedure developed in [6], and building on the results presented in [7]. The spring and the membrane, acting as reverberation units, can be simulated efficiently in the modal domain, incorporating refined loss profiles for realistic reverberation [8, 9]. Here, the interconnection between subsystems is performed in an energy-consistent framework via boundary forces, rather than using additional spring-like connections as in [4]. To this end, a novel model for the coupling between a vibrating string and a distributed bridge is presented, in the modal domain, which serves as an emulation for the neck. Given the low amplitude of vibration in the subsystems, linear models for the resonators are adopted without compromising the realism of the sound synthesis overall, as the nonlinear bowing mechanism is largely responsible for the typical drone-like sound of the instrument.

The paper is structured as follows: Section 2 presents the mathematical models of the instrument subcomponents, Section 3 describes the semi-discretisation in the modal domain, Section 4 illustrates the time-stepping algorithms, Section 5 presents the results of a case study and, finally, Section 6 concludes the paper.

2. MODELS

In this section, continuous models for the various components of the yaybahar are presented. A diagram of the instrument's subsystems (bow, string and bridge) and their couplings is as shown in Figure 2. The resulting bridge force is fed to the spring, and the spring sets the membrane into vibration. As shown below,

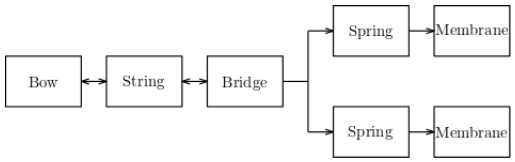


Figure 2: *A scheme of the instrument model and the elements connections.*

the string-bridge coupling modifies significantly the distribution of the eigenfrequencies compared to the isolated string. On the other hand, the spring and the membrane, acting as reverberant units characterised by a large modal density, are less affected by couplings at the boundaries. This justifies their inclusion as non-interacting subsystems, thus considerably simplifying the modal approach. This design shares some similarities with the commuted synthesis approach [10].

For simplicity, in this section, the models are presented here with no damping, except for friction losses induced by the bow. Viscous-type and radiation damping will be introduced in a frequency-dependent manner in the modal domain, as illustrated in Section 3.4.

2.1. Bowed Stiff String and Distributed Bridge

The equations for the coupled bowed string/bridge system are given here as follows:

$$\rho_s \partial_t^2 u_\mathrm{s} = T_\mathrm{s} \partial_x^2 u_\mathrm{s} - \kappa_\mathrm{s}^2 \partial_x^4 u_\mathrm{s} - F_\mathrm{b} \delta(x - x_\mathrm{b}) \phi(\eta), \tag{1a}$$

$$\rho_\mathrm{p} \partial_t^2 u_\mathrm{p} = -\kappa_\mathrm{p}^2 \partial_z^4 u_\mathrm{p} + \delta(z - z_\mathrm{s}) F_\mathrm{s}(t). \tag{1b}$$

Here, subscripts $\mathrm{s}, \mathrm{p}, \mathrm{b}$ refer, respectively, to the string, the bridge ("ponticello") and the bow. In the system above assumes that the string and the bridge vibrate in a single, vertical polarisation, thus neglecting the rocking motion observed in instruments such as the violin [11]. In (1a), $u_\mathrm{s} = u_\mathrm{s}(x,t) : [0, L_\mathrm{s}] \times \mathbb{R}_0^+ \to \mathbb{R}$ represents

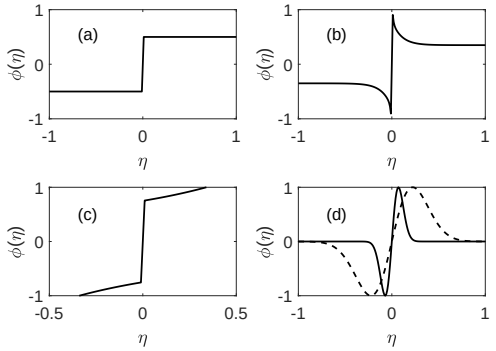


Figure 3: *Some friction characteristics: (a) Coulomb dry friction; (b) the curve by Woodhouse and Smith [12]; (c) the curve by Galluzzo [13]; (d) the continuous curve defined in equation 3, with* $a = 10$ *(dashed line) and* $a = 100$ *(solid line). For the mathematical expressions of the four friction characteristics refer to [6].*

the transverse displacement of a string of length L_s in a single polarisation, as a function of spatial coordinate x and time t. ρ_s is the string linear density in kg m^{-1}; T_s is the string tension in N, and κ_s is a rigidity constant in N$^{1/2}$m (κ_s^2 is typically given as the product of Young's modulus times the moment of inertia). Analogous definitions hold for (1b), the equation describing the displacement $u_\mathrm{p} = u_\mathrm{p}(z,t) : [0, L_\mathrm{p}] \times \mathbb{R}_0^+ \to \mathbb{R}$ of the bridge. Here and elsewhere the n$^{\text{th}}$ partial derivative with respect to the variable α is denoted by ∂_α^n.

In (1a), the string is coupled with a bow model, following [6]. The bow excitation is assumed to act pointwise downward at x_b, according to the dimensionless friction coefficient ϕ, as seen in [14]. Various choices for this coefficient are available, see e.g. [12, 13, 2] and also Figure 3. Note that all the four curves displayed in Figure 3 satisfy:

$$\eta\, \phi(\eta) \geq 0, \quad \lim_{|\eta| \to 0} \phi(\eta)/\eta < \infty\,, \tag{2}$$

allowing a non-iterative time stepping procedure to be used, following recent results in [7]. Here, for illustrative purposes, the "soft" characteristic defined in [2] was chosen, defined as:

$$\phi(\eta) = \sqrt{2a}\, \eta\, e^{-a\eta^2 + \frac{1}{2}}, \;\; \eta := \partial_t u(x_\mathrm{b}, t) - v_\mathrm{b}. \tag{3}$$

The input bow parameters are the bow force F_b, in N, and the bow velocity v_b in m s^{-1}. Furthermore, in (3), a is a free parameter of the model adjusting the slope of the curve.

The coupling between the string and the bridge takes place at the string's right boundary, and is expressed as an input force F_s in the bridge equation (1b). The string is assumed to be in contact with the bridge at z_s along the bridge's domain. The string's left boundary, as well as the bridge's endpoints, are all assumed to be simply-supported. The complete set of boundary conditions to be imposed is:

$$u_\mathrm{p}(0,t) = \partial_z^2 u_\mathrm{p}(0,t) = u_\mathrm{p}(L_\mathrm{p},t) = \partial_z^2 u_\mathrm{p}(L_\mathrm{p},t) = 0, \tag{4a}$$

$$u_\mathrm{s}(0,t) = \partial_x^2 u_\mathrm{s}(0,t) = \partial_x^2 u_\mathrm{s}(L_\mathrm{s},t) = 0, \tag{4b}$$

$$F_\mathrm{s}(t) = -T_0 \partial_x u_\mathrm{s}(L_\mathrm{s},t) + \kappa_\mathrm{s}^2 \partial_x^3 u_\mathrm{s}(L_\mathrm{s},t), \tag{4c}$$

$$u_\mathrm{p}(z_\mathrm{s},t) = u_\mathrm{s}(L_\mathrm{s},t). \tag{4d}$$

The relations above are assumed to hold $\forall t \geq 0$. Note that (4d) represents a rigid contact condition between the string and the bridge.

2.1.1. Energy Balance

An energy balance for the bridge is obtained after multiplying (1b) by $\partial_t u_\mathrm{p}$ and integrating over $[0, L_\mathrm{p}]$. After integration by parts, and owing to (4a), one obtains:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_0^{L_\mathrm{p}} \underbrace{\left(\frac{\rho_\mathrm{p}}{2} (\partial_t u_\mathrm{p})^2 + \frac{\kappa_\mathrm{p}^2}{2} (\partial_z^2 u_\mathrm{p})^2 \right)}_{\mathcal{H}_\mathrm{p}} \mathrm{d}z = \dot{u}_\mathrm{p}(z_\mathrm{s},t) F_\mathrm{s}. \tag{5}$$

The string energy balance is obtained analogously, by multiplying equation (1a) by $\partial_t u_\mathrm{s}$ and integrating. After suitable integration by parts, and taking into account the boundary conditions (4b) and (4c), one obtains:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_0^{L_\mathrm{s}} \underbrace{\left(\frac{\rho_\mathrm{s}}{2} (\partial_t u_\mathrm{s})^2 + \frac{T_\mathrm{s}}{2} (\partial_x u_\mathrm{s})^2 + \frac{\kappa_\mathrm{s}^2}{2} (\partial_x^2 u_\mathrm{s})^2 \right)}_{\mathcal{H}_\mathrm{s}} \mathrm{d}x = $$

$$= -F_\mathrm{s} \dot{u}_\mathrm{s}(L_\mathrm{s},t) - F_\mathrm{b}\, \dot{u}(x_\mathrm{b},t)\, \phi(\dot{u}(x_\mathrm{b},t)). \tag{6}$$

Finally, owing to the contact condition (4d), and by means of (5), one may express (6) as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_0^{L_\mathrm{s}} \mathcal{H}_\mathrm{s}\,\mathrm{d}x + \int_0^{L_\mathrm{p}} \mathcal{H}_\mathrm{p}\,\mathrm{d}z\right) = -\mathcal{Q} + \mathcal{P}, \tag{7}$$

where the dissipated and supplied power are defined as, respectively, $\mathcal{Q} := F_\mathrm{b}\,\eta\,\phi(\eta)$, $\mathcal{P} := -F_\mathrm{b}\,v_\mathrm{b}\,\phi(\eta)$. Owing to property (2), in the combined system the energy is non-increasing when the bow velocity v_b (and, therefore, the supplied power $\mathcal{P}$) is identically zero, leading to boundedness of the solutions.

Finally, the output force exerted by the bridge onto the spring is then computed at the desired location z_o as:

$$F_\mathrm{p}(t) = -\kappa_\mathrm{p}^2 \partial_z^3 u_\mathrm{p}(z_\mathrm{o}, t). \tag{8}$$

2.2. Spring

A model of a thin spring that takes into account the helical structure is here implemented, following [3]. In fact, the "bar" model of a spring holds for specific geometries [5], and cuts the low-frequency echoes that are characteristic of spring reverberators [1]. The system is developed starting from Wittrick's equations [15], under the assumption that the wire radius-helix radius ratio r_c/R_c is small [16] (see Figure 4). This allows the model to be reduced to a system of four equations, which relate the displacement in the transverse and longitudinal directions to the moments along the same directions:

$$\mathbf{A}\partial_t^2\mathbf{v} = \mathbf{R}\,\partial_s\mathbf{m} + \delta(s - s_\mathrm{p})\boldsymbol{\alpha}_\mathrm{p}F_\mathrm{p}, \quad \mathbf{D}\mathbf{m} = \mathbf{R}\partial_s\mathbf{v}. \tag{9}$$

Here, $\mathbf{v} := [v_\tau(s,t), v_\lambda(s,t)]^\intercal : [0, L_c] \times \mathbb{R}_0^+ \to \mathbb{R}^2$ is the vector of displacements, where the subscripts τ and λ refer to the transverse and longitudinal directions, respectively. Analogously, $\mathbf{m} := [m_\tau, m_\lambda]^\intercal$ is the vector of moments. Above, s expresses the arclength of the coil, such that $0 \leq s \leq L_c$. The input force F_p, computed in equation (8), is applied pointwise at $s = s_\mathrm{p}$, while the vector $\boldsymbol{\alpha}_\mathrm{p}$ is a unit vector that indicates the amount of force exerted in both polarisations. The matrices $\mathbf{A}, \mathbf{D} \in \mathbb{R}^{2\times2}$ are diagonal, and given by:

$$\mathbf{A} = \rho_c \begin{bmatrix} 1 & 0 \\ 0 & 1 - l^2\partial_s^2 \end{bmatrix}, \quad \mathbf{D} = \kappa_c^{-2} \begin{bmatrix} 1 & 0 \\ 0 & 1 + \nu_c - l^2\partial_s^2 \end{bmatrix}. \tag{10}$$

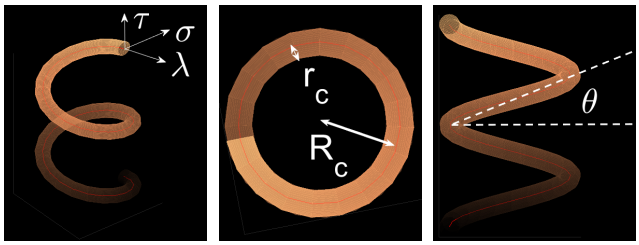
Above, ρ_c is the linear density of the coil, in kg m^{-1}, κ_c is a rigidity constant in N$^{1/2}$m, and ν_c is the Poisson ratio of the coil. $\mathbf{R} \in \mathbb{R}^{2\times2}$ is a symmetric matrix, of the form:

$$\mathbf{R} = \begin{bmatrix} -2\mu/l & (1-\mu^2)/l + l\partial_s^2 \\ 0 & 2\mu(l\partial_s^2 + 1/l) \end{bmatrix}. \tag{11}$$

The symbol l denotes the ratio $R_c/\cos^2(\theta)$, where θ is the pitch angle, R_c the coil radius, and μ is shorthand for $\tan(\theta)$. A graphical representation of the spring physical quantities is provided in Figure 4.

2.2.1. Energy Analysis

The energy balance and boundary conditions may be obtained, in the zero-input ($F_\mathrm{p} = 0$) case, as follows. First, left-multiply the first equation in (9) by $\partial_t\mathbf{v}^\intercal$. Then, take a time derivative of the second equation in (9), and left-multiply by $\mathbf{m}^\intercal$, where $\intercal$ is the transposition operator. Integrating the resulting equations gives:



(a) *Directions.* (b) *Wire/helix radius.* (c) *Tilt angle.*

Figure 4: *Spring physical quantities.*

$$\int_0^{L_c} \partial_t\mathbf{v}^\intercal\mathbf{A}\,\partial_t^2\mathbf{v}\,\mathrm{d}s = \int_0^{L_c} \partial_t\mathbf{v}^\intercal\mathbf{R}\,\partial_s\mathbf{m}\,\mathrm{d}s, \tag{12a}$$

$$\int_0^{L_c} \mathbf{m}^\intercal\mathbf{D}\,\partial_t\mathbf{m}\,\mathrm{d}s = \int_0^{L_c} \mathbf{m}^\intercal\mathbf{R}(\partial_s\partial_t\mathbf{v})\,\mathrm{d}s. \tag{12b}$$

Integrating by parts to the right-hand side of (12b), one is able to express the right-hand side of (12a), which, in turn, can be rewritten as:

$$\int_0^{L_c} \left(\partial_t\mathbf{v}^\intercal\mathbf{A}\,\partial_t^2\mathbf{v} + \mathbf{m}^\intercal\mathbf{D}\,\partial_t\mathbf{m}\right)\mathrm{d}s = \left(\mathcal{B}_0 + \mathbf{m}^\intercal\mathbf{R}\,\partial_t\mathbf{v}\right)\Big|_0^{L_c}.$$

Here

$$\mathcal{B}_0 := l\partial_s\partial_t v_\lambda\,(\partial_s m_\tau + 2\mu\partial_s m_\lambda) + \partial_t v_\lambda(2\mu l\partial_s^2 m_\lambda - l\partial_s^2 m_\tau) - l\partial_s\partial_t v_\tau(\partial_s m_\lambda) - \partial_t v_\tau(l\partial_s^2 m_\lambda). \tag{13}$$

By further applying integration by parts, one derives the energy balance:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_0^{L_c} \mathcal{H}_\mathrm{c}\,\mathrm{d}s = \left(\mathcal{B}_0 + \mathcal{B}_1 + \mathbf{m}^\intercal\mathbf{R}\,\partial_t\mathbf{v}\right)\Big|_0^{L_c} := \mathcal{B}_\mathrm{c}\Big|_0^{L_c}. \tag{14}$$

Here, the energy density is:

$$\mathcal{H}_\mathrm{c} = \frac{\rho_\mathrm{c}}{2}\left((\partial_t v_\tau)^2 + (\partial_t v_\lambda)^2 + l^2(\partial_t\partial_s v_\lambda)^2\right) + \frac{\kappa_\mathrm{c}^{-2}}{2}\left(m_\tau^2 + (1+\nu_\mathrm{c})m_\lambda^2 + l^2(\partial_s m_\lambda)^2\right); \tag{15}$$

while $\mathcal{B}_1$ is:

$$\mathcal{B}_1 = l^2\rho_\mathrm{c}(\partial_t v_\lambda)(\partial_s\partial_t^2 v_\lambda) + l^2\kappa_\mathrm{c}^{-2}m_\lambda(\partial_s\partial_t m_\lambda). \tag{16}$$

It is useful to write the boundary terms in terms of the conjugate variables forces / velocities and moments / angular velocities [3]. To that end, rearranging the boundary terms in (14) allows to write:

$$\mathcal{B}_\mathrm{c} = F_\tau\partial_t v_\tau + F_\lambda\partial_t v_\lambda + F_\sigma\partial_t v_\sigma + m_\tau\phi_\tau + m_\lambda\phi_\lambda + m_\sigma\phi_\sigma.$$

Here, the σ denotes the direction perpendicular to the (τ, λ)-plane. Denoting $g := -\frac{\mu^2}{l} + \frac{1}{l} + l\partial_s^2$, one has

$$F_\tau = g\, m_\lambda - 2\mu m_\tau / l, \tag{17a}$$
$$F_\lambda = g\, m_\tau + 2\mu \left(l\partial_s^2 + 1/l\right) m_\lambda + l^2 \rho_\mathrm{c} \partial_s \partial_t^2 v_\lambda, \tag{17b}$$
$$F_\sigma = -\partial_s (m_\tau + 2\mu m_\lambda), \tag{17c}$$
$$\phi_\tau = l\partial_s^2 \partial_t v_\lambda, \tag{17d}$$
$$\phi_\lambda = 2\mu l \partial_s^2 \partial_t v_\lambda + l^2 \kappa_\mathrm{c}^{-2} \partial_s \partial_t m_\lambda + l\partial_s^2 \partial_t v_\tau, \tag{17e}$$
$$\phi_\sigma = -\partial_s \partial_t v_\tau, \tag{17f}$$
$$m_\sigma = l\partial_s m_\lambda, \tag{17g}$$
$$v_\sigma = l\partial_s v_\lambda. \tag{17h}$$

Setting boundary displacements, forces and moments to zero leads to a generalisation of the classic beam boundary conditions of free, simply-supported or clamped type. Here, a variant of free boundary conditions along τ, λ will be used, combined with clamped conditions along σ. Hence:

$$F_\tau = F_\lambda = m_\tau = m_\lambda = v_\sigma = \phi_\sigma = 0. \tag{18}$$

These are intended to hold at the boundary points $s = \{0, L_\mathrm{c}\}$, and $\forall t \geq 0$. An output signal $F_\mathrm{c}(t)$ may be then be extracted by computing the sum of the forces F_τ, F_λ, F_σ at a s_o, close, but not equal, to the boundary L_c. Thus:

$$F_\mathrm{c}(t) = F_\tau(s_\mathrm{o}, t) + F_\lambda(s_\mathrm{o}, t) + F_\sigma(s_\mathrm{o}, t). \tag{19}$$

2.3. Membrane

A model for the membrane is given by the 2D wave equation [2, Chapter 11]:

$$\rho_m \partial_t^2 w = T_m \nabla^2 w + \delta(X - X_\mathrm{c})\delta(Y - Y_\mathrm{c}) F_\mathrm{c}(t). \tag{20}$$

In the above, the two dimensional Laplacian was introduced as $\nabla^2 := \partial_X^2 + \partial_Y^2$. For simplicity, and to avoid the introduction of further symbols, the membrane is supposed to be defined over a square, of side length L_m. Thus, $w = w(X, Y, t) : [0, L_m] \times [0, L_m] \times \mathbb{R}^+ \to \mathbb{R}$ describes the displacement of the membrane in the transversal direction, ρ_m is the material surface density in kg m^{-2}, and T_m is the tension applied at the edges in N m^{-1}. An energy analysis for this system can be found in [2, Chapter 11]. Boundary conditions of fixed type will be considered here, such that:

$$w(0, Y) = w(L_m, Y) = w(X, 0) = w(X, L_m) = 0. \tag{21}$$

3. SEMI-DISCRETISATION

The equations presented in Section 2 will be now semi-discretised in space using a modal approach. While the spring and the membrane possess an analytical form for the modes of vibration, this is not true in the case of a string coupled with a distributed bridge on one end. For this reason, the modal expansion for the latter system will be performed by solving the eigenvalue problem numerically.

3.1. Bowed Stiff String and Distributed Bridge

First, it is convenient to introduce spatial difference operators. The string domain is divided into M_s subintervals of length h, the grid spacing. This yields $M_\mathrm{s} + 1$ discretisation points, including the end points. Analogously, the bridge is divided into M_p subintervals of length h. The continuous functions $u_\mathrm{s}(x, t)$ and $u_\mathrm{p}(z, t)$ are then approximated by grid functions $u_\mathrm{s}^m(t) \approx u_\mathrm{s}(mh, t)$ and $u_\mathrm{p}^n(t) \approx u_\mathrm{p}(nh, t)$, for integer m, n. In light of the numerical boundary conditions given below, one has $m \in [1, ..., M_\mathrm{s} - 1]$, $n \in [1, ..., M_\mathrm{p} - 1]$. In vector notation, the grid functions will be denoted $\mathbf{u}_\mathrm{s}$, $\mathbf{u}_\mathrm{p}$.

Basic forward and backward difference operators, approximating the first spatial derivative, and acting on u_s^m, are:

$$\delta_x^\pm u_\mathrm{s}^m = \pm(u_\mathrm{s}^{m\pm1} - u_\mathrm{s}^m)/h. \tag{22}$$

Analogous definitions hold for the grid function u_p^n, thus, for instance, $\delta_z^+ u_\mathrm{p}^n = (u_\mathrm{p}^{n+1} - u_\mathrm{p}^n)/h$. The second and fourth spatial derivatives are approximated by difference operators obtained by combining the operators above, as:

$$\delta_x^2 := \delta_x^+ \delta_x^-, \qquad \delta_x^4 := \delta_x^2 \delta_x^2, \tag{23}$$

with similar definitions holding for δ_z^2, δ_z^4. Discrete versions of the Dirac deltas in (1) are also needed. To that end, $\delta(x - x_\mathrm{b})$ in (1a) is approximated by the column vector $\mathbf{d}_\mathrm{b}$, of length $M_\mathrm{s} - 1$, as:

$$d_\mathrm{b}^\nu = (1 - \alpha)/h, \qquad d_\mathrm{b}^{\nu+1} = \alpha/h, \tag{24}$$

where $\nu := \mathrm{floor}(x_\mathrm{b}/h)$, $\alpha := x_\mathrm{b}/h - \nu$. An analogous definition holds for $\mathbf{d}_\mathrm{s}$, approximating $\delta(z - z_\mathrm{s})$ in (1b).

3.1.1. Semi-Discrete Formulation

Given the definitions above, a semi-discrete approximation of (1) is given as:

$$\rho_\mathrm{s} \ddot{u}_\mathrm{s}^m = T_\mathrm{s} \delta_x^2 u_\mathrm{s}^m - \kappa_\mathrm{s}^2 \delta_x^4 u_\mathrm{s}^m - F_\mathrm{b} d_\mathrm{b}^m \phi(\eta), \tag{25a}$$
$$\rho_\mathrm{p} \ddot{u}_\mathrm{p}^n = -\kappa_\mathrm{b}^2 \delta_z^4 u_\mathrm{p}^n + d_\mathrm{s}^n f_\mathrm{s}(t). \tag{25b}$$

Here, $\eta = h\mathbf{d}_\mathrm{b}^\intercal \dot{\mathbf{u}}_\mathrm{s} - v_\mathrm{b}$. A discrete version of the boundary conditions and contact condition (4) ensuring numerical stability is:

$$u_\mathrm{p}^0 = \delta_z^2 u_\mathrm{p}^0 = u_\mathrm{p}^{M_\mathrm{p}} = \delta_z^2 u_\mathrm{p}^{M_\mathrm{p}} = 0, \tag{26a}$$
$$u_\mathrm{s}^0 = \delta_x^2 u_\mathrm{s}^0 = \delta_x^2 u_\mathrm{s}^{M_\mathrm{s}} = 0, \tag{26b}$$
$$f_\mathrm{s}(t) = -T_\mathrm{s} \delta_x^+ u_\mathrm{s}^{M_\mathrm{s}} + \kappa_\mathrm{s}^2 \delta_x^+ \delta_x^2 u_\mathrm{s}^{M_\mathrm{s}}, \tag{26c}$$
$$h\mathbf{d}_\mathrm{s}^\intercal \mathbf{u}_\mathrm{p} = u_\mathrm{s}^{M_\mathrm{s}}. \tag{26d}$$

By expanding the operators and applying the boundary conditions, the semi-discrete equations can be arranged in vector form. To that end, define $\mathbf{u}^\intercal := [\mathbf{u}_\mathrm{s}^\intercal, \mathbf{u}_\mathrm{p}^\intercal]$. System (25) can be then written in compact form as:

$$\mathbf{M}\ddot{\mathbf{u}} = -\mathbf{K}\mathbf{u} - F_\mathrm{b} \mathbf{J}_\mathrm{b} \phi(\eta). \tag{27}$$

Here, $\mathbf{J}_\mathrm{b}$ is a vector obtained by concatenating $\mathbf{d}_\mathrm{b}$ with a zero-vector of dimension M_{p-1}, and $\eta = h\mathbf{J}_\mathrm{b}^\intercal \dot{\mathbf{u}} - v_\mathrm{b}$. $\mathbf{M}$ is positive-definite, symmetric, square diagonal block matrix, with diagonal blocks given as:

$$\mathbf{M}_{11} = \rho_\mathrm{s} \mathbf{I}_\mathrm{s}, \ \mathbf{M}_{22} = \left(\rho_\mathrm{p} \mathbf{I}_\mathrm{p} + \rho_\mathrm{s} h^2 \mathbf{d}_\mathrm{s} \mathbf{d}_\mathrm{s}^\intercal\right). \tag{28}$$

Here, $\mathbf{I}_\mathrm{s}$ and $\mathbf{I}_\mathrm{p}$ are identity matrices, of dimension $(M_\mathrm{s} - 1) \times (M_\mathrm{p} - 1)$ and $(M_\mathrm{p} - 1) \times (M_\mathrm{s} - 1)$ respectively. Furthermore,

the stiffness matrix is a positive-definite, square block matrix, with blocks:

$$\begin{aligned}\mathbf{K}_{11} &= -T_\mathrm{s}\mathbf{D}_x^2 + \kappa_\mathrm{s}^2\mathbf{D}_x^2\mathbf{D}_x^2,\\ \mathbf{K}_{12} &= \mathbf{K}_{21}^\intercal \quad \mathbf{K}_{21} = \left[\mathbf{0}, \tfrac{\kappa_\mathrm{s}^2}{h^3}\mathbf{d}_\mathrm{s}, -\left(\tfrac{T_\mathrm{s}}{h^2} + \tfrac{2\kappa_\mathrm{s}^2}{h^3}\right)\mathbf{d}_\mathrm{s}\right],\\ \mathbf{K}_{22} &= \kappa_\mathrm{p}^2\mathbf{D}_z^2\mathbf{D}_z^2 + \left(T_\mathrm{s} + \frac{2\kappa_\mathrm{s}^2}{h^2}\right)\mathbf{d}_\mathrm{s}\mathbf{d}_\mathrm{s}^\intercal,\end{aligned} \tag{29}$$

where $\mathbf{D}_x^2$ and $\mathbf{D}_z^2$ are the second difference operators with Dirichlet end conditions of dimensions, respectively, $(M_\mathrm{s}-1)\times(M_\mathrm{s}-1)$, and $(M_\mathrm{p}-1)\times(M_\mathrm{p}-1)$ (for the explicit form of these matrices, see [2, Chapter 5]). $\mathbf{K}_{21}$ has dimension $(M_\mathrm{p}-1)\times(M_\mathrm{s}-1)$, and is a composition of a zero-matrix of dimension $(M_\mathrm{p}-1)\times(M_\mathrm{s}-3)$ with two vectors.

An energy balance in the modal domain is readily available from (27), after left-multiplying by $h\dot{\mathbf{u}}^\intercal$. When $v_\mathrm{b}=0$, one has:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{h}{2}\dot{\mathbf{u}}^\intercal\mathbf{M}\dot{\mathbf{u}} + \frac{h}{2}\mathbf{u}^\intercal\mathbf{K}\mathbf{u}\right) = -\eta\phi(\eta) \leq 0. \tag{30}$$

Since both $\mathbf{M}$, $\mathbf{K}$ are non-negative, the energy is non-negative, and decays over time.

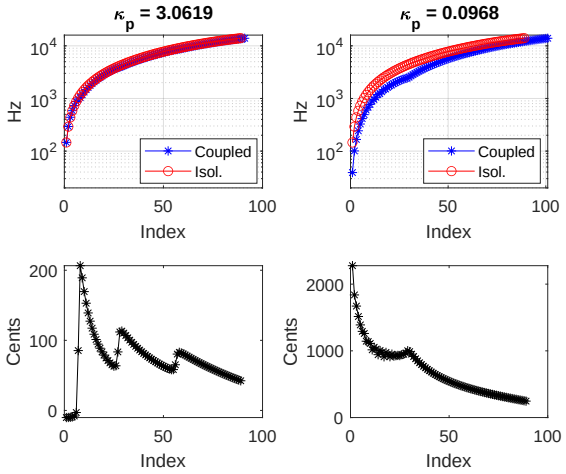


Figure 5: *Eigenfrequencies of the coupled system and the ones of a simply supported stiff string in isolation, under two different values of κ_p. The left value is typical of steel, while the right one was chosen arbitrarily low for demonstration purposes. Other physical parameters, on common between the two cases, were: $L_\mathrm{s}=0.69$ m; $T_\mathrm{s}=147.7$ N; $\rho_\mathrm{s}=0.0063$ Kg m^{-1}; $\kappa_\mathrm{s}=0.4835$ N$^{1/2}$ m; $L_\mathrm{p}=0.07$ m; $\rho_\mathrm{p}=0.0251$ Kg m^{-1}. The contact point was set to: $z_\mathrm{s}=0.03$ m. The top figures report the frequencies in log scale, while the bottom figures display the difference in cents between frequencies with the same index.*

3.1.2. Modal Expansion

A modal expansion of system (27) is now performed by solving the generalised eigenvalue problem. For that, consider the following:

$$\mathbf{K}\mathbf{U} = \mathbf{M}\mathbf{U}\boldsymbol{\Omega}_\mathrm{u}^2. \tag{31}$$

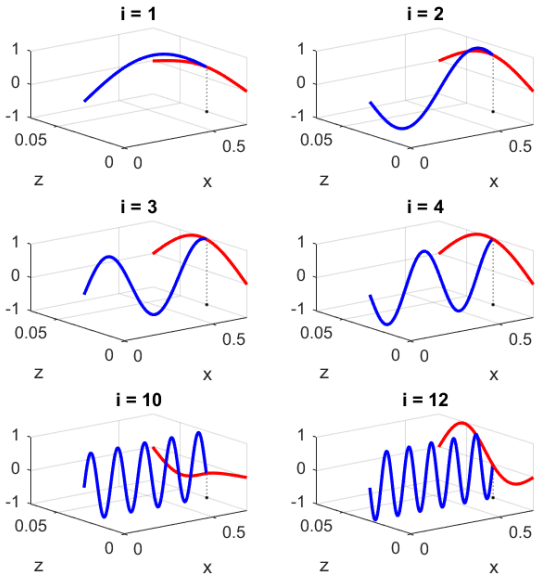


Figure 6: *Modes of the coupled string-bridge system, which were normalized, and plotted orthogonal one to another for visualisation purposes. The letter i gives the mode index. The blue line represents the string, and red one the bridge, while the projection of the contact point on the z-axis is highlighted with a black dot. The physical parameters were the same listed in Figure 5, with $\kappa_\mathrm{p}=3.0619$ N$^{1/2}$ m.*

Here, $\mathbf{U}$ is a matrix of real eigenvectors, and $\boldsymbol{\Omega}^2$ is a diagonal matrix of real, positive eigenvalues. Note that, while both $\mathbf{K}$ and $\mathbf{M}$ are symmetric, the product $\mathbf{M}^{-1}\mathbf{K}$ generally will not be. However, since $\mathbf{M}$ is positive definite, the eigenvalues are then real [17], and they must also be non-negative since so are the eigenvalues of $\mathbf{K}$. Then, define $\mathrm{u}=\mathbf{U}^{-1}\mathbf{u}$. System (27) may then be written as:

$$\ddot{\mathrm{u}} = -\boldsymbol{\Omega}_\mathrm{u}^2\mathrm{u} - F_\mathrm{b}\boldsymbol{\xi}_\mathrm{b}\phi(\eta), \quad \eta = \boldsymbol{\xi}_\eta^\intercal\dot{\mathrm{u}} - v_\mathrm{b}, \tag{32}$$

with $\boldsymbol{\xi}_\eta^\intercal := h\mathbf{J}_\mathrm{b}^\intercal\mathbf{U}$, $\boldsymbol{\xi}_\mathrm{b} := (\mathbf{M}\mathbf{U})^{-1}\mathbf{J}_\mathrm{b}$. This a modal system with a diagonal linear part, with modal coordinates u. One may of course solve the numerical eigenvalue problem (31) using a very fine grid (i.e., using a small grid spacing h), though only a number N_u is kept in (32), fixed by Nyquist requirements.

Before proceeding, it is useful to compare the eigenfrequencies of the string in isolation against those of the coupled string-bridge system. Figure 5 shows two such comparisons, under two different values of κ_p. These are computed for a bar of circular cross section, with a diameter of 5 mm. The first value ($\kappa_\mathrm{p}\approx 3$) is typical of steel, while the second value ($\kappa_\mathrm{c}\approx 0.1$) was selected to artificially amplify the effects of the coupling. The top panels report the frequencies in a log scale: it is seen that the reduced stiffness shifts the eigenfrequencies downwards by up to two octaves in the lowest range, as seen in the bottom panels. With sufficiently large values of κ_p, as is the case of steel, the frequency gap remains contained around the fundamental, as expected, but it increases up to two semitones for larger modal indices. These results underline the importance of considering the string-bridge coupling for in the distribution of the eigenfrequencies.

The eigenmodes are represented in Figure 6, for a steel bridge. Here, the first four modes are displayed, along with the 10th and the 12th. It is possible to see that the bridge eigenmodes start exhibiting a third node only after the 10th mode.

3.2. Spring

A modal version of the thin spring model was originally proposed by van Walstijn [18]. In van Walstijn's model, the modal expansion is carried out numerically, by first performing a semi-discretisation in space, and then computing the eigenvalues of the resulting matrix. Such model found practical application in [19], where a virtual analogue simulation of a spring reverb is developed. In this work an analytic form for the modes is available under a choice of the boundary conditions as per (18). To that end, consider the following:

$$\begin{aligned} \mathbf{v}(s,t) &= \sqrt{2/L_\mathrm{c}}\cos(\gamma s)\,\bar{\mathbf{v}}(t), \\ \mathbf{m}(s,t) &= \sqrt{2/L_\mathrm{c}}\sin(\gamma s)\,\bar{\mathbf{m}}(t), \end{aligned} \tag{33}$$

where $\bar{\mathbf{v}}$, $\bar{\mathbf{m}}$ are the time modal coordinates, and the factor $\sqrt{2/L_\mathrm{c}}$ is just a useful normalisation constant. It is immediate to see that these satisfy the boundary conditions (18) when $\gamma = \{\pi/L_\mathrm{c}, 2\pi/L_\mathrm{c}, ..., n\pi/L_\mathrm{c}, ...\}$.

Let now $\gamma_n := n\pi/L_\mathrm{c}$, for integer n. A solution to the equation of motion is obtained using the quantised expressions of the modes to solve an eigenvalue problem. Left-multiplying the first equation in (9) by $\sqrt{2/L_\mathrm{c}}\cos(\gamma_n s)$, and the second by $\sqrt{2/L_\mathrm{c}}\sin(\gamma_n s)$ and integrating, one is able to express (9) as:

$$\bar{\mathbf{A}}\ddot{\bar{\mathbf{v}}} = \gamma_n\bar{\mathbf{R}}\bar{\mathbf{m}} + \sqrt{2/L_\mathrm{c}}\cos(\gamma_n s_\mathrm{p})\boldsymbol{\alpha}_\mathrm{p}F_\mathrm{p}(t), \tag{34a}$$

$$\bar{\mathbf{D}}\bar{\mathbf{m}} = -\gamma_n\bar{\mathbf{R}}\bar{\mathbf{v}}. \tag{34b}$$

Here, the transformed matrices are obtained by applying the derivatives to the modal functions, and have the form (10), (11) under the replacement of ∂_s^2 by $-\gamma_n^2$. Then, (34b) is used to express $\bar{\mathbf{m}}$ is terms of $\bar{\mathbf{v}}$, and this is substituted in (34a). One gets:

$$\ddot{\bar{\mathbf{v}}} = -\mathbf{V}\boldsymbol{\Omega}_n^2\mathbf{V}^{-1}\bar{\mathbf{v}} + \sqrt{2/L_\mathrm{c}}\cos(\gamma_n s_\mathrm{p})\bar{\mathbf{A}}^{-1}\boldsymbol{\alpha}_\mathrm{p}F_\mathrm{p}.$$

where it was set:

$$\mathbf{V}\boldsymbol{\Omega}_n^2\mathbf{V}^{-1} := \gamma_n^2\bar{\mathbf{A}}^{-1}\bar{\mathbf{R}}\bar{\mathbf{D}}^{-1}\bar{\mathbf{R}}. \tag{35}$$

Here $\mathbf{V}$ is a 2×2 matrix of eigenvectors for the wavenumber γ_n, and $\boldsymbol{\Omega}_n$ is a diagnoal 2×2 matrix of eigenfrequencies. Figure 7 reports the solution to the eigenvalue problem for a typical spring. The eigenfrequencies lay on the yellow an purple lines.

Then, define $\mathfrak{v}_n := (\mathbf{V})^{-1}\bar{\mathbf{v}}$. Thus, one gets:

$$\ddot{\mathfrak{v}}_n = -\boldsymbol{\Omega}_n^2\mathfrak{v} + \sqrt{2/L_\mathrm{c}}\cos(\gamma_n s_\mathrm{p})(\bar{\mathbf{A}}\mathbf{V})^{-1}\boldsymbol{\alpha}_\mathrm{p}F_\mathrm{p}(t). \tag{36}$$

Assume now $n = 1, ..., N_\mathfrak{v}$. The modal equations for the spring are then a system of $2N_\mathfrak{v}$ equations, of the form:

$$\ddot{\mathfrak{v}} = -\boldsymbol{\Omega}_\mathfrak{v}^2\mathfrak{v} + \boldsymbol{\xi}_\mathrm{p}F_\mathrm{p}(t), \tag{37}$$

where $\mathfrak{v}$ is a vector of length $2N_\mathfrak{v}$, $\boldsymbol{\Omega}_\mathfrak{v}$ is a $2N_\mathfrak{v}\times 2N_\mathfrak{v}$ diagonal matrix, whose diagonal elements are the 2×2 diagonal blocks $\boldsymbol{\Omega}_n$, $n = 1, ...N_\mathfrak{v}$ defined in (35), and $\boldsymbol{\xi}_\mathrm{p}$ is a $2N_\mathfrak{v}$ vector made composed by stacking the 2×1 blocks: $\sqrt{2/L_\mathrm{c}}\cos(\gamma_n s_\mathrm{p})(\bar{\mathbf{A}}\mathbf{V})^{-1}\boldsymbol{\alpha}_\mathrm{p}$, $n = 1, ..., N_\mathfrak{v}$.

Output (19) is extracted by substituting the solution (33) into the boundary forces (17c) (17b) and (17a) computed at a position close to L_c.

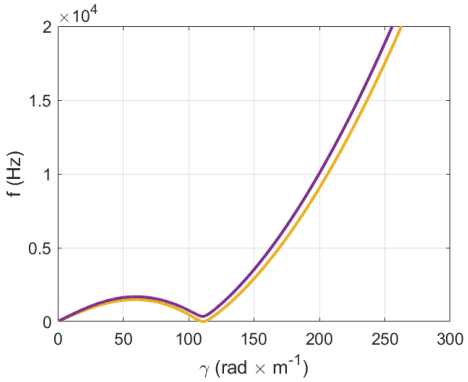


Figure 7: *Plot of the dispersion relation of a thin spring. Physical parameters were chosen to be coherent with a possible yaybahar spring:* $R_\mathrm{c} = 9$ mm, $r_\mathrm{c} = 1$ mm, $\theta = 2°$, $L_\mathrm{c} = 40$ m, $\kappa_\mathrm{c} = 9.9$ *and* $\nu_\mathrm{c} = 0.3$. *These values yield* $N_\mathfrak{v} = 2812$ *within the hearing range.*

3.3. Membrane

A particular solution to equation (20) with fixed boundary conditions (21) is given by:

$$w(X,Y,t) = \frac{2}{L_m}\sin\frac{\beta_X^j\pi X}{L_m}\sin\frac{\beta_Y^j\pi Y}{L_m}\mathfrak{w}_j(t), \tag{38}$$

for integers β_X^j, β_Y^j [20]. The associated modal frequency is:

$$\omega_j = \frac{\pi}{L_m}\sqrt{\frac{T_m}{\rho_m}\left((\beta_X^j)^2 + (\beta_Y^j)^2\right)}. \tag{39}$$

The modal system for the membrane is then given by:

$$\ddot{\mathfrak{w}} = -\boldsymbol{\Omega}_\mathfrak{w}^2\mathfrak{w} + \boldsymbol{\xi}_c F_c(t), \tag{40}$$

where $\mathfrak{w}$ is a $N_\mathfrak{w}\times 1$ vector, $\boldsymbol{\Omega}_\mathfrak{w}$ is a $N_\mathfrak{w}\times N_\mathfrak{w}$ diagonal matrix where the jth diagonal element is ω_j given above. The frequencies should here be sorted in ascending order, such that $\omega_{j-1}\leq\omega_j\leq\omega_{j+1}$, $j = 2, ..., N_\mathfrak{w} - 1$. This allows to find the corresponding modal indices β_X^j, β_Y^j. Above, $\boldsymbol{\xi}_c$ is a $N_\mathfrak{w}\times 1$ vector whose jth component is $\frac{2}{L_m\rho_m}\sin\frac{\beta_X^j\pi X_c}{L_m}\sin\frac{\beta_Y^j\pi Y_c}{L_m}$.

Output is extracted as:

$$w_\mathrm{o}(t) = \sum_{j=1}^{N_\mathfrak{w}}\frac{2}{L_m}\sin\frac{\beta_X^j\pi X_\mathrm{o}}{L_m}\sin\frac{\beta_Y^j\pi Y_\mathrm{o}}{L_m}\mathfrak{w}_j(t), \tag{41}$$

though for synthesis purposes it may be convenient to use $\dot{w}_\mathrm{o}(t)$ instead.

3.4. Modal Equations of the Full System with Loss

The full system in the modal domain can thus be written as an augmented version of (32), (37) and (40). This is:

$$\ddot{\mathfrak{u}}(t) = -\boldsymbol{\Omega}_\mathfrak{u}^2\mathfrak{u}(t) - \mathbf{C}_\mathfrak{u}\dot{\mathfrak{u}}(t) - \boldsymbol{\xi}_\mathrm{b}F_\mathrm{b}\phi(\eta), \tag{42a}$$

$$\ddot{\mathfrak{v}}(t) = -\boldsymbol{\Omega}_\mathfrak{v}^2\mathfrak{v}(t) - \mathbf{C}_\mathfrak{v}\dot{\mathfrak{v}}(t) + \boldsymbol{\xi}_\mathrm{p}F_\mathrm{p}(t), \tag{42b}$$

$$\ddot{\mathfrak{w}}(t) = -\boldsymbol{\Omega}_\mathfrak{w}^2\mathfrak{w}(t) - \mathbf{C}_\mathfrak{w}\dot{\mathfrak{w}}(t) + \boldsymbol{\xi}_c F_c(t), \tag{42c}$$

where $\mathbf{C}_\mathfrak{u}$, $\mathbf{C}_\mathfrak{v}$, $\mathbf{C}_\mathfrak{w}$ are, respectively, $N_\mathfrak{u} \times N_\mathfrak{u}$, $2N_\mathfrak{v} \times 2N_\mathfrak{v}$, $N_\mathfrak{w} \times N_\mathfrak{w}$ positive, diagonal matrices containing the modal loss coefficients in s^{-1}. System (42) depends on time only, and a suitable time stepping routine is offered below. Here, the input parameters are the bow force F_b, velocity v_b and position along the string x_b, which may be time-varying. The output is given by the membrane displacement at the output location, as per (41).

4. TIME DISCRETISATION

Now, time is discretised with a time step k, yielding a sample rate $f_s = 1/k$. Then, a continuous function $u(t)$ is approximated at time step $t = nk$ by the time series u^n. Time difference operators are then introduced, as:

$$\delta_t^\pm u^n := \pm(u^{n\pm1} - u^n)/k, \;\; \delta_t^\circ u^n := (u^{n+1} - u^{n-1})/2k. \quad (43)$$

The second time difference is defined by combining the operators above: $\delta_t^2 u^n := \delta_t^+ \delta_t^- u^n$. Finally, a time averaging operator is defined as:

$$\mu_t^+ u^n := (u^{n+1} + u^n)/2. \quad (44)$$

4.1. String-Bridge System

It is now possible to adapt the numerical solver proposed in [7, 6] to numerically integrate equation (42a) in time. To that end, (42a) is first turned into a $2N_\mathrm{u} \times 2N_\mathrm{u}$ system of first-order-in-time equations. Thus, define $\mathfrak{q} := \mathbf{\Omega}_\mathrm{u}\mathfrak{u}$, $\mathfrak{p} := \dot{\mathfrak{u}}$. Therefore, (42a) becomes:

$$\begin{aligned} \begin{bmatrix} \dot{\mathfrak{q}} \\ \dot{\mathfrak{p}} \end{bmatrix} &= \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{\Omega}_\mathrm{u} \\ -\mathbf{\Omega}_\mathrm{u} & -\mathbf{\Omega}_\mathrm{u}^{-1}\mathbf{C}_u \end{bmatrix}}_{\mathbf{G}} \begin{bmatrix} \mathfrak{q} \\ \mathfrak{p} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{\Omega}_\mathrm{u}^{-1}\boldsymbol{\xi}_\mathrm{b} \end{bmatrix} F_\mathrm{b}\phi(\eta), \\ \eta &= \left[\mathbf{0}, \boldsymbol{\xi}_\eta^\intercal\right] \begin{bmatrix} \mathfrak{q} \\ \mathfrak{p} \end{bmatrix} - v_\mathrm{b}. \end{aligned} \quad (45)$$

A second-order accurate, non-iterative numerical scheme is given as:

$$\boldsymbol{\sigma}^n \begin{bmatrix} \delta_t^+ \mathfrak{q}^n \\ \delta_t^+ \mathfrak{p}^n \end{bmatrix} = \mathbf{G} \begin{bmatrix} \mu_t^+ \mathfrak{q}^n \\ \mu_t^+ \mathfrak{p}^n \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{\Omega}_\mathrm{u}^{-1}\boldsymbol{\xi}_\mathrm{b} \end{bmatrix} F_\mathrm{b} \frac{\phi(\eta^n)}{\eta^n} \mu_{t+}\eta^n. \quad (46)$$

The form of $\boldsymbol{\sigma}^n$, adapted from [6], is:

$$\boldsymbol{\sigma}^n = \mathbf{I} + \frac{kF_\mathrm{b}}{2} \left(\frac{d\phi}{d\eta} - \frac{\phi}{\eta}\right)_{t=kn} \begin{bmatrix} \mathbf{0} \\ \mathbf{\Omega}_\mathrm{u}^{-1}\boldsymbol{\xi}_\mathrm{b} \end{bmatrix} \left[\mathbf{0}, \boldsymbol{\xi}_\eta^\intercal\right], \quad (47)$$

and is well-defined, owing to (2). Here, $\mathbf{J}_\mathrm{b}$ and $\mathbf{U}$ are as per Section 3.1.1 and 3.1.2, respectively. Expanding out the operators in (46), one is able to compute $\mathfrak{q}^{n+1}$, $\mathfrak{p}^{n+1}$ as the solution of a single linear system, thus avoiding entirely the need for iterative nonlinear root finders. The update equation in this case is:

$$\left(\mathbf{I} + \frac{kF_\mathrm{b}}{2} \left(\frac{d\phi}{d\eta}\right) \begin{bmatrix} \mathbf{0} \\ \mathbf{\Omega}_\mathrm{u}^{-1}\boldsymbol{\xi}_\mathrm{b} \end{bmatrix} \left[\mathbf{0}, \boldsymbol{\xi}_\eta^\intercal\right] - \frac{k}{2}\mathbf{G}\right) \begin{bmatrix} \mathfrak{q}^{n+1} \\ \mathfrak{p}^{n+1} \end{bmatrix} = \mathbf{b}^n,$$

where $\mathbf{b}^n$ is known from previous time steps. It is seen that the update matrix is in the form of a block matrix with fully diagonal blocks, plus a rank-1 perturbation. This can be solved efficiently, via the Sherman-Morrison formula [21], as detailed in [6].

Stability of scheme (46) is somewhat harder to prove, though partial results are available in [7]. Provided one chooses a number N_u of modes lower than the Nyquist limit, empirical evidence suggests that the proposed scheme greatly outperforms simpler explicit designs such as forward Euler or Runge-Kutta-type algorithms [22] in terms of stability, while keeping compute times within reference bounds for efficient simulation.

4.2. Spring & Membrane

The numerical integration of (42b), (42c) may be performed simply as:

$$\delta_t^2 \mathfrak{v}^n = -\mathbf{\Omega}_\mathfrak{v}^2 \mathfrak{v}^n - \mathbf{C}_\mathfrak{v} \delta_t^\circ \mathfrak{v}^n + \boldsymbol{\xi}_\mathrm{p} F_\mathrm{p}^n, \quad (48a)$$

$$\delta_t^2 \mathfrak{w}^n = -\mathbf{\Omega}_\mathfrak{w}^2 \mathfrak{w}^n - \mathbf{C}_\mathfrak{w} \delta_t^\circ \mathfrak{w}^n + \boldsymbol{\xi}_c F_c^n. \quad (48b)$$

Various other designs are possible, varying greatly in terms of stability and spectral accuracy. An attractive alternative is represented by exact integrators [23, 2], though the schemes above yield a perceptually reasonable reverberation characteristic [8]. Note that stability conditions arise as: $\mathbf{\Omega}_{\mathfrak{v},\mathfrak{w}} < 2/k$. These set upper limits for the modal frequencies.

5. OUTPUT SIGNALS

Figure 8 displays the spectrograms of the normalised signals extracted from the three subsystems. The string physical values were the ones of a C2 cello string, taken from [14], while the bridge parameters were the same listed in Figure 5. The bow pressure was $F_\mathrm{b} = 0.02$ N, and the input and output positions were set to $0.73 \cdot L_\mathrm{s}$ and $0.34 \cdot L_\mathrm{p}$ respectively. The latter values were chosen empirically to obtain a Helmholtz motion-shaped output sound [2, Chapter 7]. The damping profile applied was the one proposed by Valette [24]. The spring parameters were the ones detailed in Figure 7. Finally, the membrane physical values were: $L_m = 0.5$ m, $T_m = 3000$ N m^{-1} and $\rho_m = 1.26$ Kg m^{-2}, and the output point was $(X_\mathrm{o}, Y_\mathrm{o}) = (0.47, 0.62) \cdot L_m$. A damping profile was chosen, for the spring and the membrane, which consists of a frequency-independent (F-I) and a frequency-dependent (F-D) part; as proposed by Bilbao [2, Chapter 7] the latter depends on the square of the mode number. The damping coefficients for the spring were taken from [18]; in the mebrane case, the F-I coefficient was set to 10, while the F-D one to 5×10^{-5}, both chosen empirically. Only a few seconds-long portion of data was analysed, in order to avoid including too many signal variations. The spring and the membrane act here as reverberant components. Panel (b) from Figure 8 clearly exhibits cross stripes which correspond to the spring chirps. The membrane, on the other hand, introduces a broadband signal, which mimics late reflections. This is clearly visible in Panel (c) from Figure 8. Sound samples can be found at the following Github link[3].

6. CONCLUSION

This paper presented a physical model of the yaybahar in the modal domain. To this end, a modal decomposition of its subcomponents was offered, including a model for the coupling between a vibrating string and a distributed bridge, and the analytic modal expansion of a helical spring. In addition, an energy-consistent method for connecting the instrument components was here presented, making use of boundary forces.

This work focused on the development of a yaybahar physical model; nevertheless, different aspects were overlooked, and will be

[3] https://github.com/Nemus-Project/yaybahar-nit

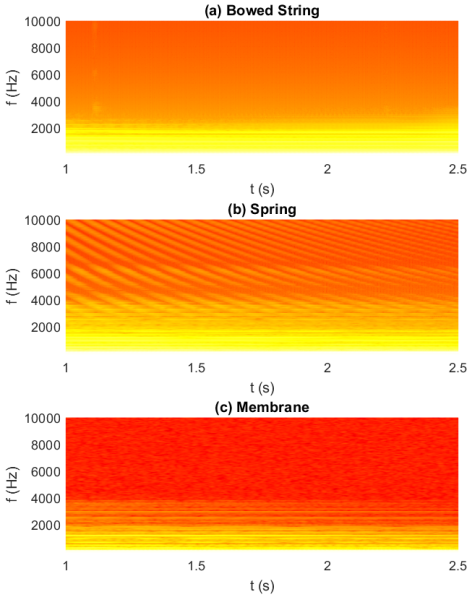


Figure 8: *Spectrograms of the three subsystems outputs after the initial transient. Panel (a) displays the spectrogram of the extracted force F_p, panel (b) shows the spectrogram of the force signal F_c, and panel (c) represents the output signal $\dot{w}_\mathrm{o}$ at the output position $(X_\mathrm{o}, Y_\mathrm{o})$.*

subject of future work. The physical values for the spring and the membrane were empirically tuned, while running proper measurements on a real instrument would significantly improve the sound quality. This is valid for the damping profiles as well. In addition, the membrane was considered to be rectangular, while an accurate reproduction would employ a circular model. Further future work will also include a real-time implementation.

7. ACKNOWLEDGMENTS

This work was supported by the European Research Council (ERC), under grant 2020-StG-950084-NEMUS.

8. REFERENCES

[1] J Parker and S. Bilbao, "Spring reverberation: A physical perspective," in *Proc. Digital Audio Effects (DAFx-09)*, Como, Italy, 09 2009.

[2] S. Bilbao, *Numerical Sound Synthesis*, John Wiley & Sons, Ltd, Chichester, UK, 2009.

[3] S. Bilbao, "Numerical simulation of spring reverberation," in *Proc. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, 09 2013.

[4] P.J. Christensen, S. Willemsen, and S. Serafin, "Applied Physical Modeling for Sound Synthesis: the Yaybahar," in *Proc. 2nd Nordic SMC Conf.*, Online, 11 2021.

[5] J. Parker, H. Penttinen, S. Bilbao, and J.S. Abel, "Modeling methods for the highly dispersive slinky spring: A novel musical toy," in *Proc. Digital Audio Effects (DAFx-10)*, Graz, Austria, 09 2010.

[6] Riccardo Russo, Michele Ducceschi, and Stefan Bilbao, "Efficient simulation of the bowed string in modal form," in *Proc. Digital Audio Effects (DAFx-2022), Vienna*, 09 2022.

[7] M. Ducceschi and S. Bilbao, "Non-iterative simulation methods for virtual analog modelling," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 30, pp. 3189–3198, 2022.

[8] M. Ducceschi and C.J. Webb, "Plate reverberation: Towards the development of a real-time plug-in for the working musician," in *Proc. Int. Conf. Acoust. (ICA 2016)*, Buenos Aires, Argentina, 09 2016.

[9] R. Russo, "Physical modeling and optimisation of a emt 140 plate reverb," M.S. thesis, Aalborg University Copenhagen, Copenhagen, Denmark, 2021.

[10] J.O. Smith, *Physical Audio Signal Processing*, `https://ccrma.stanford.edu/~jos/pasp/pasp.html`, accessed April 2023, online book, 2010 edition.

[11] J. Woodhouse, "On the "bridge hill" of the violin," *Acta Acust. united Acust.*, vol. 91, pp. 155–165, 2005.

[12] J.H. Smith and J. Woodhouse, "The tribology of rosin," *J. Mech. Phys. Solids*, vol. 48, pp. 1633–1681, 08 2000.

[13] P. Galluzzo, J. Woodhouse, and H. Mansour, "Assessing friction laws for simulating bowed-string motion," *Acta Acust. united Acust.*, vol. 103, pp. 1080–1099, 11 2017.

[14] C. Desvages, *Physical modelling of the bowed string and applications to sound synthesis*, Ph.D. thesis, The University of Edinburgh, 2018.

[15] W.H. Wittrick, "On elastic wave propagation in helical springs," *Int. J. Mech. Sci.*, vol. 8, no. 1, pp. 25–47, 1966.

[16] L. Della Pietra and S. della Valle, "On the dynamic behaviour of axially excited helical springs," *Meccanica*, vol. 17, pp. 31–43, 1982.

[17] J.N. Franklin, *Matrix Theory*, Dover Publications, Mineola, NY, USA, 2000.

[18] M. Van Walstijn, "Numerical calculation of modal spring reverb parameters," in *Proc. Digital Audio Effects (DAFx-2020), Online*, 09 2020, Proceedings of the 23rd International Conference on Digital Audio Effects.

[19] J. McQuillan and M. Van Walstijn, ," in *Proc. Digital Audio Effects (DAFx-2021), Online*, 09 2021, vol. 2.

[20] L.E. Kinsler, A.R. Frey, A.B. Coppens, and J.V. Sanders, *Fundamentals of Acoustics, 4th Edition*, Wiley, Hoboken, NJ, USA, 1999.

[21] J. Sherman and W.J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Ann. Math. Stat.*, vol. 21, pp. 124–127, 1950.

[22] R.J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations. Steady State and Time Dependent Problems*, SIAM, Philadelphia, USA, 2007.

[23] J. Cieslinski, "On the exact discretization of the classical harmonic oscillator equation," *J. Differ. Equ. Appl.*, vol. 17, no. 11, pp. 1673–1694, 2009.

[24] C. Valette and C. Cuesta, *Mécanique de la corde vibrante*, Hermès, Paris, 1993.

Oral Session 2:
Analysis & Synthesis

AN ALIASING-FREE HYBRID DIGITAL-ANALOG POLYPHONIC SYNTHESIZER

Jonas Roth, Domenic Keller, Oscar Castañeda, and Christoph Studer

Department of Information Technology and Electrical Engineering
ETH Zurich, Switzerland
joroth@ethz.ch | domkeller@ethz.ch | caoscar@ethz.ch | studer@ethz.ch

ABSTRACT

Analog subtractive synthesizers are generally considered to provide superior sound quality compared to digital emulations. However, analog circuitry requires calibration and suffers from aging, temperature instability, and limited flexibility in generating a wide variety of waveforms. Digital synthesis can mitigate many of these drawbacks, but generating arbitrary aliasing-free waveforms remains challenging. In this paper, we present the ±synth, a hybrid digital-analog eight-voice polyphonic synthesizer prototype that combines the best of both worlds. At the heart of the synthesizer is the big Fourier oscillator (BFO), a novel digital very-large scale integration (VLSI) design that utilizes additive synthesis to generate a wide variety of aliasing-free waveforms. Each BFO produces two voices, using four oscillators per voice. A single oscillator can generate up to 1024 freely configurable partials (harmonic or inharmonic), which are calculated using coordinate rotation digital computers (CORDICs). The BFOs were fabricated as 65 nm CMOS custom application-specific integrated circuits (ASICs), which are integrated in the ±synth to simultaneously generate up to 32 768 partials. Four 24-bit 96 kHz stereo DACs then convert the eight voices into the analog domain, followed by digitally controlled analog low-pass filtering and amplification. Measurement results of the ±synth prototype demonstrate high fidelity and low latency.

1. INTRODUCTION

Digital sound synthesis has many advantages over implementations with analog circuitry. Most notably, digital implementations are able to produce a wide variety of waveforms and do not suffer from temperature instabilities, aging, and component variations. However, aliasing is a ubiquitous nuisance in digital sound synthesis and specialized signal processing techniques are often necessary to combat such artifacts. For example, the work in [1] proposes low-complexity methods to generate aliasing-free waveforms of classical analog synthesizers (e.g., rectangle, sawtooth, and triangle). Nonetheless, this method is unable to generate more complex waveforms. In stark contrast, direct digital synthesis (DDS) [2] enables the generation of nearly arbitrary waveforms at very low complexity. Unfortunately, naïve DDS implementations generally suffer from aliasing. While aliasing can be reduced to a certain extent with oversampling followed by low-pass filtering, such an approach diminishes the complexity advantages of DDS. The work in [3] presents a method that is able to generate arbitrary alias-free single-period wavetable waveforms. This method, however, requires intricate trigonometric functions that must be calculated at high precision and is, thus, not well-suited for efficient software and hardware implementations.

An alternative sound-synthesis approach that eliminates aliasing altogether is to use additive synthesis [4], without ever generating partials that exceed half the sampling rate. Many software synthesizers support additive synthesis and benefit from the flexibility and user-interface capabilities that software brings. However, relying on general-purpose processors, such implementations have to balance signal quality and computational complexity, which limits the amount of partials that can be generated and affects their purity.[1] To overcome the limitations of additive synthesis software implementations, the work in [6] proposes a specialized hardware design, which is able to generate a large number of partials (up to 1200) with a single application-specific integrated circuit (ASIC). Such an implementation would enable the generation of a wide variety of complex and aliasing-free waveforms, but, to the best of our knowledge, no working system was demonstrated.

In recent years, a number of commercially available hybrid digital-analog instruments emerged, which can generate a broad range of high-quality waveforms. Specific instances are the Arturia Freak Series [7], Sequential Prophet X [8], Udo Super 6 [9], and Waldorf Quantum [10]. Unfortunately, only very little is known about the inner workings of these instruments and, thus, it remains largely a mystery how the waveforms are synthesized.

1.1. Contributions

We present the ±synth, an eight-voice polyphonic hybrid digital-analog synthesizer prototype that combines digital oscillators with analog filtering and amplification. Each voice consists of four digital oscillators, each able to generate a wide range of aliasing-free waveforms using an additive synthesis approach[2] with up to 1024 partials (harmonic, inharmonic, or subharmonic); in total, the instrument can generate 32 768 partials simultaneously. The oscillators are implemented using a custom ASIC, the *big Fourier oscillator (BFO)*, which generates two voices. To arrive at high hardware efficiency of the BFO ASIC implementation, we present a range of algorithm-level optimizations and a very-large scale integration (VLSI) architecture that utilizes CORDICs (short for coordinate rotation digital computers) to generate partials of high purity. We show how the BFO ASICs are integrated into the ±synth hardware prototype, including the analog section that incorporates voltage controlled filters (VCFs) and voltage controlled amplifiers (VCAs) based on commercial integrated circuits (ICs). Finally, we

[1] The maximum number of partials varies significantly across different plugins and depends on several parameters, such as the number of voices, signal quality, and others. Alchemy from Apple's Logic Pro X, for example, supports up to 600 partials [5], while others support less than a dozen.

[2] Thus the name ±*synth*, where + represents the additive synthesis approach and − the subtractive architecture of the instrument.

Figure 1: *Overview of the synthesizer setup. Top left: custom power supply; top middle: ±synth hardware prototype; top right: MIDI controller; bottom: MIDI keyboard.*

present implementation results of the BFO ASIC and measurement results at various output stages in the instrument. A photo of the ±synth hardware prototype, including the external MIDI keyboard and controller, is shown in Figure 1.

2. SYNTHESIZER ARCHITECTURE

Figure 2 shows a photo of the ±synth hardware prototype consisting of a main board (with audio, MIDI, and power connectors), an STM32 Nucleo development board (in white), with a custom printed circuit board (PCB) on top hosting four BFO ASICs, and four analog voice PCBs plugged into the main board (top right). The key components of the hardware design are discussed next.

2.1. System Overview

A system architecture overview of the ±synth is given in Figure 3. The instrument relies on a hybrid digital-analog version of subtractive synthesis with digital oscillators, analog filters, and analog amplifiers. The ±synth is digitally controlled by a microcontroller unit (MCU), which receives user inputs from an external MIDI keyboard and a universal MIDI controller with rotary encoders. The MCU generates all of the control signals for the digital oscillators as well as for the analog filters and amplifiers. The instrument is able to generate eight independent voices, where each BFO ASIC implements two voices and each voice consists of four digital oscillators. The digital voices are converted into the analog domain using four stereo DACs and each voice is separately processed by a VCF and a VCA; an *analog voice PCB* houses these analog components required for two voices. The control voltages (CVs) are generated using CV-DACs, which are digitally controlled by the MCU. The analog voices are summed to create the instrument's output signal, which can be either mono (eight voices) or stereo (four voices per channel); this is controlled by relays. The ±synth offers stereo line-level and headphone outputs. A custom power supply derives all of the required voltages for both the analog and digital domains from an off-the-shelf 24 V DC power supply.

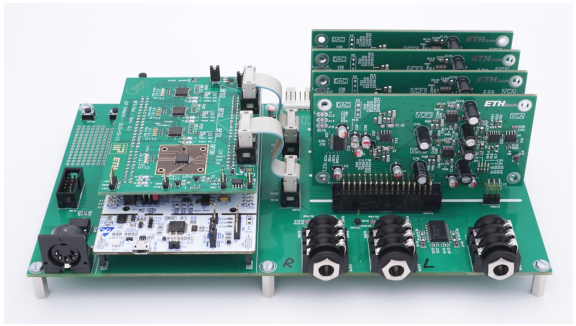


Figure 2: *Photo of the ±synth hardware prototype.*

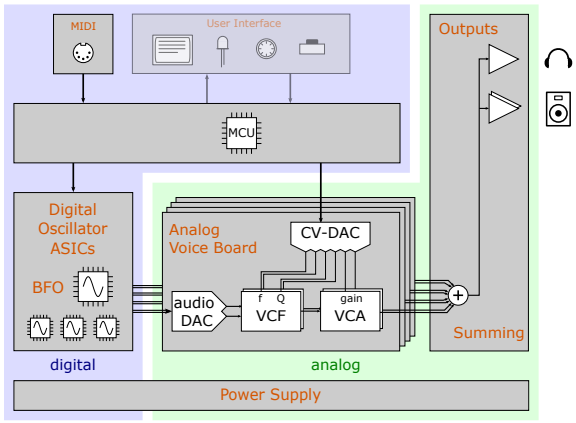


Figure 3: *Architecture overview of the ±synth.*

2.2. Digital Control and Interfaces

Digital control of the instrument is carried out on an STM23 Nucleo development board (STM32F722ZE), which was chosen to bypass the recent chip shortage. Received MIDI messages are used to compute the control signals for the audio path (e.g., oscillator pitch and volume, filter cutoff-frequency and resonance, and amplifier gain). The MCU also implements the attack, decay, sustain, and release (ADSR) envelope generators as well as low-frequency oscillators (LFOs). The MCU generates and transmits the control signals to the digital oscillators using a serial parallel interface (SPI) bus. The oscillators then generate eight voices that are converted to the analog domain via inter-IC sound (I2S) interfaces and four 24-bit stereo DACs. A second SPI bus on the MCU interfaces with the CV-DACs that control the analog voice processing paths. The details of the digital oscillators and analog voice PCBs are discussed next.

2.3. Aliasing-Free Digital Oscillator

At the heart of the ±synth are digital aliasing-free oscillators, which utilize additive synthesis, each generating up to $K = 1024$ partials.[3] Each voice is composed of four digital oscillators, which can be mixed together arbitrarily with configurable gains. Our custom ASIC, the BFO, implements two voices, and the ±synth consists of four BFO ASICs to provide eight-voice polyphony. Each oscillator

[3]With up to $K = 1024$ partials per oscillator, we can generate, for example, a sawtooth wave with a base frequency $f = 20$ Hz and the highest harmonic at 20 480 Hz, which is at the edge of the audible spectrum.

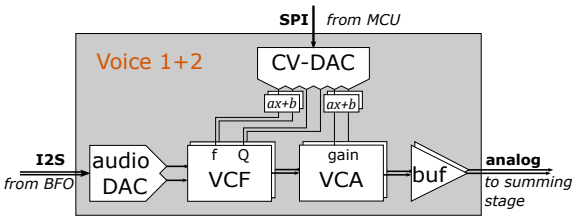


Figure 4: *Block diagram of an analog voice PCB.*

calculates its samples $x[\ell]$ using the following Fourier series:

$$x[\ell] = \sum_{k=1}^{K} a_k \cos\left(2\pi \frac{f n_k}{f_\mathrm{s}} \ell\right) + b_k \sin\left(2\pi \frac{f n_k}{f_\mathrm{s}} \ell\right). \quad (1)$$

The oscillator parameters $\{a_k, b_k, n_k\}_{k=1}^{K}$, together with the oscillator's base frequency f and the system's sampling rate f_s, fully determine the waveform to be generated. Note that each oscillator has its own set of parameters. In order to avoid aliasing altogether, we only sum the terms in (1) indexed by the set[4]

$$\mathcal{K}(f) = \{k = 1, \ldots, K : f n_k < f_\mathrm{s}/2\}, \quad (2)$$

i.e., we replace $k = 1, \ldots, K$ with $k \in \mathcal{K}(f)$ in (1). Section 3 details how this additive synthesis approach is implemented in our custom BFO ASICs.

Remark 1. *We emphasize that (1) is, strictly speaking, not a Fourier series, as we also allow the multipliers $n_k \in \mathbb{Q}$, $k = 1, \ldots, K$, to be nonnegative rational numbers represented in the chosen fixed-point format (see Section 3.1 for the details). This flexibility enables us to generate waveforms with harmonics, inharmonics, and subharmonics, which implies that a single oscillator cannot only generate standard analog synthesizer waveforms, but also arbitrary wavetable sounds or bell-like timbres.*

Each of the four oscillators of a voice generates samples according to (1), which are weighted and summed. Then, the two BFO voices can be further mixed before being passed to the I2S output. Details on the mixing stage are provided in Section 3.5.

2.4. Analog Voice Boards

The $\pm$synth features four analog voice PCBs, which can be seen at the top-right of Figure 2. Each of these PCBs carries out analog signal processing for two voices; a block diagram is depicted in Figure 4. The audio DAC receives the samples from the two voices generated by a BFO ASIC at a sampling rate of 96 kHz. We used a CS4350 24-bit stereo DAC from Cirrus Logic [11], which contains an integrated phase-locked loop (PLL) that derives its master clock from the I2S clock. This eliminates the need to route a separate 25 MHz clock signal to each analog voice PCB.

The VCFs are implemented using the SSI2144 IC from Sound Semiconductor [12], which implements a fourth-order low-pass ladder filter. The cutoff frequency and resonance (Q-factor) are set by CVs. The VCAs are implemented using the SSI2162 IC from Sound Semiconductor [13], which is used to apply the envelope determined by another CV. Finally, the analog voice signal is buffered before leaving the PCB to the final summing stage on the main board that produces headphone and line outputs. Note that some CVs are low-pass filtered, scaled, and offset in order to avoid crosstalk to the audio path and to match the required voltage range; this is implemented using operational amplifiers.

[4] We note that aliasing can still occur if one reconfigures the oscillator parameters $\{a_k, b_k, n_k\}_{k=1}^{K}$ at too fast rates.

3. VLSI DESIGN OF THE BIG FOURIER OSCILLATOR

In order to develop an efficient VLSI design that is able to implement multiple aliasing-free oscillators, each with a large number of high-quality partials, we leverage a range of algorithm- and hardware-level tricks which are discussed next.

3.1. Fixed-Point Arithmetic

Our VLSI design exclusively uses fixed-point arithmetic, mostly with 32-bit fixed-point precision, which enables the generation of extremely pure partials with low harmonic distortion at high hardware efficiency. The fixed-point number format is designated by the notation $\{\square, q_\mathrm{i}, q_\mathrm{f}\}$, where q_i is the number of integer bits, q_f is the number of fractional bits, and $\square$ is either s or u if the fixed-point number is signed or unsigned, respectively. The coefficients a_k and b_k in (1) use the format $\{s, 0, 31\}$, which gives precise control over the partials' amplitudes and phases. The multipliers n_k use the format $\{u, 16, 16\}$, which enables harmonic, inharmonic, and subharmonic partials with fine frequency resolution. The base frequency is normalized as f/f_s and uses the format $\{u, 0, 32\}$, which results in a frequency resolution of 22.4 µHz at $f_\mathrm{s} = 96$ kHz. While the generated samples have an internal resolution of 32 bits, the I2S output is reduced to the DACs' resolution of 24 bits.

3.2. Algorithm-Level Optimizations

To improve the hardware efficiency of our VLSI design, we use the following algorithm-level optimizations that reduce the complexity of calculating the samples as in (1).

Reparametrization from Radians to Turns Instead of directly computing the arguments $\phi_k[\ell] \triangleq 2\pi \frac{f n_k}{f_\mathrm{s}} \ell$ of the cosine and sine functions in (1), which are in the unit of radians, our VLSI design calculates the two functions $\mathrm{co}(\theta) \triangleq \cos(2\pi\theta)$ and $\mathrm{si}(\theta) \triangleq \sin(2\pi\theta)$ instead. Here, the arguments $\theta_k[\ell] \triangleq \frac{f n_k}{f_\mathrm{s}} \ell$ are represented in what is known as *turns*, which has several advantages. First, the arguments $\theta_k[\ell]$ are in the range $[0, 1)$, which requires one to pass arguments to the $\mathrm{co}(\cdot)$ and $\mathrm{si}(\cdot)$ functions of the form

$$\theta_k[\ell] = \frac{f n_k}{f_\mathrm{s}} \ell \bmod 1. \quad (3)$$

The modulo-1 operation can be obtained in hardware for free by simply discarding the integer part of $\theta_k[\ell]$ when represented by unsigned fixed-point numbers. Second, one can directly calculate the functions $\mathrm{co}(\cdot)$ and $\mathrm{si}(\cdot)$ in a hardware-friendly manner using CORDICs; see Section 3.3 for the details.

Sequential Calculation of Arguments For each sample $\ell = 0, 1, \ldots$ and partial $k = 1, \ldots, K$, two multiplications are required to calculate the argument $\theta_k[\ell]$ in (3). First, the argument increment $\delta_k \triangleq f n_k / f_\mathrm{s}$ is computed by multiplying the normalized base frequency f/f_s by the frequency multiplier n_k. Second, the argument increment δ_k is multiplied by the sample index ℓ. While this last multiplication occurs in modulo-1 arithmetic, it still requires

a high dynamic range, as the sample index ℓ is represented with a large number of bits to avoid unwanted resetting in the oscillators. Hence, every sample ℓ requires $K = 1024$ of these high-resolution $\theta_k[\ell] = \delta_k \ell \bmod 1$ products. This complexity could easily be reduced by tracking the argument θ_k for each partial $k = 1, \dots, K$ with an addition instead of a multiplication. In specific, one can update the argument θ_k for every new sample ℓ as

$$\theta_k \leftarrow (\theta_k + \delta_k) \bmod 1. \tag{4}$$

The arguments are initialized as $\theta_k = 0$ at sample index $\ell = 0$. We reiterate that the modulo-1 operation is free in hardware by discarding the integer part of the arguments θ_k, $k = 1, \dots, K$.

Single Argument for All Partials The remaining disadvantage of the above method is that one must keep track of the arguments θ_k for every partial $k = 1, \dots, K$. This requires additional storage for all arguments in a two-port memory that supports one simultaneous read and write per update of (4). To avoid an additional memory, we keep track of a *single* base-frequency argument $\theta \triangleq \frac{f}{f_\mathrm{s}} \ell$ from which all arguments θ_k can be derived as follows:

$$\theta_k = \theta n_k \bmod 1, \quad k = 1, \dots, K. \tag{5}$$

Unfortunately, unlike the arguments θ_k in (4), the base-frequency argument θ cannot be accumulated modulo-1 since the property

$$\theta n_k \bmod 1 = (\theta \bmod m) n_k \bmod 1 \tag{6}$$

with $m = 1$ does *not* hold for every n_k. For example, for $\theta = 1$ and $n_k = 1.5$, $(\theta \bmod 1) n_k \bmod 1 = 0$ is different from the desired $\theta n_k \bmod 1 = 0.5$. Indeed, to calculate θ_k for an arbitrary frequency multiplier n_k from the base-frequency argument θ, the quantity θ needs to be represented with an infinite number of integer bits (i.e., without using a modulo operation). Nevertheless, provided that the multipliers n_k are represented with a finite number of fractional bits q_f, it is sufficient to represent θ using a finite number of integer bits. This key insight is made rigorous by Lemma 1.

Lemma 1. *$\theta n \bmod l = (\theta \bmod m) n \bmod l$ if $mn \bmod l = 0$.*

Proof. Let $\Theta = \theta \bmod m$, so that we want to show $\theta n \bmod l = \Theta n \bmod l$. We rewrite $\Theta = \theta \bmod m$ and $mn \bmod l = 0$ as

$$\theta = am + \Theta, \text{ and} \tag{7}$$

$$mn = bl, \tag{8}$$

where $a, b \in \mathbb{Z}$. Multiplying both sides of (7) by n, we get

$$\theta n = amn + \Theta n = abl + \Theta n, \tag{9}$$

where (9) follows from (8). Since $ab \in \mathbb{Z}$, taking the modulo-l of both sides of (9) results in $\theta n \bmod l = \Theta n \bmod l$. □

By applying Lemma 1 with $n = n_k$ and $l = 1$, we can determine the value of m so that (6) is satisfied for the multipliers n_k used in the BFO. In words, Lemma 1 is telling us that, instead of keeping track of θ with infinite precision, we can just keep track of its modulo-m equivalent and any multiplication by the frequency multipliers n_k will be correct in modulo-1 arithmetic as long as $mn_k \in \mathbb{Z}$. Given that n_k has $q_\mathrm{f} = 16$ fractional bits, we can satisfy this last requirement by setting $m = 2^{q_\mathrm{f}}$. Therefore, we keep track of the base-frequency argument θ with the recursion

$$\theta \leftarrow (\theta + \delta) \bmod 2^{q_\mathrm{f}}, \tag{10}$$

where $\delta = f / f_\mathrm{s}$. The modulo-$2^{q_\mathrm{f}}$ operation is easily implemented in hardware by using only q_f bits to represent the integer part of θ and letting the result wrap-around once the maximum representable number is reached. We note that, if all $n_k \in \mathbb{Z}$, then θ could be tracked in modulo-1 arithmetic—thus, the recursion in (10) is required as we also support inharmonic and subharmonic partials.

We observe that, while this approach requires $K = 1024$ multiplications per sample ℓ as in (5), it avoids (i) storing arguments per partial and (ii) additional $K = 1024$ multiplications per sample that would be required to compute the frequency increments $\delta_k = \delta n_k$. Thus, our approach leverages a trade-off between the high complexity of naïvely computing $\theta_k[\ell]$ as in (3) and the large memory overhead of a per-θ_k-argument accumulation as in (4).

3.3. Computing Cosines and Sines with CORDICs

To calculate cosine and sine functions at high precision and in a hardware-friendly way, we utilize CORDICs [14], which essentially calculate two-dimensional Givens rotations of the following form:

$$\underbrace{\begin{bmatrix} p_1' \\ p_2' \end{bmatrix}}_{\mathbf{p}'} = \underbrace{\begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}}_{=\mathbf{G}(\phi)} \underbrace{\begin{bmatrix} p_1 \\ p_2 \end{bmatrix}}_{=\mathbf{p}}. \tag{11}$$

Evidently, by setting $\phi = 2\pi \frac{f n_k}{f_\mathrm{s}} \ell$, $p_1 = a_k$, and $p_2 = -b_k$, the output p_1' is exactly one term of the Fourier series in (1).

We now outline the idea behind CORDICs—the interested reader is referred to [15] for more details. First, one approximates the desired rotation angle $\phi \approx \sum_{m=0}^{M-1} \phi_m$ by summing M predefined micro-rotation angles ϕ_m, $m = 0, \dots, M-1$; see below for a concrete choice of these angles. Second, the Givens rotation in (11) is approximated by $\mathbf{G}(\phi) \approx \prod_{m=0}^{M-1} \mathbf{G}(\phi_m)$ with M so-called micro-rotations $\mathbf{G}(\phi_m)$, which are simply Givens rotations by the angles ϕ_m. Third, one rewrites each micro-rotation as

$$\mathbf{G}(\phi_m) = \kappa_m \begin{bmatrix} 1 & -\tan(\phi_m) \\ \tan(\phi_m) & 1 \end{bmatrix}, \tag{12}$$

where $\kappa_m = (1 + \tan^2(\phi_m))^{-\frac{1}{2}}$. Fourth, one restricts the micro-rotation angles ϕ_m to $\tan(\phi_m) = d_m 2^{-m}$ with $d_m \in \{-1, +1\}$. With this, the Givens rotation in (11) is approximated as

$$\mathbf{p}' \approx \kappa \prod_{m=0}^{M-1} \begin{bmatrix} 1 & -d_m 2^{-m} \\ d_m 2^{-m} & 1 \end{bmatrix} \mathbf{p}. \tag{13}$$

Here, the scaling factor $\kappa = \prod_{m=0}^{M-1} \kappa_m$ depends only on the number of micro-rotations M and not on the choices of d_m. Fifth, one needs to determine the micro-rotation angles ϕ_m that well-approximate the target angle ϕ. The standard procedure iteratively determines ϕ_m from the target angle ϕ, i.e., by first taking the angle $\phi_0 = d_0 \mathrm{atan}(2^{-0})$ that brings ϕ closer to zero. One then updates the target angle as $\phi \leftarrow \phi - d_0 \mathrm{atan}(2^{-0})$ and takes a new angle $\phi_1 = d_1 \mathrm{atan}(2^{-1})$ that brings the updated ϕ closer to zero. This procedure is repeated for the remaining $M - 2$ micro-rotations.

We note that every additional micro-rotation provides roughly one additional bit of precision [15]; this implies that the approximation error in (13) can be made arbitrarily small. In our application, we use a quadrant correction followed by $M = 26$ micro-rotations, which leads to a precision of $q_\mathrm{f} = 24$ fraction bits (see Section 4 for measurements). Furthermore, instead of representing the micro-rotation angles ϕ_m in radians, we represent them in turns; this

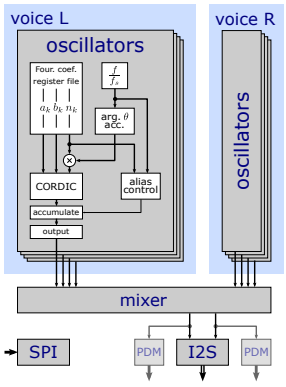


Figure 5: *BFO architecture.*

Figure 6: *ASIC micrograph.*

enables us to directly calculate the functions $\mathrm{co}(\cdot)$ and $\mathrm{si}(\cdot)$ with a CORDIC. Finally, it is crucial to realize that each micro-rotation in (13) only involves shifts, additions, subtractions, and a multiplication by the constant κ; this implies that the Givens rotation in (11) can be approximated in a hardware-friendly manner, generating samples of cosine and sine functions with extremely high purity.

3.4. VLSI Architecture

Figure 5 depicts the VLSI architecture of our BFO ASIC, which consists of two voices, referred to as L and R, each one with four oscillators. Each oscillator features a 1024×96-bit register file to store the oscillator coefficients $\{a_k, b_k, n_k\}$, $k = 1, \ldots, 1024$, an argument accumulator, a fully unrolled CORDIC module that calculates one pair of sine and cosine values per clock cycle, and a sample accumulator. The configuration registers and register files of the BFO are addressed using a memory map and are configured via SPI. Each SPI command uses 48 bits: 16 bits for the address and 32 bits for the data. The SPI interface runs at a baud rate of 13 Mb/s, with which a new waveform with $K = 1024$ partials could be reprogrammed in less than 12 ms—nevertheless, this task is currently completed in 315 ms as our firmware does not yet fully exploit the MCU's capabilities.

To generate samples $x[\ell]$ as in (1) with $K = 1024$ partials at a sampling rate $f_\mathrm{s} = 96\,\mathrm{kHz}$, each oscillator operates at a clock frequency of $Kf_\mathrm{s} = 98.304\,\mathrm{MHz}$ and calculates one sample of one partial every clock cycle. The argument accumulator uses the base frequency f/f_s, stored in a configuration register, to track the base-frequency argument θ, which is updated every $K = 1024$ clock cycles as in (10). In each of the K clock cycles that the base-frequency argument θ remains fixed, one word of the register file is read to obtain $\{a_k, b_k, n_k\}$. Then, the argument θ_k is computed by multiplying θ and n_k as in (5). With $\{a_k, b_k, \theta_k\}$ available, the CORDIC computes one partial in (1), which is then accumulated to the current sample $x[\ell]$ if aliasing will not occur, i.e., if $fn_k < f_\mathrm{s}/2$ is met. After $K = 1024$ clock cycles, the four samples generated by the four oscillators are added to create one voice sample, and the L and R voice samples are then mixed using a programmable 2×2 matrix to support, e.g., stereo and mono processing. The two mixed samples are then streamed to the DACs via I2S.

Figure 6 shows a micrograph of the $3\,\mathrm{mm}^2$ BFO ASIC, which was fabricated in TSMC 65 nm LP CMOS technology. At the nominal 1.2 V core supply and room temperature, the ASIC achieves a maximum measured clock frequency of 154 MHz, exceeding the required 98.304 MHz, while consuming only 178 mW.

3.5. Bells and Whistles

The BFO includes a number of additional features, which further improve its versatility and flexibility. These features are as follows.

Subwave Mixing Per default, each oscillator accumulates 1024 partials together to compute one sample $x[\ell]$ as in (1). We also support a *subwave mixing* mode, in which the 1024 partials are split into up to four disjoint groups (or *subwaves*) that are accumulated independently. By doing so, a single oscillator can blend various wavetable sounds with the same base frequency, each one with fewer partials; e.g., an oscillator can generate four subwaves of 256 partials each instead of a single waveform with 1024 partials. Each subwave $x_i[\ell]$, $i = 1, \ldots, 4$, has a corresponding weight v_i, so that the output of one oscillator is $y[\ell] = \sum_{i=1}^{4} v_i x_i[\ell]$. Thus, with the four oscillators per voice, subwave mixing can arbitrarily blend up to 16 different wavetables in a single voice.

Aliasing Control Per default, only partials for which $fn_k/f_\mathrm{s} < 0.5$ holds are accumulated; see (2). The BFO includes a mode that accumulates partials for which the following condition is met

$$f_\mathrm{HP} \leq fn_k/f_\mathrm{s} < f_\mathrm{LP}, \tag{14}$$

which realizes an ideal band-pass filter with the lower and upper cutoff frequencies f_HP and f_LP, respectively. These frequencies can be configured in the range $[0, 1)$; the default values are $f_\mathrm{HP} = 0$ and $f_\mathrm{LP} = 0.5$ (no aliasing allowed). Having these frequencies programmable can be used, e.g., to intentionally allow for aliasing (by setting $f_\mathrm{LP} > 0.5$), or to apply low- and high-pass filtering.

Bit- and Rate-Crusher Each oscillator output includes (optional) *bit-crusher* and *rate-crusher* distortion effects. The bit-crusher forces certain bits of the output sample $y[\ell]$ to zero; the zero bits are determined by a programmable bit-mask. This feature can be used to reduce the bit-resolution by zeroing a certain number of least-significant bits; other, more complicated masking patterns are also possible. The rate-crusher is a simple sample-and-hold sub-sampling circuit that lowers the rate at which the output samples $y[\ell]$ are updated. The sub-sampling rate is set in the range $(0, f_\mathrm{s})$ and is stored in a configuration register. This feature can be used to emulate lower sampling rates or, for example, cause intriguing aliasing artifacts that depend on the base frequency f.

PDM Output As an alternative to the I2S interface, each BFO ASIC also includes two pulse density modulation (PDM) outputs. The 1-bit PDM signal is generated from a digital first-order sigma-delta modulator running at an oversampling rate of 1024, which corresponds to the BFO's clock frequency. This output could be useful for cost-sensitive applications (as no DACs are required), but at reduced sound quality; see Section 4.1 for measurement results. Note that the PDM outputs are not used in the ±synth prototype.

Clipping Indicators At certain stages in the digital signal processing path, sample values must be clipped to a certain maximum range to reduce their word lengths. This occurs at the output of each oscillator and after the mixer stage. Thus, we included a number of clipping indicator outputs that are raised if clipping occurred; these are tied to LEDs on the ±synth prototype.

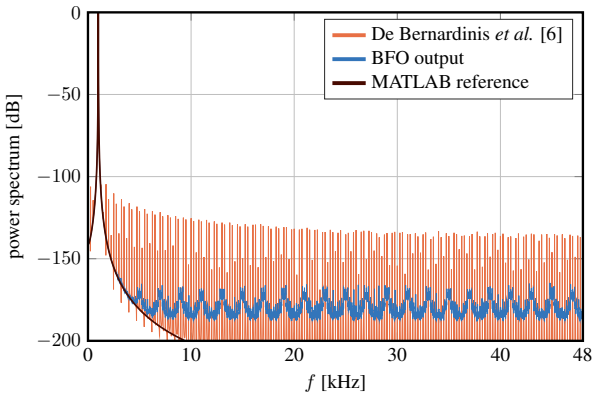


Figure 7: *Power spectrum comparison for a* 1 *kHz sine wave at a sampling rate of* $f_s = 96$ *kHz between a MATLAB floating-point reference, the BFO output, and the method described in [6].*

3.6. Comparison

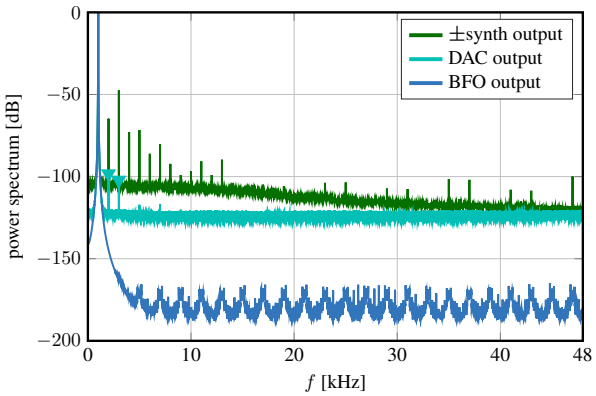
To the best of our knowledge, only the ASIC design reported in [6] is comparable to our BFO implementation. The design in [6] utilizes a marginally stable infinite-impulse response (IIR) filter to recursively generate samples of a single sinusoid only with the help of a multiply-accumulate unit. While the algorithm itself is competitive to our approach in complexity per generated sample, the recursive calculations together with fixed-point arithmetic suffer from error propagation, particularly at low frequencies (in the order of tens of Hz). To mitigate this issue, the recursion must be restarted periodically (the authors recommend restarting every 128 samples), which requires one to retrieve two sinusoids from two consecutive sample instants from a memory or a CORDIC. The design in [6] assumes that these initial sine values are generated externally. Moreover, their circuit assumes that the magnitudes and phases are stored externally and streamed into the ASIC. Thus, their design would require additional (external) logic and memory, whereas our BFO ASIC is fully self-contained.

A direct comparison between the hardware implementation characteristics of our BFO ASIC and the design in [6] is challenging due to missing details. Nonetheless, we re-implemented their method in MATLAB using the fixed-point parameters of [6] and compared it to a double-precision floating-point MATLAB reference and the BFO output for a 1 kHz sine at $f_s = 96$ kHz. Our simulations reveal that the method in [6] achieves a total harmonic distortion plus noise (THD+N) of -94.27 dB, which is 42.7 dB worse than what is achieved by our BFO ASIC (see also Table 1). The corresponding power spectra[5] are shown in Figure 7 and it is evident that our CORDIC-based approach generates sine waves with significantly higher purity than the ASIC design reported in [6].

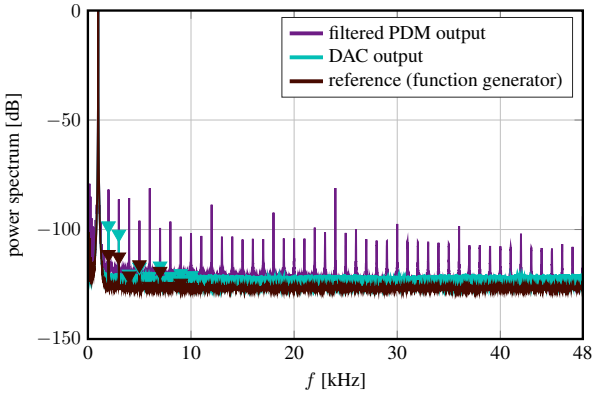
4. MEASUREMENT RESULTS

In order to quantify the performance of the ±synth prototype, we now present a number of measurement results.

[5]We analyze 960 k samples (10 s) using Welch's method with a Hann window, a 2^{14}-point FFT, 50% overlap, and a normalized peak value of 1.



(a) *Power spectra at different stages in the* ±*synth.*



(b) *Power spectra of reference, DAC, and filtered PDM outputs.*

Figure 8: *Measured power spectra (in decibels) for a* 1 *kHz sine wave at a sampling rate of* $f_s = 96$ *kHz.*

4.1. Signal Quality

We first assess the quality of a single partial at different stages of the instrument: (i) the BFO output (I2S output, digital domain), (ii) the DAC output (after reconstruction filter, analog domain), and (iii) the ±synth output (line-level output, analog domain). We generate a 1 kHz sine wave with -6 dBFS amplitude (with respect to the I2S interface) in the BFO at a sampling rate f_s of 96 kHz. The digital BFO output is captured using a logic analyzer to extract raw I2S data; the analog signals are captured using a Focusrite Scarlett 18i20 (3rd gen.) audio interface [16] with a sampling rate of 96 kHz. Figure 8(a) shows the power spectrum of the three signals. The corresponding THD+N values are reported in Table 1. We see that the test signal (sinusoidal) at the BFO output has extremely high purity and the quality is essentially limited by the DAC. We can also see that analog processing through the VCF and VCA circuitry further reduces the THD+N, which is not unexpected.

We also measured the filtered PDM output, which consists of a passive second-order low-pass filter with a -3 dB frequency of 31 kHz. This output achieves a THD+N of -75 dB, which is 13 dB higher than that of the DAC output (see Table 1); the associated spectrum is shown in Figure 8(b). Clearly, the PDM output is inferior to the DAC output, but would enable the use of our BFO ASICs with less expensive external circuitry.

Table 1: *THD+N measurements for a $f = 1\,kHz$ sine.*

Measurement	THD+N [dB]
$\pm$synth output (analog)	-51.0
PDM output (analog)	-75.0
DAC output (analog)	-88.2
BFO output (digital)	-137.0
reference: function generator (analog)	-89.5

Table 2: *SINAD for different waveforms at $f = 20\,Hz$.*

Waveform	SINAD [dB]
sine	134.0
triangle	133.3
sawtooth	135.3
super-saw	131.3
rect-saw	131.0
pulse	109.4

Remark 2. *The audio interface [16] specifies a THD+N below 0.002% ($\approx -94\,dB$) for the line inputs. As a reference, we measured a $1\,kHz$ sine wave generated with an SRS DS360 ultra-low distortion function generator [17], which resulted in a THD+N or $-89.5\,dB$ (cf. Table 1); the associated spectrum is shown in Figure 8(b). Since this THD+N result is close to that of the DAC output, our measurements are likely affected by the audio interface.*

Table 2 shows the signal-to-noise and distortion ratio (SINAD) between a MATLAB floating-point model and the digital BFO output for different waveforms generated at a base frequency of $f = 20\,\text{Hz}$. Most waveforms achieve a SINAD exceeding $131\,\text{dB}$ except for the pulse waveform, which yields $109.4\,\text{dB}$. The reason is that since the pulse waveform has the a_k-coefficients of all 1024 partials set to the same value, that value has to be reduced substantially to avoid clipping. Thus, the signal power of this waveform is much lower compared to the other waveforms, which results in lower SINAD.

4.2. Latency

As any digitally controlled instrument, the $\pm$synth exhibits non-negligible latency between a keystroke (or parameter change) and a change in the synthesizer's output. To assess the prototype's latency, we measure the delay between the reception of a note-on MIDI message at the $\pm$synth and the change in signal at the line-level output; this ignores external delays, e.g., caused by the MIDI keyboard. The measurements are repeated 34 times using an oscilloscope probing two signals: (i) the opto-coupler's output in the MIDI receiver circuit and (ii) the hot signal of the line-output. Our measurements show a mean latency of $2.09\,\text{ms}$ (minimum $1.60\,\text{ms}$; maximum $2.76\,\text{ms}$; standard deviation $0.33\,\text{ms}$). The latency is mainly caused by the MCU firmware, i.e., processing the UART data (MIDI receiver), computing new control signals, transmitting parameter data over SPI, etc.

4.3. Power Consumption

The $\pm$synth's power consumption is measured for two cases: (i) idle mode (default state after power-up) and (ii) playback (all voices playing). The supply current is measured at $499\,\text{mA}$ during idle mode and up to $522\,\text{mA}$ during playback. This corresponds to a power consumption of approximately $12\,\text{W}$ to $13\,\text{W}$. Since there are many factors influencing the instrument's power (e.g., filter resonance, load at audio outputs, etc.), these measurements should be taken with a grain of salt.

5. LIMITATIONS AND FUTURE WORK

The $\pm$synth prototype, in its current form, has a number of limitations, which we now summarize. First, the BFO ASICs are currently unable to perform oscillator synchronization, which is a direct consequence of the additive-synthesis approach discussed in Section 2.3. To mitigate this limitation, one could load in oscillator parameters $\{a_k, b_k, n_k\}_{k=1}^{K}$ of a synchronized oscillator, but this approach is limited by the rate at which all of the parameters can be rewritten (see Section 3.4). Also, such a workaround might no longer be aliasing-free. Developing hardware-friendly solutions to implement true oscillator synchronization without aliasing, e.g., inspired by the works of [18, 19], is part of ongoing work. Second, the BFO ASICs do not provide hardware support for true additive synthesis with separate envelopes per partial. Such functionality could readily be implemented in hardware, but comes at the cost of additional memory and logic to store and update the envelope parameters. A potential compromise would be to use the linear updates put forward in [6]. Third, the BFO does not provide a noise generator. We are planning to include such missing functionality in a future version. Fourth, the quality of the BFO is currently limited by the used 24-bit DAC, and our audio interface may affect our measurements; in the future, we will use a better DAC to fully exploit the BFO's high purity and also use better measurement equipment. Fifth, we are currently using an off-the-shelf external keyboard and controller—developing a dedicated user interface for the $\pm$synth would be quite exciting.

6. CONCLUSIONS

We have shown the implementation details of the $\pm$synth, an eight-voice hybrid digital-analog music synthesizer prototype that uses aliasing-free digital oscillators followed by analog filtering and amplification. By implementing the oscillators on custom ASICs, we are able to generate a wide variety of waveforms with up to 1024 freely programmable partials per oscillator, which includes not only classical waveforms of analog synthesizers, but also wavetable sounds or bell-like timbres. The $\pm$synth is able to generate a total number of $32\,768$ partials at a sampling rate of $96\,\text{kHz}$, and measurement results have demonstrated high fidelity and low latency.

While a range of commercial hybrid digital-analog synthesizers became available recently, virtually nothing is known about their inner workings. In contrast, every implementation detail of our hardware prototype is known and well-documented, and every aspect of the instrument can be modified easily. We therefore believe that the $\pm$synth will be an excellent research platform for future real-time audio synthesis experiments.

7. REFERENCES

[1] Timothy Stilson and Julius Smith, "Alias-free digital synthesis of classic analog waveforms," in *Proceedings of the International Computer Music Conference (ICMC)*, August 1996, pp. 332–335.

[2] Victor S. Reinhardt, "Direct digital synthesizers," in *Proceedings of the Annual Precise Time and Time Interval Systems and Applications Meeting*, December 1985, pp. 345–374.

[3] Thomas Schanze, "Sinc interpolation of discrete periodic signals," *IEEE Transactions on Signal Processing*, vol. 43, no. 6, pp. 1502–1503, June 1995.

[4] James A. Moorer, "Signal processing aspects of computer music: A survey," *Proceedings of the IEEE*, vol. 65, no. 8, pp. 1108–1137, August 1977.

[5] Apple Inc., "Alchemy additive element controls in Logic Pro," *Online*, available at https://support.apple.com/guide/logicpro/additive-element-controls-lgsi55ccfb06/10.7.5/mac/12.3, accessed: April 6th, 2023.

[6] Fernando De Bernardinis, Roberto Roncella, Roberto Saletti, Pierangelo Terreni, and Graziano Bertini, "An efficient VLSI architecture for real-time additive synthesis of musical signals," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 1, pp. 105–110, March 1999.

[7] Arturia, "Arturia - Freak," *Online*, available at https://www.arturia.com/ranges/freak, accessed: April 6th, 2023.

[8] Sequential LLC, "Prophet X," *Online*, available at https://www.sequential.com/product/prophet-x, accessed: April 6th, 2023.

[9] UDO Audio, "The super 6," *Online*, available at https://www.udo-audio.com, accessed: April 6th, 2023.

[10] Waldorf Music, "Quantum," *Online*, available at https://waldorfmusic.com/quantum-en, accessed: April 6th, 2023.

[11] Cirrus Logic, *192-kHz Stereo DAC with Integrated PLL*, August 2018.

[12] Sound Semiconductor Inc., *SSI2144: FATKEYS™ Four-Pole Voltage Controlled Filter*, January 2018, Rev. 3.0.

[13] Sound Semiconductor Inc., *SSI2162: FATKEYS™ Dual Voltage Controlled Amplifier*, June 2020, Rev. 1.1.

[14] Jack E. Volder, "The CORDIC trigonometric computing technique," vol. EC-8, no. 3, pp. 330–334, September 1959.

[15] Behrooz Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, Inc., USA, 1999.

[16] Focusrite, "Scarlett 18i20, 3rd generation," *Online*, available at https://focusrite.com/en/usb-audio-interface/scarlett/scarlett-18i20, section: "Specifications in Detail," accessed: April 6th, 2023.

[17] Stanford Research Systems, "DS360 — ultra-low distortion function generator," *Online*, available at https://www.thinksrs.com/downloads/pdfs/catalog/DS360c.pdf, accessed: April 6th, 2023.

[18] Eli Brandt, "Hard sync without aliasing," in *Proceedings of the International Computer Music Conference (ICMC)*, September 2001.

[19] Pier Paolo La Pastina and Stefano D'Angelo, "A general antialising method for sine hard sync," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, September 2022, pp. 109–114.

A COUPLED RESONANT FILTER BANK FOR THE SOUND SYNTHESIS OF NONLINEAR SOURCES

Samuel Poirot and Richard Kronland-Martinet *

PRISM
Aix Marseille Univ, CNRS, PRISM
Marseille, France
poirot@prism.cnrs.fr

Stefan Bilbao †

Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
s.bilbao@ed.ac.uk

ABSTRACT

This paper is concerned with the design of efficient and controllable filters for sound synthesis purposes, in the context of the generation of sounds radiated by nonlinear sources. These filters are coupled and generate tonal components in an interdependent way, and are intended to emulate realistic perceptually salient effects in musical instruments in an efficient manner. Control of energy transfer between the filters is realized by defining a matrix containing the coupling terms. The generation of prototypical sounds corresponding to nonlinear sources with the filter bank is presented. In particular, examples are proposed to generate sounds corresponding to impacts on thin structures and to the perturbation of the vibration of objects when it collides with an other object. The different sound examples presented in the paper and available for listening on the accompanying site tend to show that a simple control of the input parameters allows to generate sounds whose evocation is coherent, and that the addition of random processes allows to significantly improve the realism of the generated sounds.

1. INTRODUCTION

Modal synthesis operates according to the decomposition of the complex dynamic behavior of a vibrating object into contributions from modes, each oscillating independently at a single frequency. This approach, applicable to linear and time-invariant systems, is widely used and forms the basis for various physical modelling synthesis software packages [1] [2] and is closely related to sound synthesis methodologies employing filter banks [3] [4] [5].

For vibrating objects incorporating nonlinear effects, the modal interpretation must be generalized to include energy transfer between different modes (among other things such as e.g. frequency shifting of modes over time). It may cause the delayed and sustained appearance of tonal components that cannot be generated by linear filtering. This complex phenomenon, widely studied for the typical case of thin plates and shells [6] [7], can be modelled and solved under certain conditions. The numerical solution of the Föppl-von Kármán system [8, 9] that governs the underlying dynamics of nonlinear thin plates at moderate vibration amplitudes yields realistic and convincing sound synthesis [10], but at heavy computational cost. Ducceschi and Touzé [11] propose the modal resolution of the system with the offline calculation of coupling coefficients. They manage under certain approximations to significantly reduce the computation time without being able to achieve real-time sound synthesis (about 8 times real-time on a CPU) [12]. Another typical case of coupling between modes induced by nonlinear phenomena concerns collisions in musical instruments [13] and has been the subject of various studies, including on modal interactions [14]. Computational cost for synthesis can also be heavy in such cases.

For synthesis purposes, and particularly if real-time performance is the ultimate aim, it can be useful to depart from strict physical models, and examine modal interactions from a perceptual point of view. Skare and Abel [15] perform real-time modal synthesis of crash cymbals with a GPU-accelerated modal filterbank. Their method consists in identifying the modal parameters (including a rough approximation of the couplings) on recorded sounds, although the energy transfer mechanism is unspecified.

In this paper, we design coupled filters based on the design proposed by Mathews and Smith [16] and adapted by Skare and Abel [15] to incorporate energy transfer. In particular, we propose an equivalence between the power of the signal corresponding to a tonal component and the energy of a vibration mode from an equivalent physical system to ensure energy conservation during transfers. Inter-modal energy transfer is encoded in a matrix containing all the coupling coefficients. The aim of this paper is not to propose a synthesis model performing an accurate simulation of a physical system. Instead, we seek to develop a framework allowing direct modelling of sounds targeted to the way they are perceived. This results in an efficient way to generate sounds evoking nonlinear sources and can yield real-time event-driven synthesis of sounds in virtual or augmented reality environments, a particularly active field of research [17] [18].

Some background on modal synthesis and linear filtering is given in Section 2. Then, the coupling between the filters is presented in Sec.3, the stability of the filters is discussed in Sec.3.2, and the definition of the matrix containing the coupling terms is proposed in Sec.4. Various example systems used to generate prototypical sounds are presented in Sec.5. Sound examples are available online at the following address [19].

2. MODAL SYNTHESIS AND LINEAR FILTERING

The modal resolution of a linear partial differential equations (PDE) system describing the vibrations of a resonant object is well-described in various texts [20]. Solutions are of the following form for the displacement w depending on a spatial coordinate $\mathbf{r}$ and time t:

$$w(\mathbf{r},t) = \underbrace{w_h(\mathbf{r},t)}_{\text{homogeneous solution}} + \underbrace{w_p(\mathbf{r},t)}_{\text{particular solution}}, \quad (1)$$

where

$$w_h(\mathbf{r},t) = \sum_{i=1}^{\infty} e^{-\alpha_i t}\left[A_i\cos(\omega_i t+\varphi_i)\right]\phi_i(\mathbf{r}) \quad (2a)$$

$$w_p(\mathbf{r},t) = \sum_{i=1}^{\infty} \left(g_i(t) * h_i(t)\right)\phi_i(\mathbf{r}), \quad (2b)$$

Here, $*$ represents a convolution operation, and the impulse response $h_i(t)$ of the following form:

$$h_i(t) = \frac{1}{\omega_i} e^{-\alpha_i t}\sin(\omega_i t) \quad (3)$$

the function $\phi_i(\mathbf{r})$ is the i^{th} mode's shape or basis function, ω_i and α_i are the angular frequency and the damping coefficient of the i^{th} mode, respectively. The constants A_i and φ_i derive from the initial conditions and $g_i(t)$ is the modal excitation (projection of an excitation source $g(\mathbf{r},t)$ onto the modal basis functions).

One may note that the modal model does not necessarily derive from the solution of partial differential equations. The modal parameters may be identified directly from experimental measurements (recording) or from numerical simulations.

A straightforward approach to numerical solution at a sample rate f_s in Hz is to use recursive filters with an exponentially-damped sinusoidal impulse response (IIR). The filter proposed by Mathews and Smith [16] has this property. The implementation of this filter consists in calculating, for each time step n, the imaginary part of a complex number $z(n)$ whose rotation speed in the complex plane is constant and corresponds to the angular frequency ω of the exponentially damped sinusoid:

$$y(n) = \mathrm{Im}(z(n)) \qquad \text{where} \qquad z(n+1) = Zz(n)+u(n) \quad (4)$$

with $u(n)$ the source of the filter, and Z the constant modification of the phase and modulus in one time step:

$$Z = e^{-\alpha/f_s}e^{j\omega/f_s} = X + jY \quad (5)$$

with $X = e^{-\alpha/f_s}\cos(\omega/f_s)$ and $Y = e^{-\alpha/f_s}\sin(\omega/f_s)$.

The recurrence equation on the complex sequence $z(n)$ is computed by the following system including a recurrence equation for the real part $x(n) = \mathrm{Re}(z(n))$ and a recurrence equation for the imaginary part $y(n) = \mathrm{Im}(z(n))$, which is the output of the filter:

$$\begin{aligned} x(n+1) &= \mathrm{Re}(z(n+1)) = Xx(n) - Yy(n) + u(n) \\ y(n+1) &= \mathrm{Im}(z(n+1)) = Yx(n) + Xy(n) \end{aligned} \quad (6)$$

for a real source $u(n) \in \mathbb{R}$.

3. COUPLING BETWEEN THE FILTERS

3.1. Principle and implementation

Consider N filters defined as in the previous section in parallel and we wish to couple them. We note $z_i(n)$ the complex sequence corresponding to the i^{th} filter, with $x_i(n)$ its real part and $y_i(n)$ its imaginary part (corresponding to the output signal of the filter). The source for the i^{th} filter, corresponding to the projection of the source of the filter bank $u(n)$ onto the i^{th} modal basis function, is noted $u_i(n)$.

The mechanical energy corresponding to the vibration of a mode is proportional to the square of the amplitude of the displacement. From a signal point of view, the square of the amplitude of a tonal component corresponds to twice the power of the signal. Postulating a linear relation between the displacement of the structure and the sound produced, we have chosen to model the energy transfers between the modes by power transfers between the filters [21]. If we only look at the power evolutions linked to the energy transfers (by postulating a null source), we write the following recurrence relation on the powers of the output signals of the different filters P_i:

$$P_i(n+1) = \left(P_i(n) + \underbrace{T_i(n)}_{\text{transfer}}\right)\underbrace{e^{-2\alpha_i/f_s}}_{\text{losses}} \quad (7)$$

with

$$P_i(n+1) \geq 0 \Leftrightarrow P_i(n) + T_i(n) \geq 0, \quad (8)$$

and $P_i(n)$ the power of the tonal component:

$$P_i(n) = \frac{|z_i(n)|^2}{2} = \frac{1}{2}(x_i(n)^2 + y_i(n)^2) \quad (9)$$

$x_i(n), y_i(n) \in \mathbb{R}$.

Thus, we can express the variation of the modulus of $z_i(n)$ due to energy transfer between two time steps:

$$|z_i(n+1)| = \sqrt{|z_i(n)|^2 + 2T_i(n)}\, e^{-\alpha_i/f_e} \quad (10)$$

We can define an amplitude ratio between the modulus for two consecutive time steps if $|z_i(n)| \neq 0$:

$$\frac{|z_i(n+1)|}{|z_i(n)|} = \underbrace{\sqrt{1 + \frac{2T_i(n)}{|z_i(n)|^2}}}_{\text{transfer}}\ \underbrace{e^{-\alpha_i/f_e}}_{\text{losses}} \quad (11)$$

Thus, we can modify the recurrence equation defined in the previous section (see Eq.(4)) by incorporating the modulus variations due to energy transfers. It gives the following recurrence relation for z_i, including the source and phase variations:

$$z_i(n+1) = \begin{cases} \sqrt{2T_i(n)}Z_i + u_i(n) & \text{if } z_i(n) = 0 \\ \sqrt{1 + \frac{2T_i(n)}{|z_i(n)|^2}}Z_i z_i(n) + u_i(n) & \text{else} \end{cases} \quad (12)$$

with $Z_i = e^{-\alpha_i/f_s}e^{j\omega_i/f_s} = X_i + jY_i$, as defined in Eq(5). One can note that $T_i(n) > 0$ if $z_i(n) = 0$ (see Eq.(8)). This implies that the term $\sqrt{2T_i(n)}$ is real in the first part of Eq.(12).

Finally, we can write the system of equations for the implementation of the filters (see Figure 1 for a representation in block diagram):

$$\begin{aligned} x_i(n+1) &= \mathrm{Re}(z_i(n+1)) = X_i\tilde{x}_i(n) - Y_i\tilde{y}_i(n) + u_i(n) \\ y_i(n+1) &= \mathrm{Im}(z_i(n+1)) = Y_i\tilde{x}_i(n) + X_i\tilde{y}_i(n) \end{aligned} \quad (13)$$

with

$$\tilde{x}_i(n) = \begin{cases} \sqrt{2T_i(n)} & \text{if } z_i(n) = 0 \\ \sqrt{1 + \frac{2T_i(n)}{|z_i(n)|^2}}\, x_i(n) & \text{else} \end{cases}$$

$$\tilde{y}_i(n) = \begin{cases} 0 & \text{if } z_i(n) = 0 \\ \sqrt{1 + \frac{2T_i(n)}{|z_i(n)|^2}}\, y_i(n) & \text{else} \end{cases} \quad (14)$$

In this way, power can be transferred among the different filters without affecting the phases. The coupling intervenes in the calculation of the transfer terms $T_i(n)$ which ultimately involve the other filters (see Figures 1 and 2).

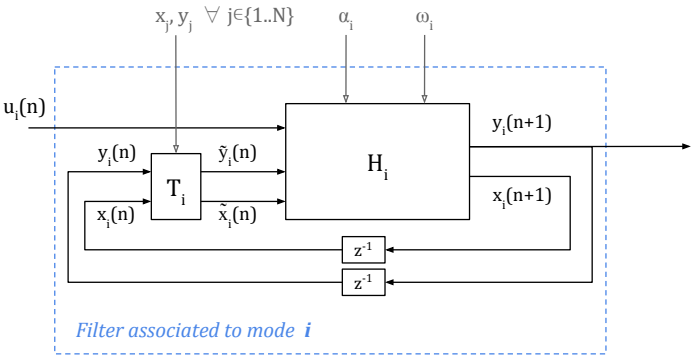


Figure 1: *Block diagram of a filter for the generation of the signal corresponding to a vibration mode. u corresponds to the source entering this particular filter and the block T_i corresponds to the calculation involving the states of other filters (coupling term).*

3.2. Energy and stability

A sufficient condition for the stability of the filter bank is to impose a non-positivity constraint for the transfer terms:

$$\sum_{i=1}^{N} T_i(n) \leq 0 \tag{15}$$

This condition impedes the creation of energy during transfer between modes for an equivalent physical system.

Also, the sum is bounded by the condition defined in Eq.(8) (a filter cannot transfer more power than it possesses):

$$\sum_{i=1}^{N} T_i(n) \geq -\sum_{i=1}^{N} P_i(n) \tag{16}$$

One can note that it is possible to consider a less restrictive stability condition that binds the transfer term to be lower than the power decrease due to losses:

$$\begin{aligned} &\sum_{i=1}^{N} \left(P_i(n) + T_i(n)\right)\, e^{-2\alpha_i/f_s} \leq \sum_{i=1}^{N} P_i(n) \\ &\quad \Leftrightarrow \quad \sum_{i=1}^{N} T_i(n)\, e^{-2\alpha_i/f_s} \leq \sum_{i=1}^{N} P_i(n) \left(1 - e^{-2\alpha_i/f_s}\right) \end{aligned} \tag{17}$$

However, this condition cancels the effect of dissipation and is not consistent with an equivalent physical system. We prefer to consider the condition presented Eq.(15) for the rest of the document.

4. DISTRIBUTION MATRIX

This section presents a formalism for the calculation and control of the coupling between filters. The challenge is to arrive at a model simple enough to be controllable (i.e., to be able to predict the sound outcome of a manipulation of the parameters) and complete enough to allow the matching of modal trajectories to a range of nonlinear phenomena.

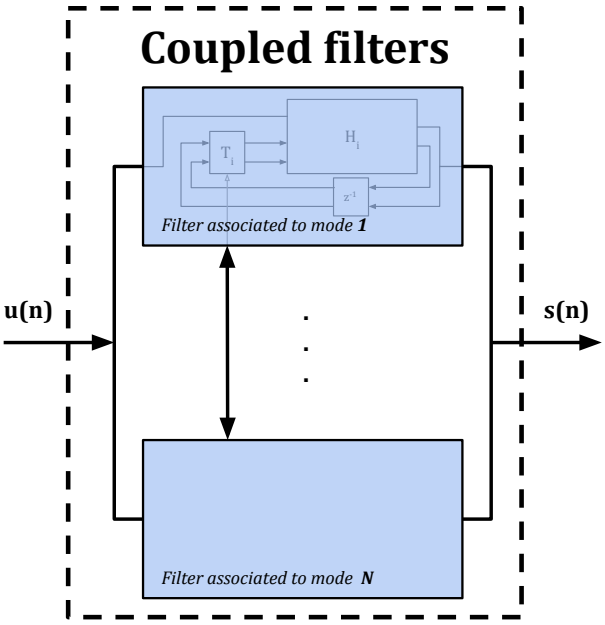


Figure 2: *Schematic representation of the coupled filter bank. the double arrow connecting the two boxes represents the coupling between the filters through the transfer vector* $\mathbf{t}$*. The output of the filter bank is the sum of the outputs of the filters* $s(n) = \sum_{i=1}^{N} y_i(n)$

Now define the column vectors $\mathbf{p}(n) = [P_1(n), \ldots, P_N(n)]^T$ and $\mathbf{t}(n) = [T_1(n), \ldots, T_N(n)]^T$. The power transfers between the tonal components $\mathbf{t}(n)$ at a given time step n are defined as:

$$\mathbf{t}(n) = \mathbf{M}\left[\mathbf{p}(n) - \boldsymbol{\tau}\right]_+ . \tag{18}$$

Here, $[\cdot]_+$ indicates the "positive part of", i.e., $[\zeta]_+ = \frac{1}{2}(\zeta + |\zeta|)$. An $N \times 1$ column vector $\boldsymbol{\tau}$ containing the thresholds τ_i, $i = 1, \ldots, N$ at which transfers are activated for each tonal component has also been introduced here.

Thus, the calculation of the transfer terms is performed by the matrix product of an $N \times N$ redistribution matrix $\mathbf{M}$ with the vector resulting from the positive part of the difference between the power of each frequency component $\mathbf{p}(n)$ and the associated threshold $\boldsymbol{\tau}$. In other words, the transfer terms $T_i(n)$ are proportional to the excess power above the corresponding threshold and the terms of the matrix $\mathbf{M}$ define the proportions distributed and received by each other component. Note that this relation is not an immediate consequence of a physical model but is a heuristic means of capturing salient phenoemna in a physical system. Our focus is on the design of a synthesis process with a predictable sound outcome rather than on the simulation of a physical system.

To respect the stability condition Eq.(15), we set the sum of all values of a given column of the matrix $\mathbf{M}$ to be lower or equal to zero. If M_{ij} is the i,jth entry of $\mathbf{M}$, then

$$\sum_{i=1}^{N} M_{ij} \leq 0 \;\; \forall\, j \;\; \Rightarrow \;\; \sum_{i=1}^{N} T_i(n) \leq 0 \tag{19}$$

The diagonal entries M_{jj} of the matrix $\mathbf{M}$ define the proportion of power of the jth mode that will be redistributed to other modes and the other terms of the column M_{ij} define the quantity that the ith mode will receive from this redistribution.

An efficient way to define the coefficients of the matrix is to use the following expression:

$$M_{ij} = \eta\lambda \frac{a_{ij}}{\sum_{i=1}^{N} a_{ij}} - \lambda\delta_{ij} \tag{20}$$

Where a_{ij} is a coefficient weighting the redistribution from the jth mode to the ith mode. In this formulation, the stability of the filter bank is ensured for arbitrary a_{ij}, provided that at least one value per column is non-zero and that $0 \leq \eta \leq 1$. η corresponds to the efficiency of the transfers ($\eta = 1 \Rightarrow \sum T_i(n) = 0$). λ is the proportion of power above threshold transferred to other modes at each time step ($0 \leq \lambda \leq 1$). The values of the off-diagonal elements M_{ij} of the matrix $\mathbf{M}$ are the proportion of energy transferred by the mode j that will be received by the mode i.

The i^{th} transfer term $T_i(n)$ can be expressed as follows:

$$T_i(n) = \underbrace{\eta\lambda \sum_{j=1}^{N} \left[\frac{a_{ij}}{\sum_{i=1}^{N} a_{ij}} \left(P_j(n) - \tau_j\right) \right]}_{\text{positive contribution } T_{i+}(n)} - \underbrace{\lambda \left(P_i(n) - \tau_i\right)}_{\text{negative contribution } T_{i-}(n)} \tag{21}$$

5. EXAMPLES

Nonlinear vibration leads to complex phenomena that can produce subtle and chaotic variations in radiated sound. We can reduce the complexity of the model and propose a heuristic that attempts to maintain the essential perceptual attributes of an object vibrating under nonlinear conditions. The resulting synthetic sound is nevertheless less realistic and versatile than sounds generated by the direct resolution of physical models (such as, e.g., the Föppl-von Kármán system) although the synthesis quality can be improved by using random processes in the implementation of the algorithms.

The coupled filter bank proposed here is dependent on many parameters: the number of filters N, the oscillation frequencies ω_i and damping α_i for each filter, the coefficients a_{ij} and the parameters λ and η for the definition of the redistribution matrix $\mathbf{M}$, and the thresholds τ_i. Strategies for setting these parameters are presented in two cases of musical interest. In the case of nonlinear plate vibration, energy is transferred to filters of near frequency in order to generate a gradual cascade of energy towards the high-frequency range. In the case of a string colliding with a rigid object, in contrast, there is simultaneous transfer or energy to many frequency components.

5.1. Energy cascade in thin plates

Consider a thin rectangular plate (according to the Kirchhoff model [22]), with mass density ρ kg$\cdot$ m^{-3}, thickness H m, and flexural rigidity D in kg$\cdot$m$^2\cdot$s^{-2}, and side lengths L_x and L_y in m. If the plate is simply supported on all its edges, the modal frequencies ω_{lm} and modal shapes $\phi_{lm}(x, y)$ can be expressed as follows [23]:

$$\omega_{lm} = \frac{\pi^2}{L_x^2}\sqrt{\frac{D}{\rho H}}\left(l^2 + \nu^2 m^2\right) \quad \phi_{lm}(x, y) = \sin(l\pi x)\sin(m\pi y) \tag{22}$$

Here, $\nu = L_x/L_y$ is the plate aspect ratio, or the ratio between the length and width of the plate. The integer indices $l, m \geq 1$ correspond to the number of vibration nodes in the main directions of the rectangular plate (Cartesian coordinates (x, y)) with x and y being normalized by the length of the plate in the corresponding direction (so that $0 \leq x, y \leq 1$).

For a point excitation force located at (x_e, y_e), we can compute the modal forces using the mode shapes evaluated at the excitation point as $\phi_{lm}(x_e, y_e)$. We define the source of the l, mth filter as follows:

$$u_{lm}(n) = \sin(l\pi x_e)\sin(m\pi y_e)u(n) \tag{23}$$

where $u(n)$ is the global excitation function.

We use a raised sinusoid for the excitation force (as proposed in [4] and [24]) to simulate an impact:

$$u(n) = \begin{cases} A\sin^2(\pi n/N_{ex}) & \text{if } n \leq N_{ex} \\ 0 & \text{else} \end{cases} \tag{24}$$

For typical plate strikes, the strike duration N_{ex}/f_s in seconds is on the order of 1-4 ms.

The damping coefficients are chosen according to an exponential law, as proposed by Aramaki et al.[25], with parameters that are set to evoke a metallic object:

$$\alpha_{lm} = e^{(\alpha_G + \omega_{lm}\alpha_R)} \tag{25}$$

with $\alpha_R = 4 \times 10^{-5}$ and $\alpha_G = 0.33220$. This set of parameters permits direct modal synthesis for linear plate vibration. To each pair of indices (l, m) we associate an index i (perhaps chosen in terms of increasing modal frequency) corresponding to the filter number used to generate the corresponding tonal component.

In order to produce the cascade of energy towards the high frequency components, we carry out transfers between filters whose frequencies are close. Indeed, the energy supplied by the impact is localised at low frequencies and transfers directed towards the neighbouring modes allow the progressive appearance of higher frequency components. The weighting coefficients a_{ij} can be set as follows (see Figure 3):

$$a_{ij} = \left[1 - \frac{|f_j - f_i|}{\Delta f}\right]_+ \tag{26}$$

with $f_i = \frac{\omega_i}{2\pi}$

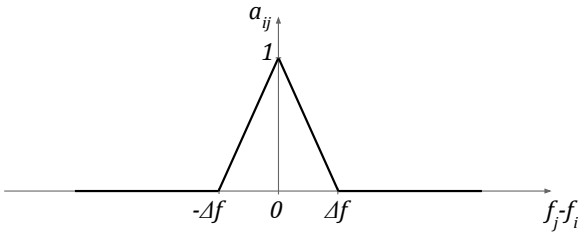


Figure 3: *Value of the coefficient a_{ij} as a function of the frequency difference between filter i and j.*

We set $\eta = 1$ (ensuring conservation of energy during the redistribution). The cascade can be mainly controlled by λ, or by the definition of thresholds τ_i (see Figures 4 and 5).

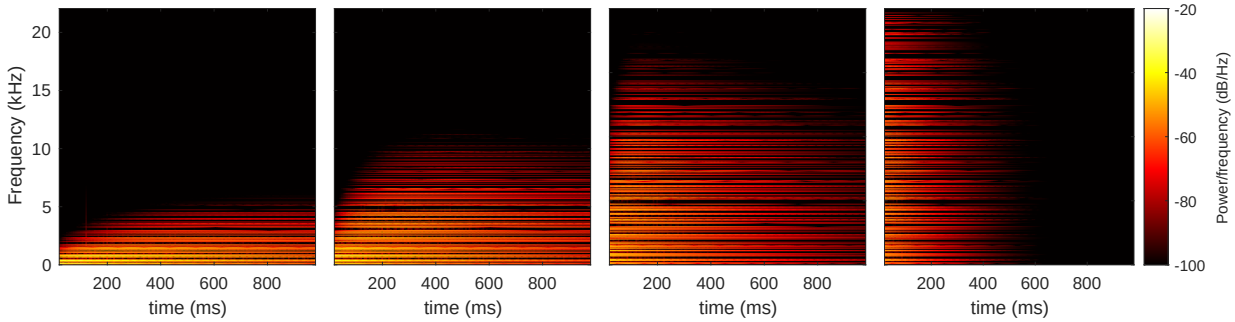


Figure 4: *Spectrograms of output for filters whose frequency corresponds to the modal frequency of a thin plate for different values of λ ($\tau_i = 0$). From left to right: $\lambda = 0.001$, $\lambda = 0.01$, $\lambda = 0.1$, $\lambda = 1$. We can observe that the energy cascade spreads faster and higher in frequency with the increase of λ. Transfers are performed at each time step.*

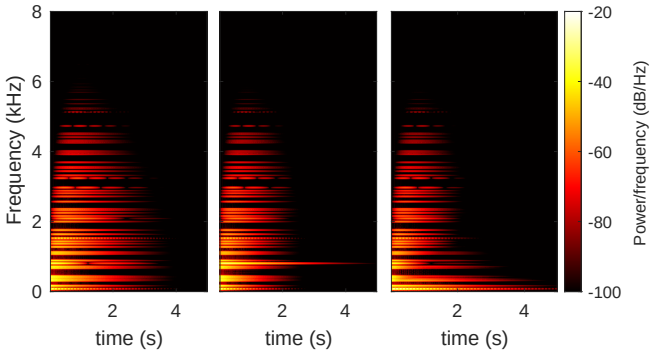


Figure 5: *Spectrograms of output for filters whose frequency corresponds to the modal frequency of a thin plate for different thresholds τ_i. Left: $\tau_i = 0$; middle: $\tau_i = 0$ except for $i = 10$ where $\tau_1 0 = 1$; right: τ_i is half the excitation amplitude. All tonal components decay simultaneously when the thresholds are zero (left). A component emerges and decays more slowly when its threshold is non-zero (middle). When all thresholds are different from zero, we observe a usual exponential decay after the delayed appearance of the high frequency component (right).*

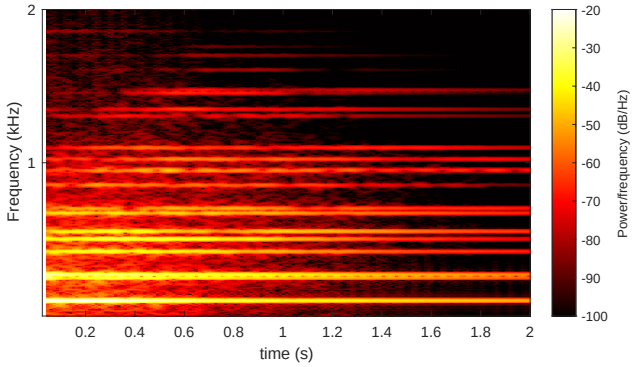


Figure 6: *Spectrogram of output for filters whose frequency corresponds to the modal frequency of a thin plate. The random modulation of the redistributions induces rapid variations in the amplitude of the tonal components which generate noise and beating in the signal.*

In the case of wave turbulence in plates [26], couplings between modes can lead to rapid variations in amplitude and frequency leading to a chaotic regime. In the chaotic regime, the resulting signal is noisy, and difficult to reproduce by a set of tonal components. One way to reproduce this phenomenon with the coupled filters presented in this paper is to pass randomized phases to the positive contributions of the transfer term in the source. In this way, the tonal components are subject to rapid random amplitude modulations that can evoke the chaotic phenomenon occurring during wave turbulence in the plates (see Figure 6).

5.2. Collisions in sound production

The perturbation of the vibrations of an object when colliding with an obstacle can lead to different types of sound events. In the typical case of a guitar, the player can choke the string, mute it, play a natural harmonic. The string can also interact with the soundboard (slap, string buzz).

The model of a vibrating string with simply supported boundary conditions gives the following modal frequencies and shapes:

$$\omega_i = i\omega_1 \qquad \phi_i(x) = \sin(i\pi x) \tag{27}$$

where here, the spatial coordinate x is normalized by the length of the string ($0 \leq x \leq 1$). For a point excitation force located at x_e, the source of the i^{th} filter can be defined as:

$$u_i(n) = \sin(i\pi x_e)u(n) \tag{28}$$

We use the same excitation force and damping model than previously (see Eqs.(24) and (25)).

The evocation of an obstacle disturbing the vibrations of the string requires the definition of thresholds that correspond to the location of the obstacle. We propose thresholds corresponding to the maximum amplitude of modal displacements without colliding with a virtual obstacle positioned at x_c, y_c, where y_c is the vertical displacement of the obstacle relative to the string:

$$\tau_i = \frac{1}{2}\left(\frac{y_c}{\sin(i\pi x_c)}\right)^2 \tag{29}$$

We define a redistribution matrix with all columns being identical in order to cause a simultaneous redistribution to a set of tonal components. The coefficients a_{ij} are defined as follows:

$$a_{ij} = |\sin(i\pi x_c)|\xi_i(f_i) \tag{30}$$

with $\xi_i(f_i)$ a parameter depending on the frequency allowing weighting of the redistribution according to the filter frequency. We define $\xi_i(f_i)$ as the Fourier transform of the raised cosine, an approximation of the force profile caused by a collision (as defined for the source, Eq.(24)):

$$\xi_i = \text{sinc}(f_i\gamma) + \frac{1}{2}\left(\text{sinc}(f_i\gamma - 1) + \text{sinc}(f_i\gamma + 1)\right) \tag{31}$$

with f_i the frequency of the i^{th} filter and γ a parameter corresponding to the duration of the raised cosine. This results in a cut-

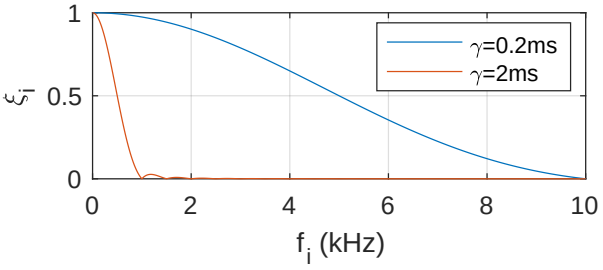


Figure 7: *Value of ξ_i as a function of the frequency of filter i.*

off frequency beyond which there is no more transfer (see Figure 7). Various examples of sound outputs for different configurations are presented—see Figures 8 and 9.

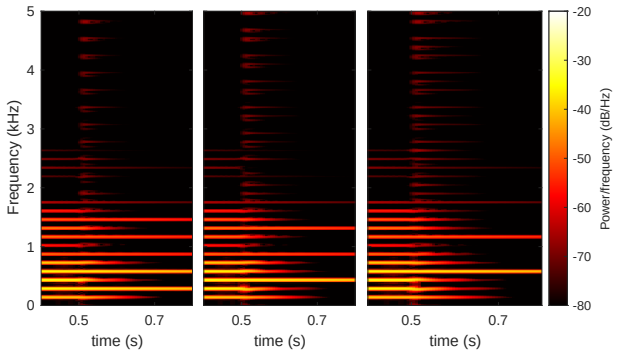


Figure 8: *Spectrograms of output for filters whose frequency are harmonic for different values of x_c ($y_c = 0$, $\gamma = 2 \times 10^{-4}s$, $\lambda = 0.25$, $\nu = 0.5$). Transfers are performed every 294 samples for times greater than 500ms, which corresponds to a collision every 6.67ms (150Hz). From left to right: $x_c = 1/2$, $x_c = 1/3$, $x_c = L/4$. We can observe that the transfer does not affect even harmonics (resp. multiples of 3 and 4) for $x_c = 1/2$ (resp.$x_c = 1/3$ and $x_c = L/4$), which allows the reproduction of a natural harmonic played on a guitar.*

Collisions in musical instruments may be the source of more subtle phenomena than a simultaneous appearance of various frequency components. The cases of string buzz and tanpura can be

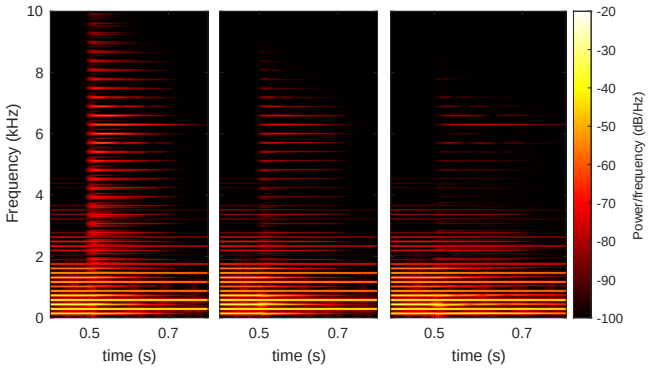


Figure 9: *Spectrograms of output for filters whose frequency are harmonic for different values of ν and γ ($x_c = 1/2$, $y_c = 0$). From left to right: ($\nu = 0.5$, $\gamma = 2 \times 10^{-4}s$), ($\nu = 0.15$, $\gamma = 2 \times 10^{-4}s$), ($\nu = 0.5$, $\gamma = 2 \times 10^{-3}s$). Transfers are performed every 294 samples for times greater than 500ms, which corresponds to a collision every 6.67ms (150Hz). There is a lower increase in the high-frequency components and a faster dissipation of all the tonal components involved in the redistribution as the efficiency decreases. As γ increases, there is also less energy distributed to the high-frequency components, but this energy is not dissipated and remains in the low-frequency components.*

approached by introducing random processes into the redistribution, as has been done for chaotic phenomena in plates (see Figure 10).

It is possible to apply the same principle for the generation of sounds corresponding to collisions with 2D objects. For example, we can generate muted plate sounds (see Figure 11).

6. CONCLUSION AND FURTHER WORK

In this paper, we have presented the design of coupled resonant filters geared towards the emulation of mode coupling effects in nonlinear vibrating structures. This filter bank allows efficient and real-time sound synthesis even for a large number of filters. The coupling, performed without modifying the phase, introduces predictable and controllable effects on the output signal. The terms controlling the coupling between the different filters are grouped in a matrix whose definition is the main challenge. The setting of the parameters of the sound synthesis process is presented through various examples corresponding to sources whose behavior is nonlinear. A simple setting allows the generation of typical sounds, though sometimes with an unnatural character. The introduction of random processes in the energy redistribution can add a lot in terms of plausibility.

Future work will be concerned with determining which sound morphologies are important from a perceptual point of view for the recognition of sound events [27] corresponding to nonlinear phenomena in order to reproduce them with this coupled filter bank. This could lead to the development of environmental sound synthesizers and virtual musical instruments (e.g. tanpura, cymbal ...), or to non-linear audio effects (such as the nonlinear reverberation of a snare drum due to the wires held under tension against the lower drumskin). The filter bank presented in this paper can also be used as an abstract sound creation tool. In this context, the chal-

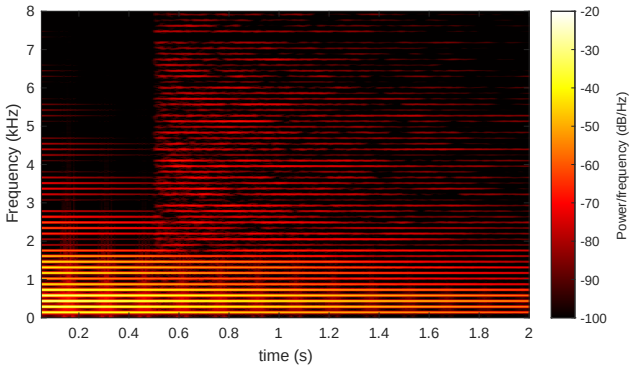


Figure 10: *Spectrograms of output for filters whose frequency are harmonic with the introduction of random processes during the redistribution ($\lambda = 0.001$, $\nu = 0.9$, $x_c = 0.38$, $y_c = 0.001$, $\gamma = 2 \times 10^{-4}s$). Transfers are performed every 294 samples for times greater than 500ms, which corresponds to a collision every 6.67ms (150Hz).*

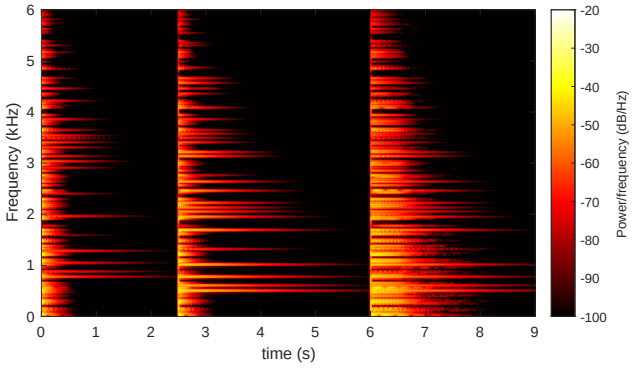


Figure 11: *Spectrograms of output for filters whose frequency corresponds to the modal frequency of a thin plate. here $\nu = 0$ and we observe the quick dissipation of certain tonal components for three distinct impacts, which creates a sensation of choking.*

lenge would be to design intuitive control for use in a musical or sound design context.

7. REFERENCES

[1] Jean-Marie Adrien, "The missing link: Modal synthesis," in *Representations of musical signals*, pp. 269–298. 1991.

[2] Joseph Derek Morrison and Jean-Marie Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[3] Davide Rocchesso, "The ball within the box: a sound-processing metaphor," *Computer Music Journal*, vol. 19, no. 4, pp. 47–57, 1995.

[4] Kees Van Den Doel, Paul G Kry, and Dinesh K Pai, "Foleyautomatic: physically-based sound effects for interactive simulation and animation," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 537–544.

[5] Simon Conan, Etienne Thoret, Mitsuko Aramaki, Olivier Derrien, Charles Gondre, Sølvi Ystad, and Richard Kronland-Martinet, "An intuitive synthesizer of continuous-interaction sounds: Rubbing, scratching, and rolling," *Computer Music Journal*, vol. 38, no. 4, pp. 24–37, 2014.

[6] KA Legge and Neville H Fletcher, "Nonlinearity, chaos, and the sound of shallow gongs," *The Journal of the Acoustical Society of America*, vol. 86, no. 6, pp. 2439–2443, 1989.

[7] Antoine Chaigne, Cyril Touzé, and Olivier Thomas, "Nonlinear vibrations and chaos in gongs and cymbals," *Acoustical science and technology*, vol. 26, no. 5, pp. 403–409, 2005.

[8] A. Föppl, *Vorlesungen über technische Mechanik*, Druck und Verlag von B.G. Teubner, Leipzig, 1907.

[9] T. von Kármán, "Festigkeitsprobleme im maschinenbau," *Encyklopädie der Mathematischen Wissenschaften*, vol. 4, no. 4, pp. 311–385, 1910.

[10] Stefan Bilbao, "A family of conservative finite difference schemes for the dynamical von karman plate equations," *Num. Meth. PDE*, vol. 24, no. 1, pp. 193–216, 2008.

[11] Michele Ducceschi and Cyril Touzé, "Modal approach for nonlinear vibrations of damped impacted plates: Application to sound synthesis of gongs and cymbals," *Journal of Sound and Vibration*, vol. 344, pp. 313–331, 2015.

[12] Michele Ducceschi and Cyril Touzé, "Simulations of nonlinear plate dynamics: an accurate and efficient modal algorithm," in *18th International Conference on Digital Audio Effects (DAFx-15)*, 2015.

[13] Stefan Bilbao, Alberto Torin, and Vasileios Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acustica united with Acustica*, vol. 101, no. 1, pp. 155–173, 2015.

[14] Clara Issanchou, Stefan Bilbao, Jean-Loic Le Carrou, Cyril Touzé, and Olivier Doaré, "A modal-based approach to the nonlinear vibration of strings against a unilateral obstacle: Simulations and experiments in the pointwise case," *Journal of Sound and Vibration*, vol. 393, pp. 229–251, 2017.

[15] Travis Skare and Jonathan Abel, "Real-time modal synthesis of crash cymbals with nonlinear approximations, using a gpu," in *Proc. 22nd Int. Conf. Dig. Audio Effects*, 2019.

[16] Max Mathews and Julius O Smith, "Methods for synthesizing very high q parametrically well behaved two pole filters," in *Proceedings of the Stockholm Musical Acoustics Conference (SMAC 2003)(Stockholm), Royal Swedish Academy of Music (August 2003)*, 2003.

[17] Laurent Pruvost, Bertrand Scherrer, Mitsuko Aramaki, Sølvi Ystad, and Richard Kronland-Martinet, "Perception-based interactive sound synthesis of morphing solids' interactions," in *SIGGRAPH Asia 2015 Technical Briefs*, pp. 1–4. 2015.

[18] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, et al., "Threedworld: A platform for interactive multi-modal physical simulation," *arXiv preprint arXiv:2007.04954*, 2020.

[19] S. Poirot, "Sound examples," Available at https://www.prism.cnrs.fr/publications-media/DaFXPoirot/, accessed April 07, 2023.

[20] Zhi-Fang Fu and Jimin He, *Modal analysis*, Elsevier, 2001.

[21] S. Poirot, *Des non-linéarités physiques à la perception sonore*, Ph.D. thesis, University of Aix-Marseille, 2021.

[22] K. F. Graff, *Wave Motion in Elastic Solids*, Dover Publications, New York, USA, 1991.

[23] Neville H Fletcher and Thomas D Rossing, *The physics of musical instruments*, Springer Science & Business Media, 2012.

[24] Stefan Bilbao, *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*, John Wiley & Sons, 2009.

[25] Mitsuko Aramaki, Mireille Besson, Richard Kronland-Martinet, and Sølvi Ystad, "Controlling the perceived material in an impact sound synthesizer," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 2, pp. 301–314, 2010.

[26] Michele Ducceschi, Cyril Touzé, Olivier Thomas, and Stefan Bilbao, "Dynamics of the wave turbulence spectrum in vibrating plates: A numerical investigation using a conservative finite difference scheme," *Physica D*, vol. 280, pp. 73–85, 2014.

[27] Richard Kronland-Martinet, Sølvi Ystad, and Mitsuko Aramaki, "High-level control of sound synthesis for sonification processes," *AI & society*, vol. 27, no. 2, pp. 245–255, 2012.

[28] Samuel Poirot, Stefan Bilbao, Mitsuko Aramaki, and Richard Kronland-Martinet, "Sound morphologies due to non-linear interactions: Towards a perceptual control of environmental sound synthesis processes," in *DAFx2018*, 2018.

MODULATION EXTRACTION FOR LFO-DRIVEN AUDIO EFFECTS

Christopher Mitcheltree *Christian J. Steinmetz* *Marco Comunità* *Joshua D. Reiss*

Centre for Digital Music, Queen Mary University of London, UK
{c.mitcheltree, c.j.steinmetz, m.comunita, joshua.reiss}@qmul.ac.uk

ABSTRACT

Low frequency oscillator (LFO) driven audio effects such as phaser, flanger, and chorus, modify an input signal using time-varying filters and delays, resulting in characteristic sweeping or widening effects. It has been shown that these effects can be modeled using neural networks when conditioned with the ground truth LFO signal. However, in most cases, the LFO signal is not accessible and measurement from the audio signal is nontrivial, hindering the modeling process. To address this, we propose a framework capable of extracting arbitrary LFO signals from processed audio across multiple digital audio effects, parameter settings, and instrument configurations. Since our system imposes no restrictions on the LFO signal shape, we demonstrate its ability to extract quasiperiodic, combined, and distorted modulation signals that are relevant to effect modeling. Furthermore, we show how coupling the extraction model with a simple processing network enables training of end-to-end black-box models of unseen analog or digital LFO-driven audio effects using only dry and wet audio pairs, overcoming the need to access the audio effect or internal LFO signal. We make our code available and provide the trained audio effect models in a real-time VST plugin[1].

1. INTRODUCTION

In music composition, production, and engineering, audio effects play a key role in altering the sound toward the desired result. Modulation effects such as phaser, flanger, and chorus, are part of a broad family of audio processors based on using a modulation signal to modify the spectrum, loudness, or spatial characteristic of the input audio. The typical modulation signals adopted are periodic (e.g., sinusoidal, sawtooth, triangular) with a frequency below the audible range (20 Hz) and are therefore called low frequency oscillators (LFO). Since oscillators are used to continuously vary the internal parameters of these effects, the exact shape, frequency, and phase of the LFO signal plays a crucial role, affecting the overall timbre and temporal behavior. This is especially evident in analog circuits where imperfections and nonlinearities may cause distortion and quasi-periodicity of the oscillation.

Digital emulation of audio effects is an area of active research [1–3], and many methods have been developed to analyze and emulate effect units. Depending on the degree of prior knowledge and reliance on measurement data, these can be divided into white- [4–7], gray- [8–13], or black-box [14–17] approaches. Most prior work on modulation effects modeling uses complex and time-consuming white-box approaches, obtaining models that are not easily transferable to other designs or LFO-driven effects. There are also examples of gray-box approaches, which strike a balance between general validity of the block-based model [8] and emulation quality of a specific unit [9, 10]. However, the modeling capabilities and robustness of such models is limited by the hand-engineered measurement techniques used to extract the LFO signal [9–12] and assumptions made about the LFO's shape.

Work in [14, 15] proposes recurrent and convolutional neural networks for black-box modeling of time-varying audio effects. Relying only on datasets of dry-wet audio, these are the first end-to-end approaches applied to LFO-driven effects. The method achieves good results with non-causal and non-controllable implementations, but does not explicitly learn the LFO signal and is not evaluated on unseen effects, audio, or LFO shapes.

To address the challenges of modeling a wide range of modulation effects and configurations, we introduce a neural architecture that is trained to extract arbitrary LFO signals from phaser, flanger, and chorus audio effects across varying parameter settings. By training this model on a dataset of guitar signals with basic phaser, flanger, and chorus implementations, we demonstrate our model achieves:

- Accurate modulation extraction from unseen audio sources.
- Extrapolation to complex modulation signals such as quasiperiodic, combined, and distorted LFO signals.
- Generalization across effect implementations for unseen analog and digital phaser and flanger effects.
- End-to-end causal modeling of analog and digital LFO-driven effects without access to the internal LFO signal.

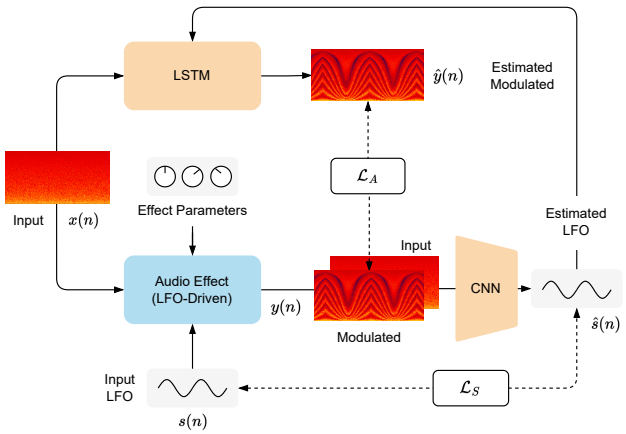


Figure 1: *By using the pretrained LFO extraction model (CNN) to analyze input and modulated audio, our proposed system enables training of a black-box neural network model (LSTM) on modulation audio effects without access to the ground truth LFO signal.*

[1] https://christhetree.github.io/mod_extraction/

2. BACKGROUND

2.1. Low Frequency Oscillators

In 1964 Robert Moog introduced the first transistor-based voltage controlled oscillator (VCO) and voltage controlled amplifier (VCA) designs [18]. These circuits are at the origin of modular synthesizers and later on led to modulation audio effects like phaser, flanger, and chorus. While VCOs were used to generate pitched sounds, VCAs were responsible for the envelope of synthesized notes. In his designs Moog also included VCOs oscillating at frequencies below 20 Hz, i.e. LFOs, to modulate other signal parameters (e.g., frequency, phase, amplitude) or processing blocks (e.g., panning, cutoff frequency). The most common types of modulations stem from periodic waveforms like sine, sawtooth, triangle, or square, but often extend to more complex shapes.

In analog effects [4, 10], non-ideal components can cause distortions from the reference shape as well as deviations that cause quasiperiodicity. There are also cases, like chorus effects, where random LFO signals are adopted. Furthermore, with the prevalence of digital emulations and software synthesizers, modulation signals can achieve an even wider diversity than their analog counterparts. As a result, the extraction of modulation signals from processed audio has applications beyond virtual analog modeling.

2.2. Modulation Effects

Phaser and flanger are examples of modulations affecting the spectrum of a signal, while chorus affects the pitch and timing.

Phaser — Phasing is achieved by using a series of notch or allpass filters [19]. The typical analog implementation uses an even number of first order allpass filters, which have flat magnitude response but phase that varies between 0° and -180°. When two filters are connected in series the phase varies back to 0° and, by mixing the filtered output with the input signal phase cancellations occur at frequencies around the 180° point. Altering the center frequency of the filters creates a characteristic sweeping sound.

Flanger — In a flanger, a delayed copy of the input signal is summed to the dry input itself causing constructive and destructive interference. The delay is periodically modulated but usually kept below ≈ 15 ms. As a result, it is often perceived as a time-varying comb filter. In contrast to phaser effects, where the frequency distance between notches is kept constant on a logarithmic scale, in flanger effects, the distance changes with the delay value.

Chorus — Chorus effects are identical to flangers in implementation, but use multiple delayed and modulated copies of the input signal. Also, by adopting larger delays - around ≈ 30 ms - the output is perceived as a sum of slightly pitch shifted copies of the input, as when multiple instruments or voices are playing in unison. Therefore, there is not a clearly observable modulation of the spectrum compared to phasers and flangers.

2.3. Virtual Analog Modeling of Modulation Effects

Research in virtual analog modeling aims to develop methods that emulate the characteristics and behaviors of a reference unit. These methods can be divided into white-, gray-, or black-box modeling depending on the degree of prior knowledge and type of measurements they rely on. To create accurate simulations, white-box modeling [4–7] requires a thorough understanding of the system, and typically employs differential equations to describe its behavior and numerical methods to solve them. Therefore, such methods are often associated with a time consuming design process and computationally demanding and non-transferable implementations. Circuit analysis together with voltage and current measurements are used to create a state-space model of a phaser effect pedal in [6], while in [7] a similar analysis is used to emulate a bucket brigade delay circuit that is then employed in flanger effect emulation. Phaser, flanger, and chorus are also modeled in [4], where the authors discretize the differential equations of JFET transistors and transconductance amplifiers used in such effects.

To reduce prior knowledge necessary to model a device, gray-box approaches combine a partially theoretical structure with input-output measurements [9, 10, 12]. However, they still require ad hoc measurement and optimization procedures [9, 12] and knowledge of the underlying implementation. A gray-box model of phaser effect pedal is presented in [10], where nonlinear allpass filter blocks are combined with analysis and measurement of the interaction between light dependent resistors and incandescent lamp optocoupler controlling the LFO. This work shows how critical the LFO signal can be in shaping the overall sound of a design. In [9] we have an example of a measurement signal and extraction algorithm specifically designed to capture a phaser's LFO signal.

A similar measurement is adopted in [11], and the extracted LFO signal is used to condition neural networks trained on phaser and flanger effects. A custom extraction algorithm is implemented: the LFO shape (rectified sine) is observed in the output and given to a least-squares solver. Furthermore, custom training data is required, where the test signal is interposed between samples so that the initial LFO phase can be extracted. This work is developed further in [12] by improving the measurement technique.

In black-box approaches, minimal knowledge of the system is required and modeling mostly relies on input-output measurements. A major advantage is that they simplify the process to collecting adequate data. However, these models often lack interpretability and might entail time-consuming optimizations. In [14, 15], we have the only examples of black-box models of time-varying audio effects. Neural networks are successfully trained on many modulation effect types. However, these models are non-causal, non-controllable, and have not been tested on unseen LFO shapes or audio signals different from the training data.

2.4. Effects Recognition and Parameter Estimation

Beyond effect modeling, there has also been research on recognition of audio effects and effect chains, as well as control values from processed audio. Our task of LFO extraction can be viewed as a specific form of audio effect parameter estimation, however, existing works have yet to consider reconstructing the LFO signal itself. Early works focus on audio effects classification [20], while others extend this task to target the identification of specific effect units and their control values [21], including within mixtures [22]. Recently, work has generalized this task to the complete reconstruction of a graph of audio effects and their parameter values [23]. In [24], the authors focus on dynamic range compression, and train neural networks to extract ratio, attack, and release times, and total harmonic distortion from a reference signal. Extracting information from audio recordings for applications in music production and sound synthesis is still at an early stage, and the work presented here also aims to contribute in these directions.

3. METHODOLOGY

We approach the problem of modeling an LFO-driven audio effect in two steps. First, we develop an LFO extraction model which can be trained to reconstruct the modulation signal from dry and wet audio pairs. Then we feed the extracted LFO signal along with the dry audio to an effect model that can be trained to reconstruct the wet audio. Figure 1 visualizes our approach with a block diagram.

3.1. LFO Extraction

The LFO extraction model (LFO-net), shown in Figure 2, is a convolutional neural network (CNN) consisting of sequential convolutional blocks. As input it takes a 2-channel Mel spectrogram of the dry and wet audio. Each block consists of LayerNorm [25] across the frequency and time dimensions, a 2D convolution, Max Pooling, and a PReLU activation [26]. Feature maps are max-pooled only along the frequency dimension and dilated only along the time dimension, similar to how a temporal convolutional network (TCN) operates [27]. As a result, the temporal receptive field of the network grows exponentially with each convolutional block while the frequency resolution decreases exponentially. The final layer of the network is a time-distributed linear layer that estimates the LFO value for the current frame between 0 and 1.

Training — LFO-net is trained using the AdamW optimizer to minimize the L_1 error between the reconstructed modulation signal $\hat{s}$ and the ground truth modulation signal s, each with N timesteps

$$L_1(s, \hat{s}) = \frac{1}{N} \sum_{n=1}^{N} |s(n) - \hat{s}(n)| \tag{1}$$

where n is the time index. We also include terms for the first-order central difference error

$$s'(n) = \frac{s(n+1) - s(n-1)}{2}. \tag{2}$$

as well as the L_1 error of the second-order central difference, which is defined recursively

$$s''(n) = \frac{s'(n+1) - s'(n-1)}{2}. \tag{3}$$

These terms are scaled by α, β, and γ respectively and encourage the network to learn smoother modulation signals. The complete loss function $\mathcal{L}_S$ can be expressed as follows:

$$\mathcal{L}_S = \alpha L_1(s, \hat{s}) + \beta L_1\left(s', \hat{s}'\right) + \gamma L_1\left(s'', \hat{s}''\right) \tag{4}$$

Based on initial testing, we selected $\alpha = 1$, $\beta = 5$, and $\gamma = 10$ to weigh the different terms. In addition, SpecAugment [28] was used for masking both frequency and time dimensions during training to increase robustness.

Post-processing — Since LFO-net imposes no restrictions on the shape of the LFO besides being bounded between 0 and 1, the output can appear noisy or irregular. To improve the quality of the extracted LFO signal we introduce three post-processing steps, shown in Figure 4. First, the signal is smoothed with a 4th order moving average filter. This is followed by "stretching" of the peaks and troughs so they are equal to 0 and 1, respectively. This is achieved by finding the locations of local minima and maxima, and then linearly interpolating consecutive sections to span from 0 to 1. Finally, when training effect models, invalid reconstructed LFO signals where there are too many peaks or troughs or where consecutive peaks or troughs are too close together are thrown out.

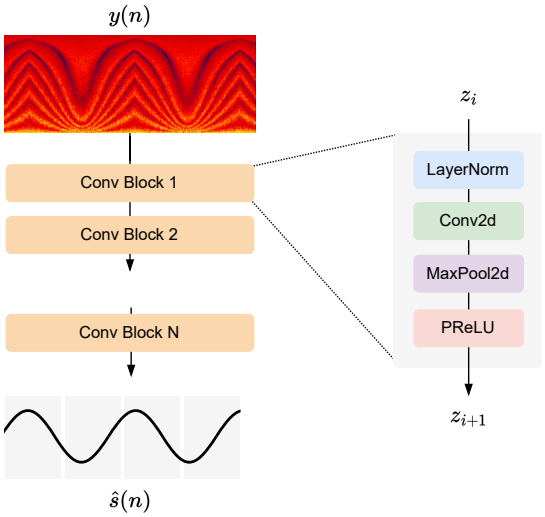


Figure 2: *LFO extraction model (LFO-net) diagram.*

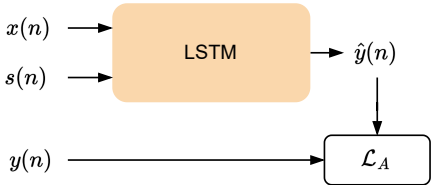


Figure 3: *LFO effect modeling block diagram.*

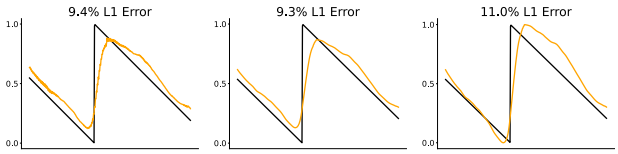


Figure 4: *Examples of original (left), smoothed (center), and stretched (right) post-processed modulation signals.*

This helps stabilize training by only using examples where the estimated LFO is likely to be an accurate prediction of the ground truth LFO signal. The last two post-processing steps are only used for the unseen effect experiments in Section 4.5.

3.2. Effect Modeling

Our effect model, shown in Figure 3, is based on previous work in black-box modeling of modulation effects [12]. It consists of a long short-term memory (LSTM) network with a time-distributed linear layer that compresses the latent space into a single sample-by-sample value which is added to the input audio and then bounded by a hyperbolic tangent activation. The network takes as input two channels: the dry audio and an LFO conditioning signal.

Training — Training is done on blocks of 1024 samples using truncated backpropagation through time (TBPTT) with 1024 samples of warmup. Once again, the AdamW optimizer is used to minimize $\mathcal{L}_A$: the L_1 loss between the output audio and the ground truth wet audio. We do not train using Error-to-Signal Ratio (ESR) and DC loss as in [29] since in our experiments we found using just the L_1 loss resulted in better results across all metrics.

Table 1: *Parameter values for the "fixed params" and "varying params" evaluation configurations.*

		LFO Parameters			Effect Parameters					
Effect	Config.	Shape	Phase	Rate	Center Freq.	Min. Delay	Delay Width	Feedback	Depth	Mix
Phaser	Fixed	Cos.	0 - 2π	0.5 - 3.0 Hz	440 Hz	-	-	0.25	1.0	1.0
	Varying				70 - 18k Hz	-	-	0.0 - 0.7	0.25 - 1.0	1.0
Flanger	Fixed	All	0 - 2π	0.5 - 3.0 Hz	-	1 ms	4 ms	0.25	1.0	1.0
	Varying				-	0 - 1 ms	2.5 - 10 ms	0.0 - 0.7	0.25 - 1.0	1.0
Chorus	Fixed	All	0 - 2π	0.5 - 3.0 Hz	-	20 ms	10 ms	0.25	1.0	1.0
	Varying				-	11 - 30 ms	2.5 - 10 ms	0.0 - 0.7	0.25 - 1.0	1.0

4. EXPERIMENTS

4.1. Modulation Extraction

Most phaser, flanger, and chorus implementations do not allow defining an arbitrary LFO signal. As a result, in order to be able to train the LFO extraction model with effects using arbitrary LFO signals, we implement our own flanger/chorus effect directly in PyTorch so that it can run on GPU and be integrated into our data pipelines. We use six different LFO shapes: cosine (cos), triangle (tri), rectified cosine (rect. cos), inverse rectified cosine (inv. rect. cos), sawtooth (saw), and inverse sawtooth (inv. saw). The LFO parameters of the module are phase, rate, and shape and the effect parameters are min. delay, delay width, feedback, depth, and mix. For the flanger effect we set the minimum delay to 0-1 ms whereas for a chorus effect we set it to 10-20 ms. We also use a modified version of the phaser provided in `Pedalboard`[2], which allows us to specify the LFO phase, while its shape remains restricted to a cosine waveform. Its LFO parameters are phase and rate and its effect parameters are center frequency, feedback, depth, and mix.

Dataset — We use the fourth subset of the IDMT-SMT-Guitar [30] dataset, which contains 64 short electric guitar pieces grouped by genre. Each piece has been recorded at a fast and a slow tempo using three different guitars. We remove the two bars of synchronization tones at the beginning of each piece and split into 75% training and 25% validation sets across the 64 unique songs. This results in 154 min of audio in the training set and 50 min of audio in the validation set. We generate LFO signals with random phase, shape, and rate between 0.5 and 3 Hz and then apply the three audio effects to random 2-second chunks of the dataset while uniformly sampling the effect parameters within their usable ranges.

Training — The input to LFO-net is a Mel spectrogram with 1024 FFT size, 256 sample hop length, 256 Mel bins, and a sample rate of 44.1 kHz. The model consists of 6 convolutional blocks, each with 64 channels, a kernel size of 5 by 13, and a frequency max-pooling and temporal dilation factor of 2. As a result, the receptive field of the network along the time axis spans 2 seconds and outputs 345 frames given 88200 input samples. SpecAugment of 25% is applied during training to both the frequency and time axes. The model contains 1.3 M parameters.

Evaluation — During evaluation of LFO-net, we smooth the signal using a 4th order moving average filter and keep phase, shape, and rate of the LFO signal random. We define two different effect parameter configurations to compare against: "fixed params" and "varying params", summarized in Table 1. We evaluate on 1000 random 2-second non-silent chunks of the dataset. As a baseline, we assume an experienced audio engineer could correctly guess the shape of the LFO signal, whether it's going up or down, and the approximate rate of modulation from listening to the wet audio. We define this as an LFO signal with the correct shape, a random phase error of up to 50%, and a random rate error of up to 25%.

[2] https://github.com/spotify/pedalboard

4.2. Unseen Audio Sources

We evaluate the LFO-net on five unseen datasets processed with the `Pedalboard` phaser and our flanger/chorus implementation using the same setup described in the previous experiment (Section 4.1). These datasets are guitar, bass/double bass, and keyboard audio from MedleyDB 2.0 [31], drums from the IDMT-SMT-Drums [32] dataset, and vocals from VocalSet [33].

4.3. Quasiperiodic, Combined, and Distorted Modulations

Irregular LFO shapes can greatly expand the creative possibilities of an effect and are commonplace in virtual synthesizers. Furthermore, the internal LFOs of analog audio effects are imperfect and can drift or become distorted. As a result, we test the ability of our LFO extraction model to generalize to irregular LFO shapes. We generate quasiperiodic LFO signals by randomly stretching each cycle of a periodic modulation by 10 - 33.33%. We generate combined LFO signals by swapping out random cycles of a periodic modulation with a different shape. We try combining all six shapes randomly together and the four symmetrical shapes (no sawtooth and inverse sawtooth). Finally, we distort LFO signals via exponentiation, which makes different sections of the signal more concave or convex. Figure 5 shows examples of these three types of irregular LFO signals. We then evaluate on the test dataset using the "fixed params" evaluation configuration.

4.4. Latent Space Visualization

In order to see whether the model learns meaningful representations in its latent space we generate three different visualizations. We perform inference on 200 samples from the validation dataset while changing one variable and keeping all others fixed. A single 64-dimensional latent vector is obtained by taking the average across the output frames of the final convolutional block in the model. We then produce a 2d visualization of the vectors using principle component analysis (PCA). We explore how the rate and shape of the LFO signal are encoded as well as the difference between the phaser, flanger, and chorus effects.

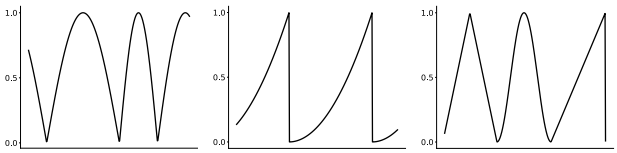


Figure 5: *Examples of quasiperiodic (left), distorted (center), and combined (right) LFO shapes.*

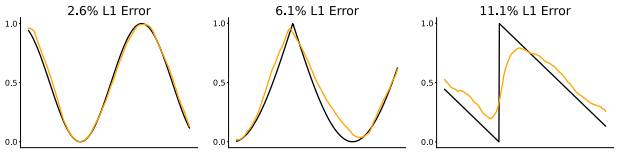


Figure 6: *Examples of 3%, 6%, and 11% extracted LFO L_1 errors.*

4.5. Unseen Analog and Digital Effects

Our final experiment evaluates whether LFO-net can be applied to unseen analog and digital effects and then be used to condition and train an effect emulation model. We use the EGFxSet dataset [34], which consists of five-second long recordings of single electric guitar notes processed with an MXR Phase 45 phaser pedal, a Mooer E-Lady flanger pedal, and a Boss CE-3 chorus pedal. Referencing the datasheets of these effects, we established that all three effects use a rounded triangle LFO shape. We peak normalize the input chunks of audio since the volume levels differ significantly between the wet and dry audio pairs. We apply all post-processing steps described in Section 3.1 during training and inference when extracting the LFO signal (8th order moving average for smoothing) and use a 70/18/12% train-val-test split.

For digital effects we use the MeldaProduction MPhaser and MFlanger plugins[3]. These effects give the user control over the LFO signal and enable combined and irregular LFO signals to be drawn in the user interface. We test two scenarios. First, we consider modeling a phaser and flanger effect with an irregular LFO signal and then with a quasiperiodic LFO signal. For the irregular case we define a skewed sinusoidal LFO shape as shown in Figure 7 at a frequency of 0.75 Hz. For the quasiperiodic case we start with a triangle shape and automate the rate of the LFO from 0.5 Hz to 2.0 Hz and back every 4 seconds. We apply both effects to 8 minute training, 2.5 minute validation, and 2 minute test sets from the fourth subset of the IDMT-SMT-Guitar dataset. During post-processing for the irregular case, we omit step 2 (stretching) to preserve the original shape of the extracted LFO signal.

Since we do not have access to the internal LFO signal for these effects, we first confirm visually that the LFO extraction model is able to output similar LFO signals when applied to these unseen effects. We then use it to train one effect model LSTM with 64 hidden units for each of the seven analog and digital effect configurations defined previously that learns to reconstruct the wet audio given the dry audio and the extracted LFO signal. As a baseline we also train effect models conditioned on a randomly generated LFO with a triangle shape and 0-25% frequency error for the analog effects, a triangle shape and random frequency between 0.5 - 2.0 Hz for the quasiperiodic experiment, and a cosine shape and 0-25% frequency error for the irregular LFO signal experiment.

[3] https://www.meldaproduction.com/effects/free

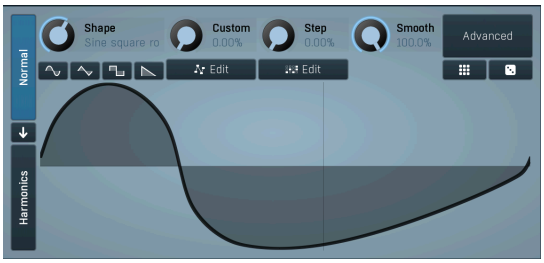


Figure 7: *Skewed sinusoidal LFO shape used in the Melda Phaser and Flanger irregular LFO effect modeling experiments.*

5. RESULTS

5.1. Modulation Extraction

Table 2 summarizes the ability of the model to extract LFO signals from the test dataset. Figure 6 provides a visual reference for the reconstruction quality corresponding to different L_1 error values. We find that an L_1 error of less than 5% corresponds to very accurate extraction with less than 10% error still being acceptable. We notice that the model struggles most with the asymmetrical sawtooth shapes. This is likely due to the waveform containing sharp edges, which can be difficult to reconstruct. We also observe that the model is better at extracting the LFO from the phaser, and worse at extracting the LFO from the chorus. This matches our intuition since the phaser is limited to a cosine LFO shape and because the chorus effect contains the largest varying delay which results in the greatest change in the wet audio spectrogram compared to the flanger. Finally, there is no difference in model performance when the parameters are fixed or varying across their entire usable ranges, thus highlighting the learning capabilities of the proposed LFO model architecture. The baseline consistently results in very large errors due to the fact that small differences in phase and frequency can cause the baseline and ground truth signal to drift apart. We also experimented with extracting the LFO signal from just the wet audio (no dry audio channel) and found that this resulted in an approximately 3% increase in the L_1 error.

5.2. Unseen Audio Sources

We find the model generalizes well to unseen data processed with our three training effects. From Table 3 we see that LFO-net performs just as well or even better on the unseen guitar, bass, and keys datasets. Performance on vocals is also only marginally worse. We expect drums to be the most challenging to extract LFO signals from due to the less tonal and dense onsets and the results match this intuition with extraction ability becoming worse for the flanger and chorus effects on the drums dataset. Varying parameters also results in a very small reduction in performance compared to fixed parameters.

5.3. Quasiperiodic, Combined, and Distorted

The quasiperiodic, distorted, and combined LFO signal results are contained in Tables 4 and 5. The ability to extract quasiperiodic signals is only slightly reduced when compared to periodic signals with the chorus and asymmetrical shapes appearing more challenging than the flanger and symmetrical shapes. This implies the system could be used to obtain an LFO signal for non-periodic audio effects.

Table 2: *LFO extraction evaluation metrics.*

		L_1 Error (%)		
Effect	LFO Shape	Fixed	Varying	Baseline
Phaser	Cosine	1.8%	2.1%	32%
Flanger	Cosine	1.9%	1.9%	32%
	Triangle	2.2%	2.3%	27%
	Rect. Cosine	2.2%	2.1%	28%
	Inv. Rect. Cos.	1.9%	2.0%	28%
	Saw	4.5%	4.5%	27%
	Inv. Saw	4.9%	4.7%	27%
	All	2.9%	2.9%	28%
Chorus	Cosine	3.6%	2.9%	32%
	Triangle	3.1%	3.3%	27%
	Rect. Cosine	2.7%	2.9%	28%
	Inv. Rect. Cos.	2.9%	2.9%	28%
	Saw	8.0%	6.9%	27%
	Inv. Saw	8.5%	7.3%	27%
	All	4.7%	4.3%	28%
All	All	3.1%	3.1%	29%

Table 3: *LFO extraction metrics for unseen datasets.*

		L_1 Error (%)			
Dataset	Params	Phaser	Flanger	Chorus	All
MDB Guitar	Fixed	1.8%	2.8%	4.7%	3.1%
	Varying	1.8%	2.8%	4.9%	3.2%
MDB Bass	Fixed	1.9%	2.4%	4.3%	2.9%
	Varying	2.3%	2.6%	4.7%	3.2%
MDB Keys	Fixed	1.8%	2.5%	4.2%	2.8%
	Varying	2.3%	2.5%	4.0%	2.9%
IDMT Drums	Fixed	1.9%	5.3%	12.2%	6.5%
	Varying	2.7%	5.8%	11.3%	6.6%
Vocalset	Fixed	2.8%	4.3%	5.4%	4.2%
	Varying	2.7%	4.2%	5.8%	4.2%

Distorted inverse rectified cosine, saw, and inverse saw are also difficult for LFO-net to extract. We believe this is because the inverse rectified cosine shape becomes closer to a square wave at the troughs when exponentiated which results in a constant delay and less sweeping patterns in the spectrum to analyze. Similarly, the saw and inverse saw shapes become even more jagged at the corners, thus making reconstruction more challenging, especially at higher LFO rates. Finally, we found that LFO-net is better at reconstructing random combinations of the LFO shapes when the asymmetrical ones are omitted. We believe this is due to the harsh discontinuities that can be introduced by combining sawtooth waves with the other symmetrical waves. Our results indicate that the model can extract symmetrical modulation shapes well, even when each period consists of a different shape.

Table 4: *LFO extraction metrics for quasi. and distorted signals.*

		L_1 Error (%)			
Effect	LFO Shape	Quasi.	Base.	Dist.	Base.
Flanger	Cosine	3.3%	32%	3.4%	33%
	Triangle	3.6%	28%	2.4%	30%
	Rect. Cosine	3.7%	28%	1.9%	32%
	Inv. Rect. Cos.	3.3%	29%	8.1%	28%
	Saw	5.8%	27%	13%	32%
	Inv. Saw	6.5%	28%	13%	31%
	All	4.5%	29%	6.7%	31%
Chorus	Cosine	4.7%	32%	4.6%	33%
	Triangle	5.3%	28%	3.1%	30%
	Rect. Cosine	4.9%	28%	3.6%	32%
	Inv. Rect. Cos.	4.3%	29%	8.7%	28%
	Saw	10%	27%	16%	32%
	Inv. Saw	11%	28%	16%	31%
	All	7.0%	29%	8.5%	31%
Both	All	5.8%	29%	7.6%	31%

Table 5: *LFO extraction metrics for combined modulations.*

		L_1 Error (%)	
Effect	LFO Shapes	Combined	Baseline
Flanger	Symmetrical	4.7%	33%
	All	9.4%	34%
Chorus	Symmetrical	6.1%	33%
	All	11.2%	34%
Both	Symmetrical	5.4%	33%
	All	10.3%	34%

5.4. Latent Space Visualization

The latent space visualizations for changing LFO shape, effect, and rate are shown in Figures 8, 9, and 10, respectively. The latent space decouples for all three visualizations with the relationship between different LFO shapes being encoded by the distance of their clusters in the latent space. Opposite pairs of shapes (i.e. saw / inverse saw and rect. cos. / inv. rect. cos.) are separated by a large distance and similar shapes like triangle and cosine are close together. Similarly, the three different effects decouple in the latent space with chorus and flanger having more overlap since they are identical in implementation, but with different delay amounts. We expect the phaser effect to be the most distinct since it is a unique implementation. Finally, the LFO rate visualization displays a clear relationship between the frequency of the LFO and position in the latent space, with high frequencies becoming more densely clustered together.

5.5. Unseen Analog and Digital Effects

We are able to use LFO-net to model unseen analog and digital audio effects using the effect model described in Section 3.2. Figure 11 shows some examples of the extracted LFO signals from the different effects. For the EGFx analog effects dataset, we see best results on the phaser effect, followed by the chorus, and then the

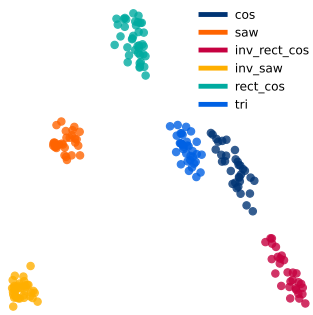


Figure 8: *LFO shape*

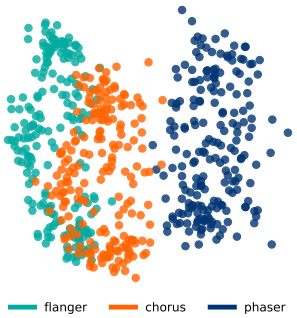


Figure 9: *LFO effect type*

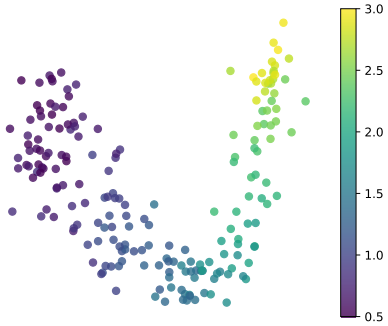


Figure 10: *LFO rate*

Table 6: *Unseen effect evaluation results.*

		Audio Error		Baseline Error	
Effect	LFO Shape	L_1 (%)	ESR	L_1 (%)	ESR
EGFx Phaser	Analog Tri.	3.5%	0.42	6.1%	0.78
EGFx Flanger	Analog Tri.	5.8%	0.94	5.9%	0.95
EGFx Chorus	Analog Tri.	5.0%	0.59	6.6%	0.82
Melda Phaser	Quasi. Tri.	1.4%	0.21	2.7%	0.61
	Irregular	0.76%	0.08	3.0%	0.78
Melda Flanger	Quasi. Tri.	2.3%	0.13	5.3%	0.51
	Irregular	2.9%	0.18	5.2%	0.45

flanger, which is not able to be modeled effectively. We found this dataset to be challenging due to large differences in power supply noise between dry and wet audio pairs, making it difficult to interpret the error metrics and forcing the LSTM to learn to model these differences as well. Despite this, the phaser is able to be modeled and sounds close to the wet audio from informal listening. We provide audio samples in the supplemental material.

The chorus effect is not modeled very well, but in our initial experiments we found that the LSTM effect model is unable to learn chorus effects, even when presented with the ground truth LFO signal, due to the long delays they make use of. As a result, we are surprised to see that the chorus model performs better than the baseline and is sometimes able to match the volume envelope of the wet audio. We also notice that the flanger appears to have two modulations occurring in its spectrogram. LFO-net is able to reliably extract one of them, but this is insufficient for modeling the effect. We believe extracting multiple modulations from audio is a natural future research direction to continue this work on.

For the Melda digital effects we see that both the irregular and quasiperiodic phaser and flanger effects are able to be captured successfully by the effect model. Our informal listening tests also confirm that they sound close to the target wet audio. The baseline model is able to capture the effects to an extent, but struggles especially with the quasiperiodic and irregular phaser LFO signals. The difference in the final ESR highlights the importance of providing an accurate LFO signal to the effect model.

We plot extracted LFO signals from unseen audio effects in Figure 11. A similar LFO shape to the one shown in Figure 7 is extracted for the flanger, but for the phaser it is extracted as two individual rounded peaks, one taller than the other. Since the irregular phaser is able to be modeled with a lower ESR than the irregular flanger, this indicates that this may be an artifact of the internal implementation of the Melda phaser, or that the exact LFO shape may not be required to successfully model an LFO-driven effect. We consider this another interesting future research direction.

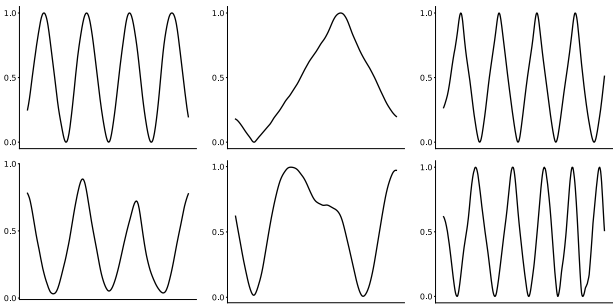


Figure 11: *Extracted LFO patterns from unseen audio effects.*
Top row: EGFx Phaser, Flanger, Chorus
Bottom row: Melda Phaser Irregular, Flanger Irregular, Quasi.

6. CONCLUSIONS

In this work, we propose a system that extracts arbitrary LFO signals from processed audio for multiple LFO-driven audio effects (phaser, flanger, and chorus), parameter settings, and instrument configurations. Our approach does not impose any restrictions on the LFO shape, which allows our neural network architecture to generalize to quasiperiodic, combined, and distorted modulation signals. We test our pretrained network on LFO extraction from a multitude of unseen audio sources, including guitar, bass, keyboards, drums, and singing voice. We show through a visualization of the latent space that the network learns meaningful representations of the different modulation shapes, rates, and effects. Finally, we demonstrate that our pretrained extraction network enables end-to-end modeling of unseen analog and digital LFO-driven audio effects when coupled with a simple processing network, overcoming the need for cumbersome and hand-engineered LFO measurement methods. We find that asymmetrical and discontinuous LFO shapes, such as saw waveforms, are the most difficult to extract and that the effect model cannot learn LFO-driven effects that make use of larger delays or contain multiple modulations. We make our code available and provide the trained audio effect models in a real-time VST plugin.

7. ACKNOWLEDGMENTS

Funded by UKRI and EPSRC as part of the "UKRI CDT in Artificial Intelligence and Music", under grant EP/S022694/1.

8. REFERENCES

[1] Jyri Pakarinen and David T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music Journal*, vol. 33, no. 2, 2009.

[2] David T. Yeh, Jonathan S. Abel, and Julius O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—Part I: Theoretical development," *TASLP*, vol. 18, no. 4, 2009.

[3] Vesa Välimäki, Federico Fontana, Julius O. Smith, and Udo Zölzer, "Introduction to the special issue on virtual analog audio effects and musical instruments," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, 2010.

[4] Antti Huovilainen, "Enhanced digital models for analog modulation effects," in *DAFx*, 2005.

[5] Julian Parker, "A simple digital model of the diode-based ring-modulator," in *DAFx*, 2011, vol. 14.

[6] Felix Eichas, Marco Fink, Martin Holters, and Udo Zölzer, "Physical modeling of the mxr phase 90 guitar effect pedal," in *DAFx*, 2014.

[7] Jaromír Mačák, "Simulation of analog flanger effect using bbd circuit," in *DAFx*, 2016.

[8] Julius O. Smith, "An allpass approach to digital phasing and flanging," in *ICMC*, 1984.

[9] Roope Kiiski, Fabián Esqueda, and Vesa Välimäki, "Time-variant gray-box modeling of a phaser pedal," in *DAFx*, 2016.

[10] Champ Darabundit, Russell Wedelich, and Pete Bischoff, "Digital grey box model of the uni-vibe effects pedal," in *DAFx*, 2019.

[11] Alec Wright and Vesa Välimäki, "Neural modelling of periodically modulated time-varying effects," in *DAFx*, 2020.

[12] Alec Wright and Vesa Välimäki, "Neural modeling of phaser and flanging effects," *Journal of the Audio Engineering Society*, vol. 69, no. 7/8, 2021.

[13] Joseph T. Colonel, Marco Comunità, and Joshua D. Reiss, "Reverse engineering memoryless distortion effects with differentiable waveshaper," in *153rd AES*, 2022.

[14] Marco A. Martínez Ramírez, Emmanouil Benetos, and Joshua D. Reiss, "A general-purpose deep learning approach to model time-varying audio effects," in *DAFx*, 2019.

[15] Marco A. Martínez Ramírez, Emmanouil Benetos, and Joshua D. Reiss, "Deep learning for black-box modeling of audio effects," *Applied Sciences*, vol. 10, no. 2, 2020.

[16] Christian J. Steinmetz and Joshua D. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *152nd AES*, 2022.

[17] Marco Comunità, Christian J. Steinmetz, Huy Phan, and Joshua D. Reiss, "Modelling black-box audio effects with time-varying feature modulation," in *ICASSP*, 2023.

[18] Robert A. Moog, "Voltage controlled electronic music modules," *Journal of the Audio Engineering Society*, vol. 13, no. 3, 1965.

[19] William M. Hartmann, "Flanging and phasers," *Journal of the Audio Engineering Society*, vol. 26, no. 6, 1978.

[20] Michael Stein, Jakob Abeßer, Christian Dittmar, and Gerald Schuller, "Automatic detection of audio effects in guitar and bass recordings," in *128th AES*, 2010.

[21] Marco Comunità, Dan Stowell, and Joshua D. Reiss, "Guitar effects recognition and parameter estimation with convolutional neural networks," *Journal of the Audio Engineering Society*, vol. 69, no. 7/8, 2021.

[22] Reemt Hinrichs, Kevin Gerkens, Alexander Lange, and Jörn Ostermann, "Convolutional neural networks for the classification of guitar effects and extraction of the parameter settings of single and multi-guitar effects from instrument mixes," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2022, no. 1, 2022.

[23] Sungho Lee, Jaehyun Park, Seungryeol Paik, and Kyogu Lee, "Blind estimation of audio processing graph," in *ICASSP*, 2023.

[24] Di Sheng and György Fazekas, "A feature learning siamese model for intelligent control of the dynamic range compressor," in *IJCNN*. IEEE, 2019.

[25] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015.

[27] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager, "Temporal convolutional networks for action segmentation and detection," in *CVPR*, 2017.

[28] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Interspeech 2019*, Sep 2019.

[29] Alec Wright, Eero-Pekka Damskägg, and Vesa Välimäki, "Real-time black-box modelling with recurrent neural networks," in *DAFx*, 2019, pp. 1–8.

[30] Christian Kehling, Jakob Abeßer, Christian Dittmar, and Gerald Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score-and instrument-related parameters," in *DAFx*, 2014.

[31] Rachel M. Bittner, Julia Wilkins, Hanna Yip, and Juan P. Bello, "Medleydb 2.0: New data and a system for sustainable data collection," *ISMIR Late Breaking Demo*, p. 36, 2016.

[32] Christian Dittmar and Daniel Gärtner, "Real-time transcription and separation of drum recordings based on nmf decomposition," in *DAFx*, 2014.

[33] Julia Wilkins, Prem Seetharaman, Alison Wahl, and Bryan Pardo, "Vocalset: A singing voice dataset," in *ISMIR*, 2018.

[34] Hegel Pedroza, Gerardo Meza, and Iran R. Roman, "Egfxset: Electric guitar tones processed through real effects of distortion, modulation, delay and reverb," in *ISMIR Late Breaking Demo*, 2022.

INFORMED SOURCE SEPARATION FOR STEREO UNMIXING — AN OPEN SOURCE IMPLEMENTATION

Sylvain Marchand

L3i
University of La Rochelle
France
sylvain.marchand@univ-lr.fr

Pierre Mahé

LIS
University of Toulon
France
pierre.mahe@univ-tln.fr

ABSTRACT

Active listening consists in interacting with the music playing and has numerous potential applications from pedagogy to gaming, through creation. In the context of music industry, using existing musical recordings (e.g. studio stems), it could be possible for the listener to generate new versions of a given musical piece (i.e. artistic mix). But imagine one could do this from the original mix itself. In a previous research project, we proposed a coder / decoder scheme for what we called informed source separation: The coder determines the information necessary to recover the tracks and embeds it inaudibly (using watermarking) in the mix. The decoder enhances the source separation with this information. We proposed and patented several methods, using various types of embedded information and separation techniques, hoping that the music industry was ready to give the listener this freedom of active listening. Fortunately, there are numerous other applications possible, such as the manipulation of musical archives, for example in the context of ethnomusicology. But the patents remain for many years, which is problematic. In this article, we present an open-source implementation of a patent-free algorithm to address the mixing and unmixing audio problem for any type of music.

1. INTRODUCTION

Active listening of music is an artistic as well as a technological topic of growing interest, that concerns offering listeners the possibility to interact in real time with the music, e.g. to modify the elements, the sound characteristics, and the structure of the music while it is played. This involves, among other examples, advanced remixing processes such as generalized karaoke (muting any musical element, not only the lead vocal track), respatialization, or upmixing. The applications are numerous, from learning / teaching of music to gaming, through new creative processes (disc jockeys, live performers, etc.). In the context of ethnomusicological archiving, the recordings can consist of several tracks, but for the purpose of compatibility, only the mix can often be distributed in the archive. Thus, a technique allowing the user to get access back to the separate tracks from the stereo mix can be very useful.

To get this new freedom, a simple solution would be to give the user access to the individual tracks that compose the mix, by storing them into some multi-track format. This approach has two main drawbacks: First, it leads to larger multi-track files. Second, it yields files that are not compatible with the prevailing stereo standards. Another solution is to perform some blind separation of the sources from the stereo mix. The problem is that even with state-of-the-art blind source separation techniques the quality is usually insufficient and the computation is heavy (see [1]).

In the DReaM project (see [2]), we proposed an Informed Source Separation (ISS) approach (see [3]) to accurately recover the separate tracks from the stereo mix. The present article will focus on this approach only, where the system consists of a coder and a decoder. The coder is used at the mixing stage, where the separate tracks are known. It determines the information necessary to recover the tracks from the mix and embeds it in the mix. In the classic case of Pulse-Code Modulation (PCM), this information is inaudibly hidden in the mix by a watermarking technique. With a legacy system, the coded stereo mix can be played and sounds just like the original, although it includes some additional information. Apart from backward compatibility with legacy systems, a further advantage concerns the fact that the file size stays comparable to the one of the original mix. The decoder performs source separation of the mix with parameters given by the additional information. This ISS approach permits producing good separate tracks, thus enabling active listening applications.

The original target of the DReaM project was the music industry, which turned out to be quite conservative. For instance, the actors of music industry appeared to be reserved with the use of audio formats that are alternative to conventional stereo encoding, hence hindering the development of object-based formats or advanced spatial audio formats such as Ambisonics. Another example is the fact that listeners are considered as (passive) consumers, even if some want to behave as musicians (active listeners, content producers, etc.).

Fortunately, there is some opportunity for the system developed in the project for an application to musical archives (see [4]). Indeed, some recordings contain several tracks, but the diffusion format is still legacy stereo. Thus, having a format backward compatible with standard stereo but allowing to recover the individual tracks present in the mix can be of interest. The DReaM project showed that it is possible. However, since the finality of the project was industrial, the ISS methods were patented. For new – non commercial – applications, a patent-free method was needed. The contribution of the present article is the definition and implementation of such a method.

The remainder of this article is organized as follows. Section 2 presents the DReaM project: its fundamentals and target applications. Section 3 describes the separation / unmixing methods developed in the project, whereas Section 4 introduces a patent-free method: ReaLiTy. Finally, Section 5 draws some conclusions.

The DReaM project was supported by the French ANR *(Agence Nationale de la Recherche)*, from 2009 to 2014 (ANR-09-CORD-006).

2. THE DREAM PROJECT

DReaM is a French acronym for *"le Disque Repensé pour l'écoute active de la Musique"*, which means "the disc thought over for active listening of music". This is the name of an academic project (2009–2014) with industrial finality, coordinated by the first author, and funded by the French National Research Agency (ANR). The project involved academic partners (LaBRI – University of Bordeaux, Lab-STICC – University of Brest, GIPSA-Lab – Grenoble INP, LTCI – Telecom ParisTech, ESPCI – Institut Langevin) together with iKlax Media, a company for interactive music that contributed to the Interactive Music Application Format (IMAF) standard (see [5]).

The origin of the project comes from the observation of artistic practices. More precisely, composers of acousmatic music conduct different stages through the composition process, from sound recording (usually stereophonic) to diffusion (multiphonic). During live interpretation, they interfere decisively on spatialization and coloration of pre-recorded sonorities. For this purpose, the musicians generally use a mixing console to upmix the musical piece being played from an audio CD. This requires some skills, and imposes musical constraints on the piece. Ideally, the individual tracks should remain separate. However, this multi-track approach is hardly feasible with a typical (stereophonic) audio CD.

Nowadays, the audience is more eager to interact with the musical sound. Indeed, more and more commercial CDs come with several versions of the same musical piece. Some are instrumental versions (e.g. for karaoke), other are remixes. The karaoke phenomenon gets generalized from voice to instruments, in musical video games such as *Rock Band*. But in this case, enabling interaction translates to users having to buy a video game, which includes the multi-track recording.

Yet, the music industry seems to be reluctant to releasing the multi-track versions of big-selling hits. The only thing the user can get is a standard CD, thus a stereo mix, or its digital version available for download or streaming, now that the physical version (at least the CD) disappears.

2.1. Objectives

In general, the project aims at solving a so-called inverse problem, to some quality extent, at the expense of additional information. In particular, an example of such an inverse problem can be source separation: recovering the individual source tracks from the given mix.

On the one hand coding the solution (e.g. the individual tracks and the way to combine them) can bring high quality, but with a potentially large file size, and a format not compatible with existing stereo formats. On the other hand the blind approach (without information) can produce some results, but of insufficient quality for demanding applications (see [1]). The blind approach can be regarded as an estimation without information, while coding can be regarded as using information (from each source) without any estimation (from the mix).

The informed approach proposed by DReaM is just in between these two extremes: getting musically acceptable results with a reasonable amount of additional information. The problem is now to identify and encode efficiently this additional information. Remarkably, ISS can thus be seen both as a multi-track audio coding scheme using source separation, or as a source separation system helped by audio coding.

This approach addresses the source separation problem in a coder / decoder configuration. At the coder (see Figure 1), the additional information is estimated from the original source signals before the mixing process and is inaudibly embedded into the final mix. At the decoder (see Figure 2), this information is extracted from the mix and used to assist the separation process.

So, a solution can be found to any problem, thanks to the additional information embedded in the mix.

"There's not a problem that I can't fix,
'cause I can do it in the mix!"
(Indeep – Last Night a DJ Saved my Life)

The original goal of the project was to propose a fully backward-compatible audio-CD permitting musical interaction.

The idea was to inaudibly embed (using a high-capacity watermarking technique) in the audio track some information enabling to some extent the musical decomposition, that is the inversion of the music production chain: dynamics decompression, source separation (unmixing), deconvolution, etc.

With a standard CD player, one would listen to the fixed mix. With an active player however, one could modify the elements and the structure of the audio signal while listening to the music piece.

Now that the music is getting all digital, the consumer gets access to audio files instead of physical media. In this article we will consider only audio files without compression.

2.2. Applications

Active listening (see [6]) amounts to performing various operations that modify the elements and structure of the music signal during the playback of a piece. This process, often simplistically called remixing, includes generalized karaoke, respatialization, or applying certain effects to individual audio tracks (e.g. adding some distortion to an acoustic guitar). The goal is to enable the listener to enjoy freedom and personalizing of the musical piece through various reorchestration techniques. Alternatively, active listening solutions intrinsically provide simple frameworks to the artists to produce different versions of a given piece of music. Moreover, it is an interesting framework for music learning / teaching applications.

2.2.1. Respatialization

The original application was to let the public experience the freedom of composers of electroacoustic music during their live performances: moving the sound sources in the acoustic space. Although changing the acoustical scene by means of respatialization is a classic feature of contemporary art (electroacoustic music), and efforts have been made in computer music to bring this practice to a broader audience (see [7]), the public seems just unaware of this possibility and rather considered as passive consumers by the music industry. However, during the public demonstrations of the DReaM project, we felt that the public was very reactive to this new way of interacting with music, to personalize it, and was ready to adopt active listening, mostly through musical games.

2.2.2. Generalized Karaoke

The generalized karaoke application is the ability to suppress any audio source, either the voice (classic karaoke) or any instrument ("music minus one"). The user can then practice singing or playing

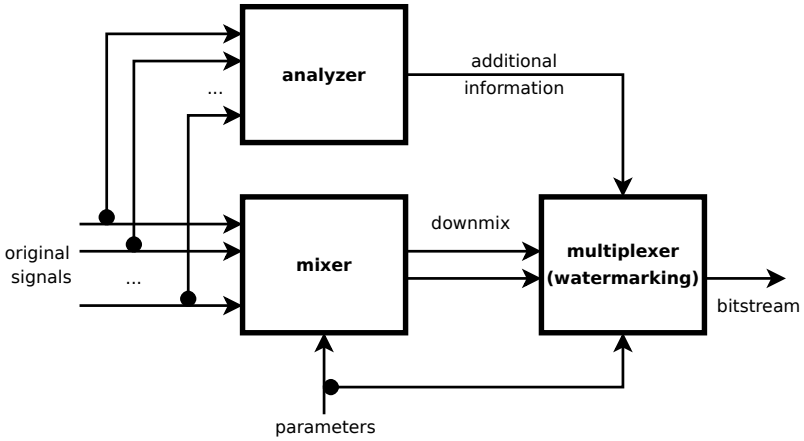


Figure 1: *Architecture of an Informed Source Separation (ISS) coder.*

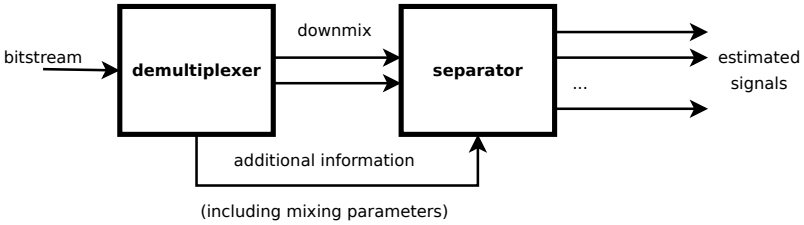


Figure 2: *Architecture of an Informed Source Separation (ISS) decoder.*

an instrument while being integrated in the original mix and not a cover song.

Note that these two applications (respatialization and generalized karaoke) are related, since moving a source far away from the listener will result in its muting, and reciprocally the ability to mute sources can lead to the monophonic case (the spatial image of a single source isolated) where respatialization is much easier (possible to some extent even without recovering the audio object from this spatial image).

2.2.3. Sound Archives

It turns out that the system developed in the project might be very useful for musical archives. Indeed, some recordings contain several tracks, but the diffusion format is still legacy stereo. Thus, having a format backward compatible with standard stereo but allowing to recover the individual tracks present in the mix can be of interest.

3. INFORMED SOURCE SEPARATION METHODS

A stereo (2-channel) mixture $\{y_c(n)\}_{c=1,2}$ will be produced from K source signals $\{x_k(n)\}_{k=1}^{K}$ and panning angles θ_k (which are not azimuths, see [8] for details), the latter leading to a mixing matrix A where A_{ck} denotes the contribution of the kth input source to the cth output channel. In this article, we will consider a simple case where the mixing matrix A is obtained from panning angles θ using Equations (1) and (2)

$$A_{1k} = \sin(\theta_k) \quad (1)$$

$$A_{2k} = \cos(\theta_k) \quad (2)$$

such that $A_{1k}^2 + A_{2k}^2 = 1$ (energy conservation). The values for the panning angles θ will range from 0 (right) to $\pi/2$ radians (left).

Source separation then consists in recovering (estimates of) the source signals x_k from the mix signals y_c, possibly with the help of additional information extracted from x_k (informed approach).

Over the years of the project, several Informed Source Separation (ISS) methods were proposed. More precisely, this section presents the similarities, differences, strengths, and weaknesses of four of them. A detailed technical description or comparison is out of the scope of this article. Instead, we will propose a new – free – method, which is a mix of the original methods. The detailed descriptions of the following methods can rather be found in [9], [10], [8], and [11], while their comparison is done in [12].

The majority of the ISS methods aims at extracting the contribution of each source from each Time-Frequency (TF) point of the mix, at least in terms of magnitude, and sometimes phase too.

3.1. Local Inversion

The first method performs a local inversion (see [9] and [13]) of the mix for each TF point, using the information of the two predominant sources in this point (as well as the knowledge of the mixing matrix). More precisely, at each TF point two sources can be reconstructed from the two (stereo) channels, by a local two-by-two inversion of the mixing matrix. This way, we get estimates of the magnitude and phase of the prominent sources. But the problem is that the remaining $K-2$ sources exhibit a spectral hole (no estimated signal), which is perceived as quite annoying in subjective listening tests (see [8]). Also, this method requires the mixing matrix A to be of rank K.

3.2. Minimum Mean-Square Error Filtering

The second method performs classic Minimum Mean-Square Error (MMSE) filtering (see [10] and [14]) using Wiener filters driven by the information about the power of the sources (as well as the mixing matrix), the corresponding spectrograms being transmitted using either sound (NMF) or image (JPG) compression techniques (see [12] for details). In contrast to the local inversion method, MMSE does not constrain as much the mixing matrix A and is therefore more flexible towards the mixing configurations. The separation quality, however, is much better when A is of rank K.

3.3. Linearly Constrained Spatial Filtering

The third method is called Undetermined Source Signal Recovery (USSR), and performs linearly constrained spatial filtering (see [8] and [15]) using a Power-Constraining Minimum-Variance - (PCMV) beamformer, also driven by the information about the power of the sources (and their spatial distribution) and ensuring that the output of the beamformer matches the power of the sources (additional information transmitted in ERB/dB scales, see Section 4.1.2). In the stereo case, if only two predominant sources are detected, the beamformer is steered such that one signal component is preserved while the other is canceled out. Applying this principle for both signal components results in inverting the mixing matrix (first method). Moreover, dropping the power constraint will turn the PCMV beamformer into an MMSE beamformer (second method). Otherwise, the PCMV beamformer takes advantage of the spatial distribution of the sources to produce better estimates.

3.4. Iterative Phase Reconstruction

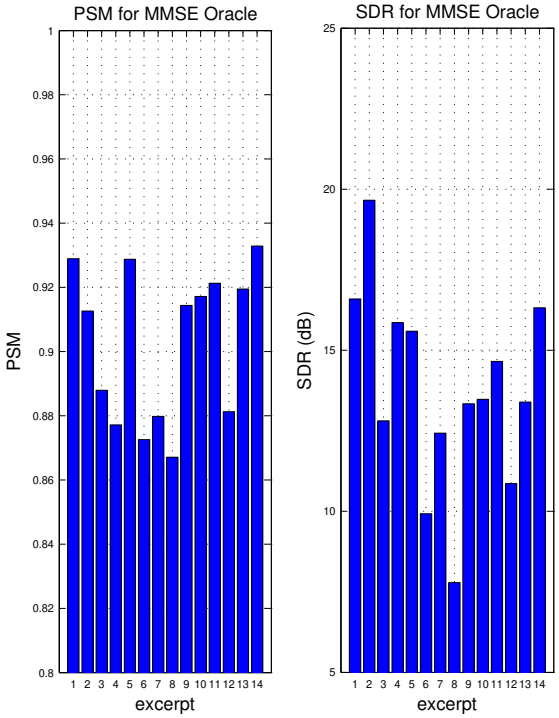
The fourth method performs iterative phase reconstruction and is called IRISS (Iterative Reconstruction for Informed Source Separation), see [11]. It also uses the magnitude of the sources (transmitted in ERB/dB scales) as well as a binary activity map as an additional information to the mix. The main point of the method is to constrain the iterative reconstruction of all the sources so that Equation (5) is satisfied at each iteration very much like the Multiple Input Spectrogram Inversion (MISI) method (see [16]). Contrary to MISI, both amplitude and phase of the STFT are reconstructed in IRISS, therefore the remix error should be carefully distributed. In order to do such a distribution, an activity mask derived from the Wiener filters is used. The sources are reconstructed at the decoder with an initialization conditioned at the coding stage. It is noticeable that this technique was specifically designed for mono mixtures (1-channel), where it gives the best results.

3.5. Evaluation

3.5.1. Performances

The quality performance of the system reaches the needs of many real-life applications (for each of the four methods described above). The comparison of the four original methods can be found in [12], for the linear instantaneous and convolutive case, using either the objective Signal-to-Distortion Ratio (SDR) criterion of BSSEval (see [17]) or the subjective Perceptual Similarity Measure (PSM) of PEMO-Q (see [18]), closer to perception. A set of 14 musical excerpts from the Quaero database has been considered (see [12] for details).

Figure 3 shows the performances of MMSE (Wiener) filtering with access to full information (oracle situation) about sound sources for both subjective (PSM) and objective (SDR) measures. The PSM is often above 0.9 (1 corresponding to perfection), and the SDR is around 15dB (which is quite good).



(a) Subjective results (PSM) (b) Objective results (SDR)

Figure 3: *Performances of MMSE filtering with access to full information (oracle) about sound sources (estimated signals compared to the original signals).*

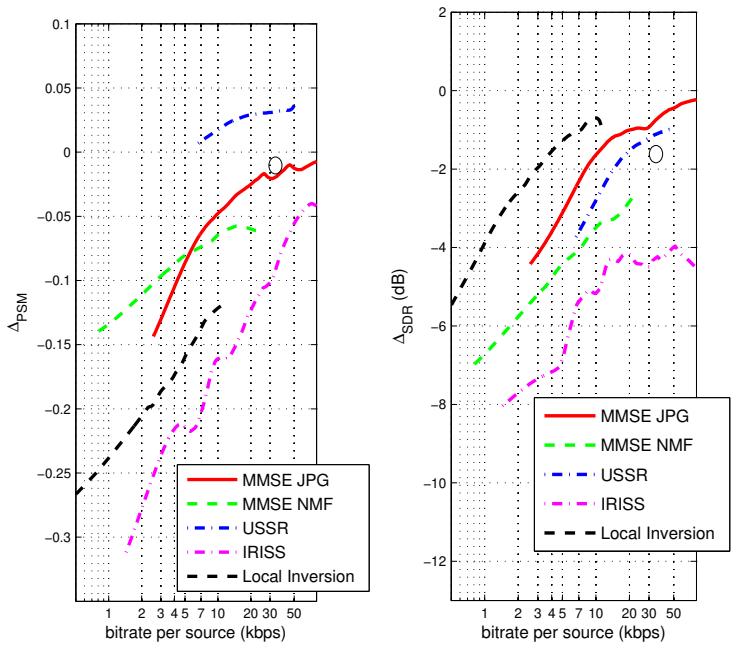
Figure 4 shows the relative performances of the DReaM methods, relatively to the MMSE oracle, as functions of the additional information bitrate. It turns out that the first method (local inversion) exhibits the best objective (SDR) results, while the third method (USSR) exhibits the best subjective (PSM) results; this was also verified in a formal listening test conducted in [8].

It is important to note that the complexity of these methods is low, allowing real time. Moreover, as shown in [12], the typical bitrates for the additional information are approximately 5 to 10 kbits per second for each source, which is quite reasonable.

The problem with these methods is that they are protected with patents.

3.5.2. Patents

The patent of the first method (see [13]) protects the local inversion technique as well as the encoding of the active sources indices. The patent of the second method (see [14]) protects the coding of the additional information, but not Wiener filtering which is a well-known technique. The patent of the third method (see [15]) protects the use of the PCMV beamformer for source separation, whereas the ERB and dB scales used for the additional information reduction are well-known, and also used by the fourth method.



(a) Subjective results (PSM) (b) Objective results (SDR)

Figure 4: *Performances of the DReaM methods (relatively to the MMSE oracle), as functions of the additional information bitrate. The black circle indicates the performance of the proposed ReaLiTy implementation.*

This last method is patent-free, but unfortunately not suitable for stereo mix, with a lower quality and a higher complexity (increased processing time).

Thus, there is room for an efficient patent-free method, with additional information represented in ERB/dB scales and filtering performed using the standard Wiener (MMSE) filtering, provided the watermarking technique used is also patent-free.

This method, called ReaLiTy, is described in the next section. The code is distributed as free software, and comes with a sound example. The performance of the proposed method on that specific example is indicated[1] by a black circle on Figure 4. The subjective result is close to MMSE oracle performance thanks to the high bitrate per source (35 kbps) for the additional information, but stays below this limit unlike USSR (whose filtering is not MMSE). The objective result is comparable to those of USSR and MMSE JPG (the bitrate being unfortunately not suitable for MMSE NMF), although slightly below.

4. REALITY: A FREE IMPLEMENTATION

All the methods developed during the DReaM project are based on a coder / decoder scheme. The coder produces a stereo mix from the K source signals using panning angles, and the decoder recovers (estimates of) these signals from the mix, using the additional information inaudibly embedded by the coder. This section gives the details for the coder and the decoder of the proposed ReaLiTy method, which is patent-free and comes with a free software implementation in Python programming language.

The source signals $\{x_k(n)\}_{k=1}^{K}$ are block-wise time-frequency mapped by means of the Short-Time Fourier Transform (STFT) using Equation (3)

$$X_k(h,b) = \sum_{n=0}^{N-1} x_k(hH+n)w(n)e^{-\jmath 2\pi nb/N} \qquad (3)$$

where $0 \leq b < N$ is the frequency index, N is the frame size, h is the frame index, H is the hop size, and $\jmath$ is the imaginary unit. In practice, for w we use the Hann window of size $N = 2048$, for a sampling frequency $F_s = 44100$Hz. We will allow a 50% overlap, thus $H = N/2$.

4.1. Coder

As shown on Figure 1, the coder consists of three building blocks: mixer, analyzer, and multiplexer.

4.1.1. Mixer

The sources are defined as K mono signals x_k of same length L, with sampling rate F_s. The mixer takes these original signals x_k and panning angles θ_k and produces a stereo (2-channel) mixture $\{y_c(n)\}_{c=1,2}$.

[1] However this is only an indication, since this figure was originally generated for [12] with different sound excerpts, and unfortunately more than 10 years after it was impossible to get access to them. More precisely, if the database is known, the excerpts were not specified and this information is apparently lost now. Running a new comparison on different excerpts turned out to be also impossible, since we do not have access to the (patented) code of the original methods anymore.

We first consider linear and time-invariant mixing systems. Formally, we suppose that each source signal x_k is mixed into each destination channel c through the use of some mixing coefficients a_{ck}, leading to Equation (4).

$$y_c(n) = \sum_{k=1}^{K} a_{ck} \cdot x_k(n) \tag{4}$$

Since the mixing coefficients are constant over time the mixing is said to be linear instantaneous.

If the mixing coefficients a_{ck} are replaced by filters, and the product in Equation (4) is replaced by the convolution, the mixing is then said to be convolutive. We can easily handle this case (see [12]) with the STFT representation if the length of the mixing filters is sufficiently short compared to the window size, thanks to the convolution theorem, with Equation (5)

$$Y_c(h,b) \approx \sum_{k=1}^{K} A_{ck}(b) \cdot X_k(h,b) \tag{5}$$

where $A_{ck}(b)$ is understood as the frequency response of filter a_{ck} at frequency index b. When the mixing process is linear instantaneous and time invariant, A_{ck} is constant and the $2\times K$ matrix A is called the mixing matrix. The mixing process can thus be rewritten as matrix multiplication in Equation (6)

$$Y \approx A \cdot X \tag{6}$$

where $Y = [Y_1, Y_2]^\top$ and $X = [X_1, \cdots, X_K]^\top$ are column vectors respectively gathering all mixtures and sources at the time-frequency (TF) point (h, b).

4.1.2. Analyzer

The analyzer also takes the original signals x_k as inputs, to compute the additional information to be embedded in the mix.

At the origin of the DReaM project, this information consisted of the indices of the two most prominent sources, that is the two sources with the highest energy at the considered TF point, since this information can be used to solve the interference of the sources at this point, by local inversion (see [9]). This information can be efficiently coded with $\lceil \log(K(K-1)/2) \rceil$ bits per TF point. But the local inversion technique is patented (see [13]).

The information about the power spectrum of each source turned out to be extremely useful and more general. Indeed, if we know the power of all the sources, we can determine the two predominant sources. We can also derive activity patterns for all the sources. As shown in [10], this information can be coded using sound or image compression techniques. The problem, again, is that it is patented (see [14]).

Let us consider the instantaneous Power Spectral Density (PSD) $E_k(h, b)$, calculated according to Equation (7).

$$E_k(h,b) = |X_k(h,b)|^2 \tag{7}$$

Fortunately, this information can efficiently be coded on a double-logarithmic scale using simple psychoacoustic considerations. More precisely, a significant reduction of this information can be achieved in two ways: first, by reducing the frequency resolution of the PSDs $E_k(h, b)$ in approximation of the critical bands (see [19]), and second, by quantizing the obtained PSD values $\hat{E}_k(h, z)$ with a step size equal to some value Δ, which is put in relation to an appropriate psychoacoustic criterion.

Scaling. The peripheral auditory system is usually modeled as a bank of overlapping bandwidth filters, the auditory filters, which possess an Equivalent Rectangular Bandwidth (ERB). The scale that relates the center frequency of auditory filters to units of the ERB is the ERB-rate scale. Using the ERB-rate function of [20] we can define a relation between the frequency index b and the critical-band index z_b by Equation (8)

$$z_b = \lfloor 21.4 \log_{10}(4.37 b(F_s/1000)/N + 1) \rfloor \tag{8}$$

where $\lfloor \cdot \rfloor$ is the floor function. The zth critical-band value of the approximate PSDs is then calculated as the arithmetic mean between $\text{lower}(z) = \inf\{b\colon z_b = z\}$ and $\text{upper}(z) = \sup\{b\colon z_b = z\}$ according to Equation (9).

$$\bar{E}_k(h,z) = \frac{1}{\text{upper}(z) - \text{lower}(z) + 1} \sum_{b=\text{lower}(z)}^{\text{upper}(z)} E_k(h,b) \tag{9}$$

Recovering the Short-Time PSDs (STPSDs) in linear scale (to the resolution of the STFT) is as easy as Equation (10).

$$E_k(h,b) \approx \bar{E}_k(h,z_b) \tag{10}$$

Quantization. Furthermore, under the assumption that the the minimum just-noticeable-difference level and so the maximum allowed quantization error is 1dB (see [19]), the quantization step size Δ is chosen as 2dB, and the irrelevancy-reduced PSD values are obtained from the uniform quantizer in Equation (11)

$$\bar{E}_k^{\Delta}(h,z) = [5 \log_{10} \bar{E}_k(h,z)] \tag{11}$$

where $[\cdot$ denotes the round-to-nearest rounding function. Note that replacing 5 by 10 in the previous equation would lead to the classic dB scale. Recovering the STPSD values in linar scale (by "dequantization") is as easy as Equation (12).

$$\bar{E}_k(h,z) \approx 10^{\bar{E}_k^{\Delta}(h,z)/5} \tag{12}$$

4.1.3. Multiplexer

The multiplexer takes the downmix y_c as well as the mixing parameters (panning angles) and the additional information as inputs, in order to produce a bitstream: the resulting stereo sound file.

The panning angles θ_k are simply rounded to the nearest integer value and quantized on 8 bits. The additional information consists of the STPDSs of the K source signals, $\bar{E}_k^{\Delta}$, quantized on B_S bits. Increasing B_S will lead to a better audio quality, but at the expense of a greater number of bits necessary at each STFT frame to encode the full additional information (including mixing parameters): $(8 + W \times B_S) \times K$ bits, where W is the number of bands of the ERB scale. In our implementation, we use $W = 136$, $B_S = 6$, for $K = 5$ sources, for a total of 4120 bits per frame.

These bits will be embedded inaudibly using watermarking. To avoid any patent, we will consider the most basic technique consisting in hiding the data in the Least Significant Bits (LSBs) of the downmix samples.

Since this downmix is stereo (2 channels), and each STFT frame consists of N samples, with a hop size of $H = N/2$ meaning 50% overlap, if we use B_C LSB bits per channel we can hide $B_C \times N$ bits per frame. In our implementation we use $N = 2048$ and $B_c = 3$, for a total of 6144 bits available.

With these settings, we could handle up to 7 sources. To go further, one can increase B_C or decrease B_S. In the first case, the quality of the mix will begin to degrade (the watermark becoming audible), and the second case the quality of the estimated source signals will degrade. Using a higher capacity watermarking or entropy coding for the data could also be solutions.

4.2. Decoder

As shown on Figure 2, the decoder consists of two building blocks: demultiplexer and separator.

4.2.1. Demultiplexer

From the input bitstream, the demultiplexer has to recover the downmix plus the additional information (including mixing parameters). As an approximation, the downmix y_c will be the stereo signal of the input. The additional information (θ_k and $\bar{E}_k^{\Delta}$) is simply extracted from the LSBs of the samples of this input signal.

4.2.2. Separator

The core block of the decoder is the separator, aiming at estimating the K original signals x_k from the downmix y_c and this additional information, consisting of the K source STPSDs in ERB/dB scale ($\bar{E}_k^{\Delta}$) together with the mixing parameters (θ_k), leading to the mixing matrix A (see Section 3).

Filtering. As shown in Section 3.5, if one wants to maximize the objective quality (SDR), one could use the first DReaM method (local inversion) but then mess with a patent ([13]), and if one wants to maximize the subjective quality (PSM), one could use the third method (USSR) but then mess with another patent (see [15]). The Wiener (MMSE) filtering used by the second method is a good compromise, and patent-free. This filtering is done according to Equation (13).

$$\hat{X}_k(h,b) = \sum_{c=1}^{2} Y_c(h,b) \cdot \frac{A_{ck} \cdot E_k(h,b)}{\sum_{s=1}^{K} A_{cs} \cdot E_s(h,b)} \qquad (13)$$

Adjusting. Since the STPSDs of the sources E_k are known, we can scale the estimated source spectra $\hat{X}_k$ to adjust these STPSDs. The spectra $\left\{\hat{X}_k(h,b)\right\}_{k=1}^{K}$ are then transformed back to the time domain to get the signals $\{\hat{x}_k(n)\}_{k=1}^{K}$ using the inverse STFT (ISTFT) with a classic overlap-add (OLA) procedure, with 50% overlap ($H = N/2$), the Hann window used for the STFT ensuring perfect reconstruction of the signals in this case.

In practice, it could be a good idea (in case of non-linear spectral processing) to apply the window w at both STFT and ISTFT stages, using the square root of the Hann window (so that the product of the windows of the two stages results in the original Hann window). This is done is our free software implementation[2], programmed in Python.

5. CONCLUSION

Originally thought as a way to interact with the music signal through its real-time decomposition / manipulation / recomposition, in the DReaM project the emphasis has been laid on the mixing stage, leading to source separation / unmixing techniques using additional information to improve the quality of the results. DReaM can also be regarded as a multi-track coding system based on source separation.

The initial aim was to give freedom to the listener, in the context of music industry, but artistic as well as industrial problems arose. For example, the artwork is sacred – it shall not be "altered". Also, the method requires studios recordings – involving copyright issues with studios / producers / majors. Finally, the method requires mastering the whole production chain – meaning entering Digital Audio Workstations (DAWs), which can hardly be done. But DReaM has shown the possibility to produce a mix allowing source separation, backward compatible with legacy stereo, thus without the need of some multi-track format. Unfortunately, the industrial finality of the project led to patents on the original methods.

In this article, we proposed ReaLiTy, a patent-free version of the system. It is based on well-known techniques such as LSB watermarking, ERB/dB scale, or Wiener filtering. A free software implementation in Python programming language is available online. We hope that it could be used e.g. for storing / spreading multi-track sound archives within the standard stereo format, or could serve as a basis for future research.

6. ACKNOWLEDGMENTS

"You may say I'm a dreamer, but am not the only one." (John Lennon – Imagine). Thus, the first author would like to thank all the members of the project consortium for having made the DReaM come true. Thanks must also go to Jim Beauchamp, for his interest in music signal processing and his invitations to meetings of the Acoustical Society of America (ASA), and to Rolf Bader, who saw there the opportunity of the DReaM methods for academic applications, and allowed the first version of ReaLiTy to be developed in Hamburg, Germany. Stanislaw Gorlow provided the Matlab implementation of the USSR method, which served as a basis for this first Python development by Pierre Mahé, which in turn served for the final version of ReaLiTy by Sylvain Marchand.

[2] ReaLiTy:
`https://www.sylvain-marchand.info/ReaLiTy/`

7. REFERENCES

[1] Pierre Comon and Christian Jutten, Eds., *Handbook of Blind Source Separation – Independent Component Analysis and Applications*, Academic Press, 2010.

[2] Sylvain Marchand, Roland Badeau, Cléo Baras, Laurent Daudet, Dominique Fourer, Laurent Girin, Stanislaw Gorlow, Antoine Liutkus, Jonathan Pinel, Gaël Richard, Nicolas Sturmel, and Shuhua Zhang, "DReaM: A novel system for joint source separation and multi-track coding," in *Proceedings of the 133rd AES Convention*, San Francisco, California, USA, October 2012.

[3] Kevin H. Knuth, "Informed source separation: A Bayesian tutorial," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Antalya, Turkey, September 2005.

[4] Sylvain Marchand, "Spatial manipulation of musical sound: Informed source separation and respatialization," in *Computational Phonogram Archiving (Current Research in Systematic Musicology)*, Rolf Bader, Ed., pp. 175–190. Springer Nature, Switzerland, 2019.

[5] ISO/IEC, *ISO/IEC 23000-12, Information technology – Multimedia application format (MPEG-A) – Part 12: Interactive Music Application Format (IMAF)*, 2010.

[6] Philippe Lepain, "Écoute interactive des documents musicaux numériques," in *Recherche et applications en informatique musicale*, Marc Chemillier and François Pachet, Eds., pp. 209–226. Hermes, Paris, France, 1998, In French.

[7] François Pachet and Olivier Delerue, "A constraint-based temporal music spatializer," in *Proceedings of the ACM Multimedia Conference*, Brighton, United Kingdom, 1998.

[8] Stanislaw Gorlow and Sylvain Marchand, "Informed audio source separation using linearly constrained spatial filters," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 1, pp. 3–13, January 2013.

[9] Mathieu Parvaix and Laurent Girin, "Informed source separation of linear instantaneous under-determined audio mixtures by source index embedding," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1721–1733, August 2011.

[10] Antoine Liutkus, Jonathan Pinel, Roland Badeau, Laurent Girin, and Gaël Richard, "Informed source separation through spectrogram coding and data embedding," *Signal Processing*, vol. 92, no. 8, pp. 1937–1949, August 2012.

[11] Nicolas Sturmel and Laurent Daudet, "Informed source separation using iterative reconstruction," *IEEE Transactions on Audio, Speech, and Language Processing*, , no. 1, pp. 178–185, January 2013.

[12] Antoine Liutkus, Stanislaw Gorlow, Nicolas Sturmel, Shuhua Zhang, Laurent Girin, Roland Badeau, Laurent Daudet, Sylvain Marchand, and Gaël Richard, "Informed audio source separation: A comparative study," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Bucharest, Romania, August 2012.

[13] Mathieu Parvaix, Laurent Girin, Jean-Marc Brossier, and Sylvain Marchand, *Method and Device for Forming a Mixed Signal, Method and Device for Separating Signals, and Corresponding Signal*, FR2944403, EP2417597, US20120203362, KR1020120006050, JP2012523579, WO2010116068 (patents), France, Europe, United States, Korea, Japan, World, 2010.

[14] Laurent Girin, Antoine Liutkus, Gaël Richard, and Roland Badeau, *Method and Device for Forming a Digital Audio Mixed Signal, Method and Device for Separating Signals, and Corresponding Signal*, FR2966277, EP2628154, US20140037110, WO2012049176 (patents), France, Europe, United States, World, 2012.

[15] Sylvain Marchand and Stanislaw Gorlow, *Method and Device for Separating Signals by Minimum Variance Spatial Filtering Under Linear Constraint*, FR2996043, EP2901447, US20150243290, JP2015530619, WO2014048970 (patents), France, Europe, United States, Japan, World, 2014.

[16] David Gunawan and Deep Sen, "Iterative phase estimation for the synthesis of separated sources from single-channel mixtures," *IEEE Signal Processing Letters*, vol. 17, no. 5, pp. 421–424, May 2010.

[17] Emmanuel Vincent, Rémy Gribonval, and Cédric Févotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, July 2006.

[18] Rainer Huber and Birger Kollmeier, "PEMO-Q – a new method for objective audio quality assessment using a model of auditory perception," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 6, pp. 1902–1911, November 2006.

[19] Hugo Fastl and Eberhard Zwicker, *Psychoacoustics: Facts and Models*, Springer, third edition, 2007.

[20] Brian R. Glasberg and Brian C. J. Moore, "Derivation of auditory filter shapes from notched-noise data," *Hearing Research*, vol. 47, no. 1-2, pp. 103–138, August 1990.

FEATURE-BASED DELAY LINE USING REAL-TIME CONCATENATIVE SYNTHESIS

Niccolo Abate and Brian Hansen

Department of Computational Media
University of California, Santa Cruz
Santa Cruz, California, USA
nlabate@ucsc.edu | brmhanse@ucsc.edu

ABSTRACT

In this paper we introduce a novel approach utilizing real-time concatenative synthesis to produce a Feature-Based Delay Line (FBDL). Expanding upon the concept of a traditional delay, its most basic function is familiar – a dry signal is copied to an audio buffer whose read position is time shifted producing a delayed or "wet" signal that is then remixed with the dry. In our implementation, however, the traditionally unaltered wet signal is modified such that the audio delay buffer is segmented and concatenated according to specific audio features. Specifically, the input audio is analyzed and segmented as it is written to the delay buffer, where delayed segments are matched to a target feature set, such that the most similar segments are selected to constitute the wet signal of the delay. Targeting methods, either manual or automated, can be used to explore the feature space of the delay line buffer based on dry signal feature information and relevant targeting parameters, such as delay time. This paper will outline our process, detailing important requirements such as targeting and considerations for feature extraction and concatenation synthesis, as well as discussing use cases, performance evaluation, and commentary on the potential of advances to digital delay lines.

1. INTRODUCTION

1.1. Concatenative Synthesis

Concatenation synthesis, thought of as directed granular synthesis [1] and referred to as its natural successor, is a type of synthesis where small segments of audio are selected according to their descriptors and concatenated together to create a unique audio stream. The origins of the idea are espoused in Iannis Xenakis' Formalized Music in which he describes a stochastic approach of concatenating small segments of audio together to create new sounds [2]. Due to processing limitations of the time, the analysis was a time consuming endeavor that had to take place pre-synthesis and was thus a major limiting factor for the synthesis technique. Concatenation synthesis gained prominence in vocal synthesis in the 1990s through work presented by various researchers including Hunt and Black in 1996 [3] and J Olive in 1997 [4]. The technique was quickly seen to be effective for resynthesizing sounds from a corpus of source audio comprised of vocal sonic components such as phonemes, sibilants, or fricatives. Concatenation synthesis remains as one of the predominant means of performing vocal synthesis today.

The early 2000s, referred to as the early years of concatenation synthesis [5], presented an increase in processing power spawning a surge of new interest in the technique exhibiting several different applications with an exploration towards musical ends. Ari Lazier and Perry Cook developed Mosevius [6], a tool for creating "audio mosaics." This application was available as a standalone tool or as a library, where it allowed users to perform concatenative synthesis on a corpus of audio using MIDI or real-time feature extraction on a control signal to inform segment selection. Similarly, Diemo Schwarz' CataRT [7] was a collection of patches for Max/MSP built to perform concatenative synthesis. This system also used manual control or real-time feature extraction of input audio to inform segment selection from a corpus of audio. These two applications were nearly identical in their approach to concatenative synthesis and established a standard paradigm for the process.

More recently, advancements in processing and research have led to the development of more tools utilizing concatenation synthesis in various ways. In 2011, Beller [8] created a physical gestural controller to control segment selection in a concatenative speech synthesis system for performance at IRCAM, which was later built upon by Zbyszyński et al. [9] in 2019, bringing the idea to musical ends along with the introduction of machine learning algorithms. In 2012, S. An et al. created a framework in which plausible accompanying audio is generated for physics based cloth animations by producing a simple target sample based upon simulation information which informs concatenation on a database of higher fidelity cloth samples [10]. In 2016, the audio plug-in Mosaic [11] brought concatenative synthesis towards the audio effects world by layering sounds from an audio corpus on top of incoming audio by means of real-time feature extraction of the input signal guided by user-specified thresholds for particular features. In 2017, MIT produced an application called RhythmCAT [12] that used concatenation synthesis to power a drum programmer and beat maker for electronic music. The application had a focus on refined interface and streamlined functionality, taking advantage of methods like dimension reduction and onset detection to quantize the output and smooth out the user experience. In 2019, C Moore and W Brent explored concatenation synthesis with a new level of interactivity using virtual reality technology to allow users to explore the feature space in three dimensions while forming clustering structures and allowing the user to explore the space with rays and other physical means [13].

While each of these applications explored concatenation synthesis in unique ways, they all rely on a corpus of audio that is pre-analyzed, and they appeal to the design paradigm established in the 2000s by apps such as Mosevius and CataRT. The reason for this is clear, as it is computationally efficient to pre-analyze an audio corpus and then use the resulting meta-data analysis to perform real-time synthesis. However, with current advances in processing

and research in the field, we can expand this paradigm and tap into unexplored territory via generating audio corpora and performing end-to-end concatenation synthesis all in real-time. This may allow the technique to proliferate in the audio effects world, where relatively little work has been done but more potential exists for the technique in audio production and sound design.

1.2. The Delay Line

The phenomenon of audio delay fundamentally influences our auditory experience in the physical world. Acoustical sound waves propagate throughout a space and reflect off surfaces causing the superposition of time-offset waves at the position of a listener's ear or a microphone. Depending on the delay time, this phenomenon yields changes to the perceived audio ranging from an audible echo to intricate alterations in audio timbre resulting from spectral filtering. Notably, the perception of the quality of a space, such as a concert hall, is an amalgamation of all delays resulting from sonic reflections propagating throughout that space [14].

Given this correlation, it comes as no surprise that delay, harnessed through "delay lines," constitutes a crucial aspect of signal processing, whether analog or digital. Delay lines form the foundation for numerous common operations such as filtering, reverb emulation, and physical modeling, as well as various other audio effects like flanging and chorus [15].

With such a fundamental role in audio processing, advances in delay line technology have the potential to permeate multiple areas of signal processing and audio effects. As a result, the delay line represents a significant object of inquiry for modern processing power and algorithms, including real-time music information retrieval and concatenative synthesis.

2. FEATURE-BASED DELAY LINE

This brings us to our proposed model of a Feature-Based Delay Line (FBDL), a novel application of a traditional delay line that utilizes concatenation synthesis where the corpus of audio exists as an ever-changing delay line buffer, analyzed, segmented, and concatenated all in real-time (Figure 1). To accomplish this, segmentation and feature extraction take place as the audio is copied to the delay buffer, where the resulting analysis remains paired with its associated audio as it travels through the buffer. Segments are then selected according to a process we call targeting and concatenated to create the wet signal, which is then mixed with the dry signal. To control concatention of the delay line, the user can set several important parameters, including segment size, feature set, feature weights, targeting method, and targeting parameters. In particular, the critical method of targeting produces a system of selection criteria that expands on the concatenation synthesis norm of simple matching, adding tunable depth to the system and creating the delay-like behavior. The aggregate result is an audio effect that expands on the traditional delay line, exhibiting creative potential for audio production and sound design, as well as potential as a component for signal processing.

As discussed above, the fundamental structure of the FBDL is a delay line, where the content of the delay line buffer constitutes the audio corpus for concatenation synthesis. Our approach is rooted in the functionality of a traditional digital delay line [15], but it expands on the capabilities of the traditional paradigm to create new possibilities. A traditional digital delay line utilizes a circular buffer, where given an audio buffer X of sample size N, the next input sample will write to an indexed buffer position n, such that X(n) = input sample. As audio samples are written, the index n will then be incremented, wrapping around when it reaches the end of the buffer. In a digital delay line, given a delay time t in samples, the delayed signal will read every sample from index n - t (this index will also wrap to stay in the bounds of the buffer), such that the output = X(n - t). Our FBDL is similar to the traditional paradigm, where incoming audio is written to a circular buffer of length N. However, in our case the delay time, now denoted c, is dynamically determined as a result of concatenative synthesis, where the delay buffer position n - c is a function of audio features defined via the targeting process (Figure 2).

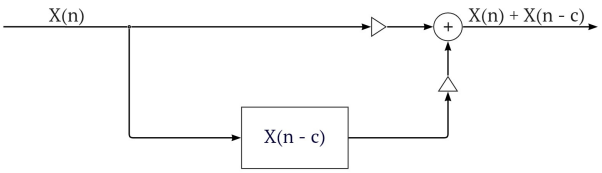


Figure 2: *Feature-Based Delay Line, where delay time c is determined by audio feature analysis (targeting).*

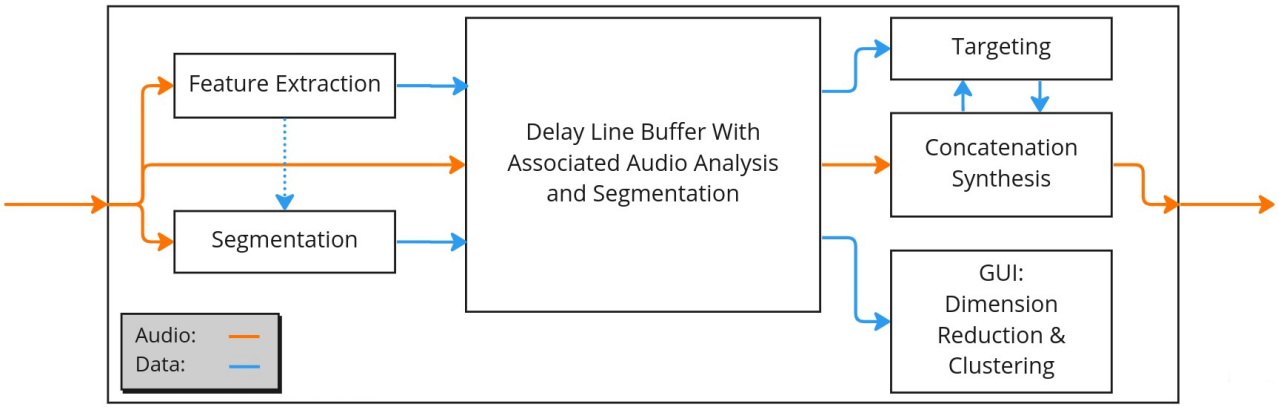


Figure 1: *Data and signal flow of the Feature-Based Delay Line Architecture.*

3. TARGETING

Targeting is our process of traversing a defined feature space and selecting audio segments according to their qualitative position in the space. The method of targeting plays the primary role in determining the delay position within our delay buffer, serving as the bridge between the concatenative synthesis technique and the delay line buffer. All audio that exists inside the delay buffer has been segmented in time and analyzed such that each segment of audio has an associated feature set of descriptors positioning it within a multidimensional feature space. Segments organize in the space such that qualitatively similar audio will be positioned close together and selected accordingly.

For audio selection, a target position in the feature space and a radius about the position are specified. At any given point in time, audio will be selected with a position in feature space contained within the target radius. We define the targeting method as follows:

$$T_c = Targeting(T_{ref}, R) \tag{1}$$

where T_c is the delay time in buffer position output from targeting (equivalent to c in the previous Figure 2), T_{ref} is the reference delay time used to determine the target position in feature space, and R is the target radius about the target position.

Inside the targeting function, given T_{ref} and R, the analysis of each segment of audio in the delay line is compared to the analysis of the audio segment containing X(n - T_{ref}), the "reference segment," where n is the current write position in the delay buffer.

Formally, all audio segments in the buffer can be stated as:

$$S_i = X(n - T_i), \cdots, X(n - T_i + l) \tag{2}$$

where S_i is the segmented audio starting at delay time T_i, with length l (Figure 3).

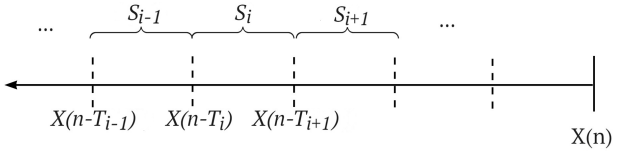


Figure 3: *Delay line segmented for targeting.*

Each audio segment, S_i, is associated with a feature set vector FS_i:

$$FS_i = (F_1, F_2, \cdots, F_m) \tag{3}$$

where F_m is a feature of the audio in segment S_i. The collection of all feature set vectors in the delay line forms our feature space.

The association between each S_i and FS_i is unique except in special cases. For example, given a feature set vector containing only RMS, two segments may share the same value. However, such an occurrence would be scarce and become progressively improbable as the dimensionality of the feature space increases. The only other case where this scenario may occur is with repetitive input signals, in which two segments contain identical audio. Even here, the scenario remains highly unlikely because it necessitates precise alignment with respect to segmentation and FFT framing.

The reference segment of audio, S_{ref}, is defined as the segment which contains the sample $X(n - T_{ref})$:

$$S_{ref} = S_k \mid X(n - T_{ref}) \in S_k \tag{4}$$

where T_{ref} is the target delay time, and S_k is the audio segment containing the sample $X(n - T_{ref})$ (Figure 4).

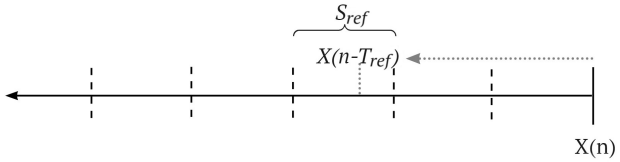


Figure 4: *Reference Segment S_{ref} determined by reference delay time Tref.*

Then, the Euclidean distance between the reference feature vector FS_{ref} and all other vectors in our feature space is computed. If the distance between a given feature space vector FS_i and the reference vector is less than the specified target radius, then the vector is stored in the set of viable segments V.

$$V = \{S_k \forall k \mid d(FS_k, FS_{ref}) \leq R\} \tag{5}$$

A selection candidate S_{sel} is then randomly selected from V, and its reference time, T_{sel}, is output from the targeting function, ultimately setting the delay time T_c equal to T_{sel} (Figure 5).

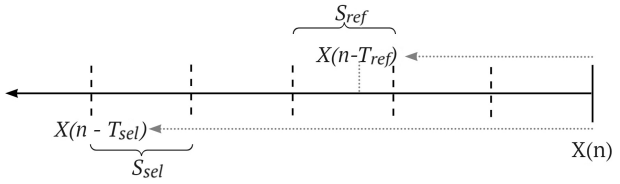


Figure 5: *Segment S_{sel} selected from the set of viable segments. S_{sel} must be sufficiently similar to reference segment S_{ref}. Associated time delay T_{sel} is set as c for the delay tap $X(n - c)$.*

The FBDL can be made to act like a traditional digital delay. For the targeting function detailed above, it is trivial to show how this is achieved. If we let $R = 0$ and assume the case that all FS_i are unique, then the only viable segment for which the euclidean distance between FS_i and FS_{ref} is less than or equal to R is the segment $S_i = S_{ref}$. Thus, the only viable segment is the reference segment, S_{ref} (Figure 6).

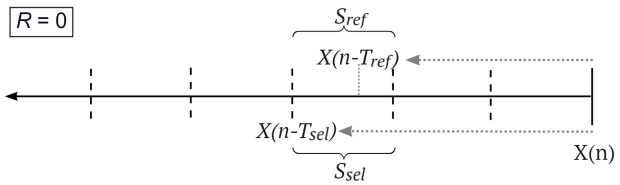


Figure 6: *Reference Segment S_{ref} is the only viable segment as Target Radius R = 0. S_{ref} therefore will be selected.*

However, there is a small discrepancy here: because T_{ref} might fall anywhere inside S_{ref}, there is some potential error from the exact reference delay time given depending on when the targeting method is queried. While this is not typically noticeable depending on the segment size used, it can be remedied with slight shifting of the selected segment. In this case, the selected segment will be shifted to align with the reference delay time if it is the reference segment containing T_{ref} (Figure 7).

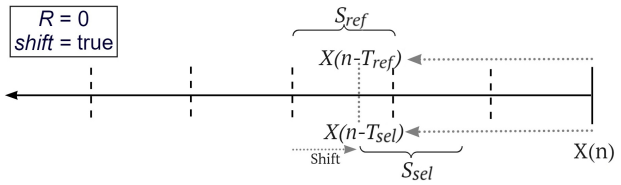


Figure 7: *Reference Segment S_{ref} is the only viable segment as Target Radius R = 0. With shifting enabled S_{sel} = S_{ref} is now shifted to align with T_{ref}.*

Now the output from the targeting method will always be T_{ref}. Therefore, in this scenario, the FBDL and the traditional digital delay line are identical, as $X(n - T_c) = X(n - T_{ref})$.

As the value of R increases, more audio segments become viable candidates for selection, and thus more sonic variety is introduced into the delayed signal.

4. TARGETING EXPANSIONS - PARAMETERS AND METHODS

The targeting function can be modified or parametrically expanded to impact the behavior of the delay. This can be accomplished via adding parameters to a given targeting method or the targeting method itself can be modified to process input parameters in various ways. Our targeting function's expanded signature can be generalized as:

$$T_c = Targeting(T_{ref}, R, \cdots) \tag{6}$$

For our implementation we include the addition of feature weights and target smoothing, defining our final targeting function as follows:

$$T_c = Targeting(T_{ref}, R, FW, S) \tag{7}$$

where FW is a vector of feature weights, and S is a smoothing factor.

4.1. Feature Customization and Weighting

The feature set can be customized to alter the selection process and sound quality of the delay effect. Customization can result from a combination of features being added or removed from a given feature vector, or specific features in the vector may be replaced with others that are more desirable. The introduction of a feature weight vector allows the user to control the relative strength of each feature in the vector during the selection process, thereby accentuating specific audio characteristics in determining the viability of a segment. Computationally, this is a simple enhancement to the targeting function that can allow greater real-time control over the selection process. To accomplish this, compute the Hadamard Product [16], where given a feature vector and vector of feature weights both of the same length n:

$$FV_{wgt} = FV \circ FW = (F_iW_i, \cdots, F_nW_n) \tag{8}$$

where FV_{wgt} is the weighted feature vector, FV is the original feature vector, and FW is the vector of feature weights. The entry-wise product is computed for all vectors in the feature space before computing Euclidean distances and determining segment viability.

4.2. Smoothing

The smoothing parameter S changes the way the target position traverses the feature space by setting the feature set of the reference segment equal to the average of the feature sets of the prior N segments trailing the reference segment in the delay line. Thus, by applying a smoothing factor, the feature set of the reference segment is defined as:

$$FS_{smth} = \frac{1}{N}\sum_{i=0}^{N-1} FS_{ref-i}. \tag{9}$$

This effectively makes the size of the reference segment larger, as it incorporates the features of more audio into its average. As a result, the target may remain more centralized about the feature space and mitigate effects of outliers.

4.3. Targeting Methods

Targeting methods define distinct paradigms for traversing and organizing the feature space. Targeting methods are composed of unique targeting functions with varying parameter sets existing as arguments that fundamentally impact targeting behavior. Each targeting method may have a collection of parameters that apply to it that may or may not also apply to other targeting methods.

The primary targeting method utilized is a best fit approach, based on the audio input into the delay line, as well as a collection of other potential parameters such as target delay time, audio feature sets, feature weights, and smoothing, as detailed above. This method is parametrically automated, and as shown lends itself to unique potential for the FBDL.

Alternatively to our automated approach, manual targeting was also implemented. For this method the user manually determines the target position in feature space via use of a graphical user interface or other direct interactive methods. Our implementation focuses on the former, allowing a user to traverse and explore the feature space more freely and develop a deeper understanding of the feature space bounds. For example, it can be effectively used when writing to the delay line is paused, "freezing" the state of the delay line into a temporarily static corpus. With a static corpus and feature space, the listener can take time to consider how audio segments are associated with various qualities in the space. This can greatly inform the user on setting and refining parameters that impact the targeting process and segment selection.

Functionally, the behavior of the FBDL is critically determined by the targeting parameters and method. As shown above, with restrictive settings it will act exactly like a traditional delay line. However with modifications, the FBDL can produce a broad array of sonic behavior ranging from light variation to entirely new textures that are generated and layered into the original audio. In this way, our FBDL encompasses the full scope that a traditional delay line affords while introducing a broad set of new possibilities.

5. FEATURE EXTRACTION AND CONCATENATION CONSIDERATIONS

5.1. Feature Extraction

The characteristics of the feature space impact the capabilities of the FBDL architecture. Every feature is a descriptor that is used to organize the segments of audio in the delay line. Different

features afford different ways to organize the segments of audio, and this organization affects segment selection during the targeting method. Importantly, each feature is a characteristic that can be compared to the target position in order to determine a segment's viability. For example, the presence of a "loudness" feature or a "noiseness" feature in the feature set allows for volume or noise to affect a segment's viability respectively.

5.1.1. Feature Impact

A given feature is only as useful as the amount of variation present among the population of segments in the delay line. In other words, "noisiness" provides no useful differentiation in a population of segments that all have the same amount of noise. The same is true for pitch, loudness, or any other descriptor. With this notion, variations in acoustic properties among different sound types may necessitate the selection of distinct features for optimal signal processing. For example, with percussive sound sources, measurements of energy and noise and change in spectra are likely to be more useful than specific pitch related features such as fundamental frequency, as opposed to tonal sources which are likely to benefit from the opposite. Part of exploring the feature space is discovering which set of features provide the most utility for a given sound source.

5.1.2. Feature Set

Access to a broad feature set is important in order to maximize the customization possibilities during audio segment selection. To this end many features are available, including some that measure similar characteristics through different means. The full feature set is as follows: MFCC, Spectral Centroid, Spectral Bandwidth, Spectral Rolloff, Spectral Flatness, Spectral Flux, Spectral Contrast, Short Time Energy, Short Time Variance, RMS, and Fundamental Frequency Estimation. Feature sets can be streamlined by selecting a more manageable set that maximizes contrast and minimizes redundancy.

5.1.3. Dimension Reduction and Clustering

Dimension Reduction and Clustering algorithms are used in the implementation's GUI as means of simplifying the complex n-dimensional feature space into a more comprehensive, understandable, 2-dimensional representation. Dimension reduction is computed using Principal Component Analysis (PCA) [17] and clustering is computed using DBScan [18]. While versions of these algorithms have often been used in concatenative synthesis applications and other audio database visualizations to reduce dimensionality and present clustering structures, there are unique considerations involved with this architecture due to the ever-changing dynamic database (delay line buffer), compared to past applications with static databases. Namely, this includes stabilization of the analysis in the presence of rapid change, especially for dimension reduction, where small changes may cause the reduction to flip orientation or change basis dramatically. Real-time PCA is still an open problem in the data science community [19].

5.2. Concatenation

Operations required for concatenation synthesis can also materially impact the sonic quality of the delay line. Most importantly, the treatment and combination of delay line segments requires the greatest consideration, where particular focus may be given to segment definition, windowing, number of segments, layering of segments, and segment effects processing.

5.2.1. Segment Definition

Segment definition plays a crucial role in the concatenation synthesis of the delay line audio. Segment delineation can be determined automatically via onset detection, or from designation of regular units of time such as length in samples or beats per minute. The segment size has a substantial effect on the sound quality of the concatenation. Segment size must be set at a minimum such that it can constitute a frame for spectral analysis. For our implementation, the default size is 2048 samples with 50% overlap. Segment size may be tuned for different use cases, where short segments (grains) sound more textural when stitched together, maintaining timbre but not temporal events, and longer segments (syllables) sound more like musical events strung together in sequence.

5.2.2. Windowing and Overlap

Windowing is applied to each segment in order to smooth the transition between disparate audio segments. For our implementation, a Tukey windowing function [20] is used, which has a sinusoidal onset and offset, and a flat band in the center. Control of the center bandwidth affects the quality of the concatenated audio stream, where a larger bandwidth yields greater individual presence of each segment and a smaller bandwidth results in less individual presence as onset / offset periods of segments meld together. Segments read from the delay line may be overlapped during the onset and offset periods of their windowing function in order to more seamlessly stitch them together. This is particularly important with small segment sizes in order to mitigate the introduction of unwanted spectral artifacts. For the special case of configuring the delay line to perform as a traditional delay line, the center bandwidth may be set to the size of the entire window with zero overlap.

5.2.3. Number of Segments

The number of segments determines how many segments are being read from the delay line at any given time. Each additional segment is another tap into the delay line. This parameter greatly increases the textural capabilities of the concatenation process. Additionally, multiple segments may layer to produce chorus-like effects, as similar audio segments are combined together with slight pitch and time offsets. When combining large numbers of segments, they are typically offset from each other for the following reasons. Firstly, this naturally offsets the onset and offset periods of the segment windows. Secondly, this generally allows for more variation in the segments selected, and if multiple copies of the same segment are selected, their constructive amplification is mitigated. Finally, this is computationally advantageous, as it spreads out the targeting queries between more calls to the audio processor.

5.2.4. Adding Effects

A multitude of effects can also be applied to the concatenated segments read from the delay line. These include speed and pitch shifting, reverse playback, panning, waveshaping, and more. Effects can be applied universally across all segments or uniquely to each segment read from the delay line. Variation in how the effects

are applied allows for more diversity of texture. For example, this is important for pitch and panning, as it allows for chorus-like pan and pitch width effects, where pan and pitch offsets are applied to segments spread evenly around a center value.

5.2.5. Parameter Mapping

All the effects and parameters can be mapped to the read segments in different ways. They can be manually designated by the user or parametrically automated to produce interesting results. One of the unique affordances of the FBDL architecture is the access to an array of analysis of all the contained audio, which can be used to achieve powerful real time automatic control of parameters. Specifically, effects can be mapped to any of the feature axes of the current target position or the feature set of each individual segment read with custom graphs. This allows for numerous possibilities, including automatic equalization of segment volume, pitch normalization of segments, silencing noisy sounds, etc., as well as many custom behaviors for other specific goals. Generally, the characteristics of the sound can uniquely determine audio effects processing for each segment, allowing for endless customization of the playback of audio from the delay line.

6. USE CASES

The Feature-Based Delay Line architecture promotes many different use cases which were explored in our implementation. As an expansion of the traditional delay line, it fulfills the same functionality. However, with increased control over the delay behavior and extra affordances of the architecture, it expands the boundaries of traditional use cases into new territory for sound design. Furthermore, although currently a high level tool, we believe that future iterations of the FBDL approach may have potential use cases as a lower level signal processing component.

6.1. General Purpose Delay

Typical delay use cases can be enhanced in many ways. Subtle new affordances can be introduced with conservative FBDL settings, such as small target radius, no segment layering, and minimal segment effects processing. For example, targeting parameters can subtly enhance traditional use cases by affecting segment selection, such as expanded target radius introducing variation into the delayed signal. Additionally, unwanted portions can be filtered from the input signal using parameter mapping, via mapping characteristics of unwanted audio such as noisiness to volume. Alternatively, desirable sections of the input signal can be accentuated by linking characteristics of such segments to parameters such as pan, pitch, feedback amount, or target radius, increasing the possibilities of the effect.

6.2. Textural Audio

The introduction of concatenation synthesis into the FBDL promotes a unique use case for textural audio and layering. By increasing the intensity of the FBDL settings, including expanded targeting radius, increased number of concurrently read segments, and more liberal segment effects processing, the delayed signal can be pushed more towards the textural "audio mosaic" realm. With this, the well established traits of prior concatenative synthesis applications are exhibited, while still maintaining the link to the rhythm and quality of the input audio. The texture created either can stand alone, or be layered on top of the dry signal. The unique strength of the dynamic corpus of the FBDL is exhibited here, as the texture melds with the the original source audio in real time, adding additional layers of timbre.

6.3. Resonance and Comb Filtering

The FBDL presents an interesting use case in comb filtering and resonant delay modeling, due to the expression of the architecture with very short delay times. A selective comb filter effect can be created using a small but non-zero target radius, where the filtering will be predominantly active, as the reference segment will be selected at sub-25ms delay time, but sometimes inactive, when a non-reference segment is selected further back in the delay line. Similarly, with high feedback amounts, novel resonant effects can be achieved. The frequency at which the signal is delayed introduces resonant spectral content, which compounds as it repeatedly feeds back into the delay line. Targeting parameters provide a selective and natural way to periodically break out these feedback cycles, allowing for interesting but manageable resonant delay effects. These approaches can be adjusted via targeting parameters and interacts with concatenation parameters and parameter mappings in interesting ways. This also serves as a high-level example of how the FBDL might be used to enhance existing signal processing operations that make use of delay lines.

7. EVALUATION

Evaluation in concatenative synthesis systems has often historically been lacking [12, 21], due to the creative and / or subjective goals of the creator, often in the role of composer. Nonetheless, we maintain that the evaluation of the Feature-Based Delay Line is crucial for the advancement of the delay line as a signal processing component and as an audio effects processor. To this end, a prototype of our FBDL architecture was realized as a JUCE C++ plugin. Specifically our prototype should encompass the possibilities of the traditional delay line, while also introducing new dimensions in the space, expanding the potential range of behaviors. Utilizing this implementation, we conducted experiments to evaluate performance both as a traditional delay line and with new capabilities, such as targeting radius and parameter mapping. These experiments were conducted in isolated circumstances to allow for targeted assessment.

7.1. Delay Line Variation Experiment

As previously stated, the Feature-Based Delay Line is designed to incorporate traditional digital delay functionalities while also introducing new possibilities. The present study aims to evaluate the FBDL's performance as a basic delay line and to analyze the changes in output as the target radius is introduced. The experiment pursues two main objectives. The first objective is to compare the delayed signal from the FBDL with that of a standard digital delay line using neutral settings and consecutive increases in target radius. The second objective is to assess the impact of the introduction of the target radius on the output signal and its feature analysis data.

To achieve these objectives, measurements were taken by computing the differences in the waveform and feature analysis

between the control signal and the FBDL signal with different target radii. The focus was on two specific results: (1) whether the output signal from the FBDL architecture is identical to that of the control delay under neutral settings, and (2) how much the introduction of target radius affects the output signal and its feature analysis data.

Figure 8 presents the results of the experiment. It shows that with a target radius of zero and default parameters, there is no meaningful difference between the delay signal from the FBDL and the control signal. However, as the target radius increases, the differences between the waveform and feature analysis of the two signals also increase. These differences are strongly correlated with the target radius. The plot indicates a contour due to a spike in variation introduced as the target radius encompasses clusters of segments. This is followed by a slight plateau until the target radius expands sufficiently to include different clusters, eventually resulting in complete randomness at a target radius of 1.0. The shape of this contour may vary depending on the distribution of points throughout the feature space, but it will always be positively correlated with the target radius (with some expression of randomness due to nondeterministic segment selection).

These findings demonstrate the FBDL's ability to encompass the behaviors of a typical delay line and evaluate one of the new axes (targeting, specifically target radius) introduced into the possibility space of the architecture, under circumstances that isolate that particular axis.

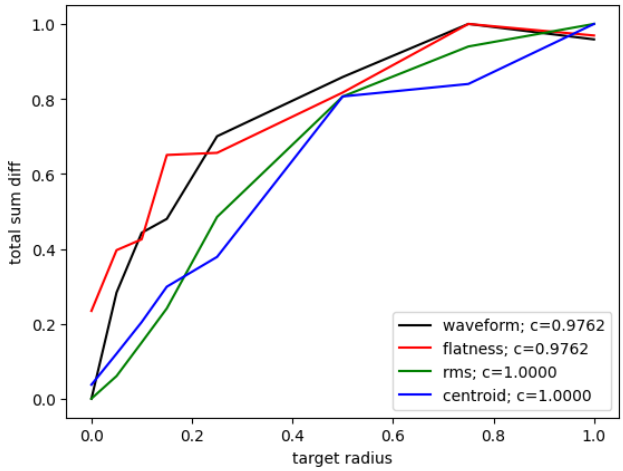


Figure 8: *Waveform & Feature Differences vs. Target Radius.*

7.2. De-Essing Experimental

There are numerous ways to take advantage of the FBDL's utilization of parameter mapping to access analysis of the delay buffer. For our second experiment, de-essing presented itself as an interesting ability of the architecture and an apt candidate for evaluation. De-essing is a well-defined, isolated practice with a clear connection to feature analysis. For example, sibilance in a vocal sample corresponds to increased measures of noise. Specifically, this experiment tests the de-essing capabilities of parameter mapping by using a mapping of spectral flatness to volume and compares the results to a commercial de-essing plugin (set to max strength).

The de-essed signals from both the FBDL and the commercial de-esser are compared to each other and to the control signal through waveform and feature analysis differences. The FBDL's signal is recorded with a delay time equal to the minimum analysis frame size, then time shifted to be in sync with the other waveforms for comparison. The amount of noise in the signals is computed by summing the multiplication of each sample by its spectral flatness value. The remaining noise ratio is the amount of noise in the de-essed signal divided by the amount of noise in the control signal. The removed noise ratio is the amount of noise in the difference between the de-essed signal and the control divided by the amount of noise in the control signal.

As shown in Figure 9, the de-essing created by the parameter binding is effective at filtering out noise from the control signal, removing 45.62%, compared to 23.53% removed by the commercial de-esser. Note that the sum of the remaining and removed noise sums roughly to the amount of noise in the unaffected waveform. The de-essed waveforms are plotted in Figures 10 and 11, with removed signal highlighted in red.

These results show FBDL's success as a de-esser within this scope of evaluation, as well as serving nicely to display the efficacy of parameter mapping as a general technique, which could be applied in other ways towards unique ends.

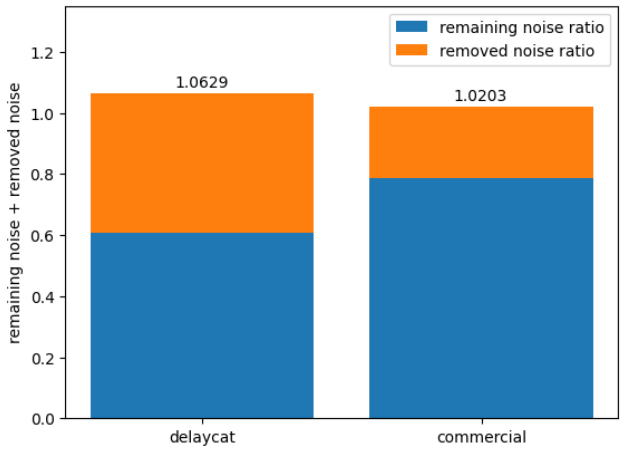


Figure 9: *Removed & Remaining Noise*

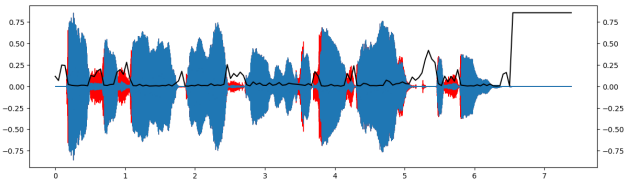


Figure 10: *FBDL De-Essing Effect*

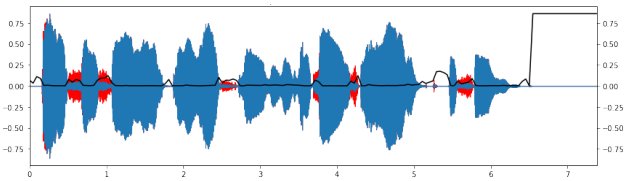


Figure 11: *Commercial De-Essing Effect*

8. CONCLUSION

The union of concatenation synthesis and dynamic delay buffer into an intelligently guided Feature-Based Delay Line (FBDL) offers unique possibilities both as an expansion of the traditional delay line and as an application of concatenation synthesis. With this architecture, we aim to inspire further exploration of music information retrieval and concatenative synthesis in the area of audio effects processing, and innovate upon the prior applications of concatenative synthesis by introducing a real-time dynamic database and expandable targeting method to traverse the feature space based on delay time. Furthermore, expansion on the delay line as a fundamental component may result in progress throughout related areas of signal processing.

Future work on the FBDL architecture will address unique considerations of this approach, such as segment alignment and feature analysis with an arbitrary delay time, along with real-time dimension reduction and clustering integration into the targeting and parameter mapping parts of the architecture as additions or substitutions in the feature set. Additionally, continued development in FFT optimization, especially through GPU accelerated implementations [22] and / or dedicated FFT processing hardware [23] will limit the amount of error in the architecture and expand its potential. Finally, we look to conduct a detailed performance evaluation and seek qualitative user feedback from sound designers to identify areas of improvement in our design and implementation.

We are eager to share our approach with the broader audio and music community. A video demonstration, as well as builds of the plugin, experimental notebooks, and performance evaluation notes are attainable via the project repository located at `https://github.com/NiccoloAbate/DelayCat`.

9. REFERENCES

[1] Diemo Schwarz, "Current research in concatenative sound synthesis," in *ICMC*, 2005.

[2] Iannis Xenakis and Mrs John Challifour, *Formalized Music: Thought and Mathematics in Composition*, Bloomington: Indiana University Press, 1971.

[3] Andrew J. Hunt and Alan W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, pp. 373–376 vol. 1, 1996.

[4] Joseph P Olive, "Section introduction. concatenative synthesis," in *Progress in Speech Synthesis*, pp. 261–262. Springer, 1997.

[5] Diemo Schwarz, "Concatenative sound synthesis: The early years," *Journal of New Music Research*, vol. 35, pp. 3–22, 03 2006.

[6] Ari Lazier and Perry Cook, "Mosievius: Feature driven interactive audio mosaicing," in *Digital Audio Effects (DAFx)*. Citeseer, 2003.

[7] Diemo Schwarz, Grégory Beller, Bruno Verbrugghe, and Sam Britton, "Real-time corpus-based concatenative synthesis with catart," in *Proceedings of the 9th International Conference on Digital Audio Effects, DAFx 2006*, 2006.

[8] Grégory Beller, "Gestural control of real time concatenative synthesis," *ICPhS*, 09 2011.

[9] Michael Zbyszyński, Balandino Di Donato, Federico Ghelli Visi, and Atau Tanaka, "Gesture-timbre space: Multidimensional feature mapping using machine learning and concatenative synthesis," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2019, pp. 600–622.

[10] Steven An, Doug James, and Steve Marschner, "Motion-driven concatenative synthesis of cloth sounds," *ACM Transactions on Graphics - TOG*, vol. 31, 07 2012.

[11] "Echobit," `https://echobit.myshopify.com`, Accessed: 2022-05-01.

[12] Cárthach Ó Nuanáin, Perfecto Herrera, and Sergi Jordà, "Rhythmic concatenative synthesis for electronic music: Techniques, implementation, and evaluation," *Computer Music Journal*, vol. 41, pp. 21–37, 06 2017.

[13] Carl Moore and William Brent, "Interactive real-time concatenative synthesis in virtual reality," in *Proc. ICAD*, 2019, pp. 317–320.

[14] Eric Tarr, *Algorithmic Reverb Effects*, pp. 349–379, Routledge, 06 2018.

[15] Udo Zölzer, *DAFX: Digital Audio Effects: Second Edition*, John Wiley & Sons, 03 2011.

[16] Roger A. Horn and Charles R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[17] Michael Greenacre, Patrick Groenen, Trevor Hastie, Alfonso Iodice D'Enza, Angelos Markos, and Elena Tuzhilina, "Principal component analysis," *Nature Reviews Methods Primers*, vol. 2, pp. 100, 12 2022.

[18] Martin Ester, Peer Kröger, Joerg Sander, and Xiaowei Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 01 1996, vol. 96, pp. 226–231.

[19] Ranak Roy Chowdhury, Muhammad Abdullah Adnan, and Rajesh Gupta, "Real-time principal component analysis," *ACM/IMS Transactions on Data Science*, vol. 1, pp. 1–36, 06 2020.

[20] "Tukey window," `https://www.mathworks.com/help/signal/ref/tukeywin.html`, Accessed: 2022-05-16.

[21] Cárthach Ó Nuanáin, Perfecto Herrera, and Sergi Jordà, "An Evaluation Framework and Case Study for Rhythmic Concatenative Synthesis.," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, New York City, United States, Aug. 2016, pp. 67–72, ISMIR.

[22] Dmitrii Tolmachev, "Vkfft - a performant, cross-platform and open-source gpu fft library," *IEEE Access*, vol. PP, pp. 1–1, 01 2023.

[23] Joseph Ja'Ja and Robert Owens, "An architecture for a vlsi fft processor *," *Integration, the VLSI Journal*, vol. 1, pp. 305–316, 12 1983.

DYNAMIC PITCH WARPING FOR EXPRESSIVE VOCAL RETUNING

Daniel Hernan Molina Villota
Institut Jean Le Rond d'Alembert
Equipe Lutheries-Acoustique-Musique
Sorbonne Université
Centre National de la Recherche Scientifique
Paris, France
daniel.molina_villota
@sorbonne-universite.fr

Christophe d'Alessandro
Institut Jean Le Rond d'Alembert
Equipe Lutheries-Acoustique-Musique
Sorbonne Université
Centre National de la Recherche Scientifique
Paris, France
christophe.dalessandro
@sorbonne-universite.fr

Olivier Perrotin
Université Grenoble Alpes
CNRS, Grenoble INP
GIPSA-lab
Grenoble, France
olivier.perrotin
@grenoble-inp.fr

ABSTRACT

This work introduces the use of the Dynamic Pitch Warping (DPW) method for automatic pitch correction of singing voice audio signals. DPW is designed to dynamically tune any pitch trajectory to a predefined scale while preserving its expressive ornamentation. DPW has three degrees of freedom to modify the fundamental frequency (f_0) signal: detection interval, critical time, and transition time. Together, these parameters allow us to define a pitch velocity condition that triggers an adaptive correction of the pitch trajectory (pitch warping). We compared our approach to Antares Autotune (the most commonly used software brand, abbreviated as ATA in this article). The pitch correction in ATA has two degrees of freedom: a triggering threshold (flextune) and the transition time (retune speed). The pitch trajectories that we compare were extracted from autotuned-in-ATA audio signals, and the DPW algorithm implemented over the f_0 of the input audio tracks. We studied specifically pitch correction for three typical situations of f_0 curves: staircase, vibrato, free-path. We measured the proximity of the corrected pitch trajectories to the original ones for each case obtaining that the DPW pitch correction method is better to preserve vibrato while keeping the f_0 free path. In contrast, ATA is more effective in generating staircase curves, but fails for not-small vibratos and free-path curves. We have also implemented an off-line automatic picth tuner using DPW.

1. INTRODUCTION

Pitch correction (or automatic pitch tuning) is nowadays one of the most commonly used digital audio effects for vocal music. Initially known as the "Cher" effect, the audible distortion produced by sharp pitch transition in retuned singing became appreciated on its own in popular electronic music. The sharp transition is a case of use where all minor expressive singing variations are flattened. Noticeable gliding appears often in the transitions between notes. The success of Autotune in the music industry has sparked much discussion and debate. Some argue that it is a tool that helps artists achieve a perfect pitch singing, while others criticise its use as it can lead to a loss of natural expression and emotion in the music. Despite this, Autotune has become a staple in modern music production and is used in various genres such as pop, hip-hop, and electronic music [1]. Although it is a common practice to use DAFx effects which involve perceptual features such as [2] melody (pitch), source (timbre, [3]), or space [4],pitch correction is one of the most commonly used. I became a stylistic signature for many popular music genre.

Antares Autotune (ATA)[1] is a digital audio effect developed by H. Hildebrand in 1997 [5] and its enduring popularity has spanned over 25 years. ATA uses an autocorrelation method that was initially developed for seismic imaging, with the help of short-time Fourier transform. Although the initial purpose of ATA was not to enrich the voice with a new vocoder-like audio effect but to correct out-of-tune melodies, the unique electronic texture produced has been embraced in popular music and has even become a hallmark of specific musical styles, often employed systematically. ATA offers two use cases: one the one hand pitch correction is used for better rendering of out of tune singing and on the other hand the distortion effect occurring extreme correction situations is appreciated on its own. The need for melodic correction also appeared in digital music instruments (DMI) [6, 7, 8, 9]. These DMIs use interfaces with particular features that involve learnability, explorability, and controllability [10]. A new pitch tuning correction, Dynamic Pitch Warping (DPW) [11], has been developed for performative vocal synthesis in Cantor Digitalis [8] where the fundamental frequency (pitch) is controlled in real-time with the help of a stylus on a graphic tablet. Pitch correction helps for singing accurate notes. However, it is very important to preserve small expressive ornaments like vibrato [12] without flattening the notes to preserve naturalness.

The purpose of this paper is to study the DPW pitch correction method. This method was designed to preserve expressive variations like vibrato while adjusting the main shape of the f_0 curve to a predefined scale. We identify three cases of particular interest: abrupt pitch transitions (staircase notes), notes with vibrato and free path curves that should not be corrected. The results of this paper allow us to open perspectives for developing dynamic and singer-controlled vocal digital audio effects that are able to preserve expressive ornaments in real-time. Section 2 presents a review of the pitch correction method studied (ATA and DPW). Section 3 compares DPW and ATA on typical pitch patterns. Section 4 presents the off-line implementation of DPW for audio signals.

2. PITCH CORRECTION SYSTEMS

An audio pitch correction system contains three parts: a pitch detection algorithm (PDA), a pitch correction algorithm, and finally

[1] https://www.antarestech.com/ last checked: 6 April 2023

a pitch warping modification (vocoder). The present paper aims to apply DPW as a pitch correction algorithm for vocal speech intonation. DPW offers three control parameters when other correction methods have one or two parameters. DPW uses an adaptive function, the term "adaptive" is related to the adaptive digital audio effects (aDAFx) that are recent solutions designed to respond to changes in the input signal and adjust specific audio parameters accordingly to it, thanks to specific denominated adaptive functions. These kind of effects are more dynamic and responsive that the traditional DAFx, some examples of aDAFx being the compressor, the expander and the limiter (auto-adaptive on loudness).

Along this line, several DMIs have introduced the use of pitch correction methods to improve the expressivity of musical user interfaces. That is the case for devices such as the Continuum Fingerboard [6, 7][2], the Seaboard[13][3], Garageband[4], TouchKeys[14], and Cantor Digitalis [8][5]. The latter is particularly interesting since it uses a Dynamic Pitch Warping method to correct the continuous position of the pitch controller relative to a pitch scale. The corresponding adaptive warping function proposed by Perrotin and d'Alessandro [11] attracts real pitch values towards integer values, using a MIDI scale. The integer values are tuned notes. DPW is based on a pitch velocity condition expressed as the pitch stability within a pitch interval during a critical time threshold before triggering the automatic correction. We will review the warping methods applied in ATA and DPW in the following two subsections.

2.1. Autotune Antares

Autotune was developed by H. Hildebrand using techniques originally developed for mapping the Earth's subsurface and is considered a time-domain vocoder that modifies the signal both on the frequency and time domain using a short-time Fourier transform with a window function to frame the inner transform. Autotune is a full pitch correction system including the three steps described above: pitch detection, pitch correction and pitch modification. We present in this section the pitch correction method. For this purpose, the sung notes are shifted to the closest note in a predefined scale, and the transition is carried out over a duration equal to a transition time (named "retune speed" on ATA). Autotune also includes the flextune parameter, which acts as a threshold for the correction and represents the size of the neighborhood of a note in which a pitch correction can be triggered.

Due to lack of detail in the patent [5], the ATA algorithm can only be reproduced for an extreme correction case, meaning a value 0 on the Decay parameter in the patent of ATA (internal parameter of the code, and related to the retune speed parameter). This case corresponds to force the input trajectory to match integer MIDI values, i.e., the target notes. For the non-zero Decay cases we cannot reproduce the algorithm as the patent doesn't describe exactly the configuration of the smoothing step. To treat cases with non-zero transition time we will apply the ATA VST on audio signals and then extract the retuned f_0 to study correction actually carried on.

[2] `https://www.hakenaudio.com/continuum-fingerboard` last checked: 25 may 2023

[3] `https://www.roli.com` last checked: 25 may 2023

[4] `https://www.apple.com/mac/garageband/` last checked: 25 may 2023

[5] `http://www.lam.jussieu.fr/cantordigitalis/` last checked: 25 may 2023

2.2. Dynamic Pitch Warping

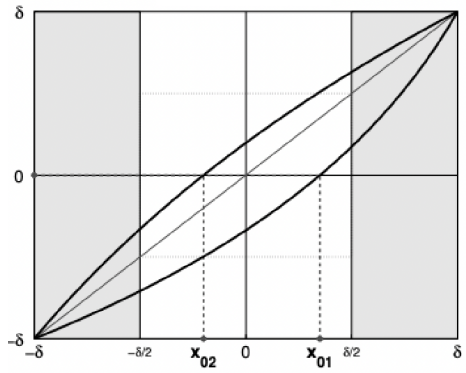


Figure 1: *The arc of curvature for the dynamic pitch correction method, took from [11]*

DPW is a real-time pitch correction method developed by Perrotin and d'Alessandro for Cantor Digitalis. Although it was originally designed to correct a driven by stylus pitch on a graphic tablet, we aim to use DPW for vocal correction. DPW relies on pitch velocity (speed) to trigger an adaptive correction that modifies the input f_0 curve gradually, enabling the output f_0 to converge to the nearest semitone on the MIDI scale. When pitch velocity falls below a threshold, DPW smoothly shifts subsequent f_0 values to converge to a tuned semitone, while preserving some expressive motion of the original f_0 value. The adaptive function remains static when the pitch velocity condition is not met, allowing intended notes to be corrected while retaining expressiveness and preserving all dynamics for non-corrected notes. To review the method, we first analyze the isolated adaptive function, as seen in Figure 1 that maps the input f_0 (x axis) to the output f_0 (y axis). On both axes, zero represents the closest target (ideal) pitch, and $-\delta$ and $+\delta$ correspond to the previous and next notes on the discrete target pitch scale, respectively. While it works on any arbitrary scale, $\delta = 1$ when working with semitones. For input pitch x_{01}, the closest target note is zero. Therefore, at the time the correction is triggered, the corresponding adaptive function that is initially diagonal will smoothly shift towards the lowest arc-shaped curve, to eventually map the input f_0 to the pitch target (zero) as output f_0. The adaptive function then becomes static until it is newly triggered. To avoid introducing a constant shift on the full pitch range, the adaptive function is arc-shaped so that if the input moves from the x_{01} value to the neighbour notes on the pitch scale ($-\delta$ or $+\delta$), the output f_0 will continuously reach $-\delta$ or $+\delta$. If those boundaries are reached, the adaptive function goes back to a linear mapping between input and output, until it is triggered again for a new input.

The adaptive function is derived from the analytic definition of an arc. To ease formulation, the inverse function is first defined:

$$x(y) = Ae^{\gamma(y+B)} + C \tag{1}$$

where the parameters A, B, C and γ can be calculated from the boundary conditions, i.e., the arc must satisfy $x(\pm\delta) = \pm\delta$. If we use this condition, we can write A and C in terms of γ, δ, and B

as follows:

$$C = -\delta\left(1 + \frac{2}{e^{2\gamma\delta} - 1}\right), A = 2\delta\frac{e^{\gamma(\delta - B)}}{e^{2\gamma\delta} - 1} \tag{2}$$

Replacing these values in the original equation 1, we find that the dependency on B disappears. Furthermore, the function is not defined for $\gamma = 0$, but it corresponds to an absence of correction, i.e., the mapping is linear. So the function of the arc curvature can be written as:

$$x(y) = \begin{cases} \delta\left[2\frac{e^{\gamma(\delta+y)}-1}{e^{2\gamma\delta}-1} - 1\right] & if \quad \gamma \neq 0 \\ y & if \quad \gamma = 0 \end{cases} \tag{3}$$

The adaptive warping function is defined as the inverse of 3:

$$y(x) = \begin{cases} 1/\gamma\left[\log\left[(e^{2\gamma\delta} - 1)(\frac{x}{\delta} + 1)\frac{1}{2} + 1\right]\right] - \delta & if \quad \gamma \neq 0 \\ x & if \quad \gamma = 0 \end{cases} \tag{4}$$

Where γ is the factor of correction, y is the output pitch after the correction, and x is the input pitch. When the correction is triggered (at that moment $x = x_o$) , the value of $\gamma = \gamma_0$ can be calculated from the input value x_0 to ensure that $y(x_0) = 0$ following the equation:

$$\gamma_0 = \frac{1}{\delta}\log\left(\frac{\delta - x_0}{\delta + x_0}\right) \tag{5}$$

The DPW has two stages that can be seen on Figure 2. One is the triggering part and the other is the warping stage. For the correction to be triggered, the pitch trajectory has to be stable enough, i.e., it has to stay within an interval of detection (ID) during a critical time (T_c) [11]. If these conditions are met, we can calculate the curvature γ_0 given the input pitch at triggering time (x_0 in the definition, f_0 for us). To ensure a smooth transition, γ is linearly interpolated from 0 (linear mapping) to γ_0. This transition spans a time interval denominated transition time (T_t). When the transition is completed, the input pitch has converged to the closest integer notes on the midi scale. This transition is carried out similarly to the static case of ATA, not over the frequency but over the γ value, then f_0 (input) is warped with the adaptive function.

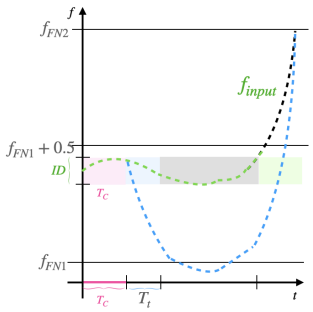


Figure 2: *Illustration of the dynamics of DPW. The input f_0 (green curve) is stable in a detection interval ID during the critical time T_c (pink region). The correction is triggered during the transition time T_t (blue region). The input f_0 can vary continuously during the transition time, until it reaches the next semitone on the pitch scale (integer, black).*

3. CASE STUDIES OF PITCH CORRECTION

In this section, we compare both ATA and DPW methods. Firstly, we want to show the difference between the methods through a simple case. We take as example a constant flat note (C♯) with a pitch shift of 0.15 semitone (ST), and we use both methods to correct it. In Figure 3, we see a DPW correction (blue) triggered with the following parameters: $ID = 0.1$ ST, $T_c = 0.5$ s, and $T_t = 0.5$ s. The ATA correction (red) has a retune speed equal to T_t. We have chosen a non-zero value for T_c to show the inclusion of the new parameter. The critical time is the main difference between both methods. While it introduces a triggering delay in DPW, we find similar results for both corrections once after that trigger.

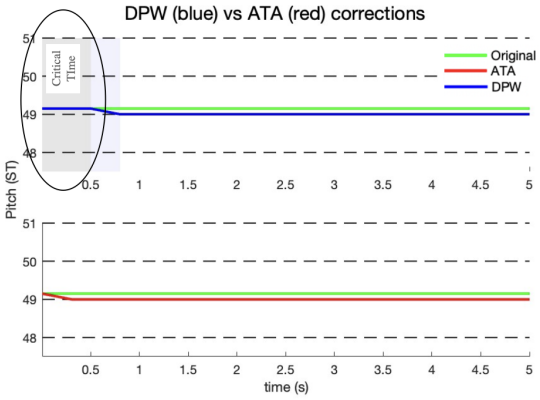


Figure 3: *DPW correction (blue curve) and ATA correction (red curve) of a constant input pitch (green curve).*

3.1. Extreme correction with zero transition time parameter

We denominate extreme correction to a full discretization of the input pitch trajectory. To check the extreme correction, we chose two typical examples: the first one is a glissando, and the second is a melody taken from [15]. After trying some configurations, we have found a combination of parameters that provides similar results with both methods. For DPW, we have chosen the parameters $T_c = 0$ s, $ID = 0.01$ ST, and $T = 0.001$ s (the minimal value). For ATA we choose just the zero retune speed the minimal value), that as described in the patent generates discrete notes (integers on ST scale). We can see the results in Figure 4 and 5. The f_o-signal treated with DPW is in blue, and the one treated with ATA is in red and the original is in green.

3.2. Expressive Correction with ATA

One of the most important artifacts of vocal expression is vibrato. Expressive Correction is the term we use here to refer to a fast transition within pitch correction that correspond to oscillatory ornaments, particularly vibrato. As we will see, a vibrato with a small amplitude can be shifted around the target pitch with DPW, while it is not well centered under the ATA correction. The expressive correction requires a non-zero transition time parameter. We don't have access to the full implementation of the transition time parameter in ATA (also referred to as the Decay parameter in the

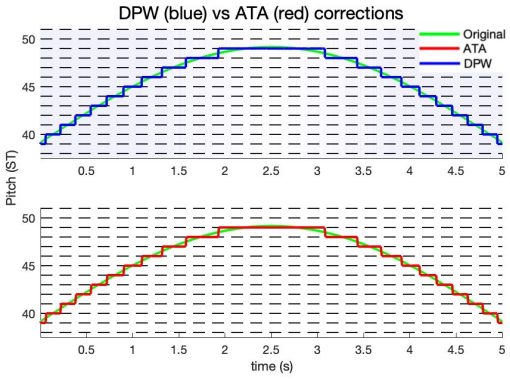


Figure 4: *Extreme correction for a glissando*

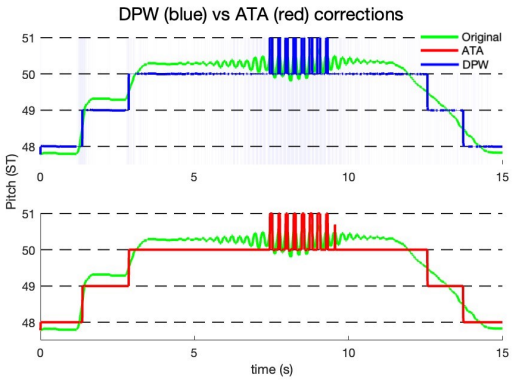


Figure 5: *Extreme correction for an expressive melody*

patent), so we cannot reproduce the exact expected pitch correction of ATA. Therefore, the most effective way to fully understand how the ATA pitch correction algorithm works is to utilize the ATA VST plugin to retune voice samples and extract the corrected f0 from the resulting audio using Praat software [6]. This curves are compared with the DPW correction. To generate the input audio samples, we use Cantor Digitalis (CaD), which is a continuous pitch input synthesizer. CaD takes the trajectory of a wacom stylus, the it generates f_0 and synthesizes a vocal sound. We modified its code to have purposely not-intonated sounds related to the original stylus trajectory. Audio examples can be found in soundcloud [7]. The non-intonated audio samples can be corrected with ATA vist but also with an off-line DPW implementation that we explain later in section 4. Now we proceed to the comparison of both both pitch correction methods.

The simplest case of correction is a shifted note with vibrato. Small vibratos can be effectively corrected with ATA using a retune speed of 50ms. For sustained notes, ATA performs very well and there is no difference with DPW, so we do not present this example here. The difference arises when we have a signal that contains flat notes, free paths, and vibratos. Therefore, it is important to demonstrate how a correction can be performed with ATA using different values of the retune speed parameter, refer to Figure 6. Going up to down we use a retune speed parameter from 0, 15, 50, 100 and 200 ms. The correction is effective at 50ms for the vibrato, but the pitch trajectory after the 12-second mark becomes lost and flattened. Only with a retune speed parameter set to 200ms is it possible to preserve some of the pitch trajectory, but at that configuration, the vibrato is not corrected.

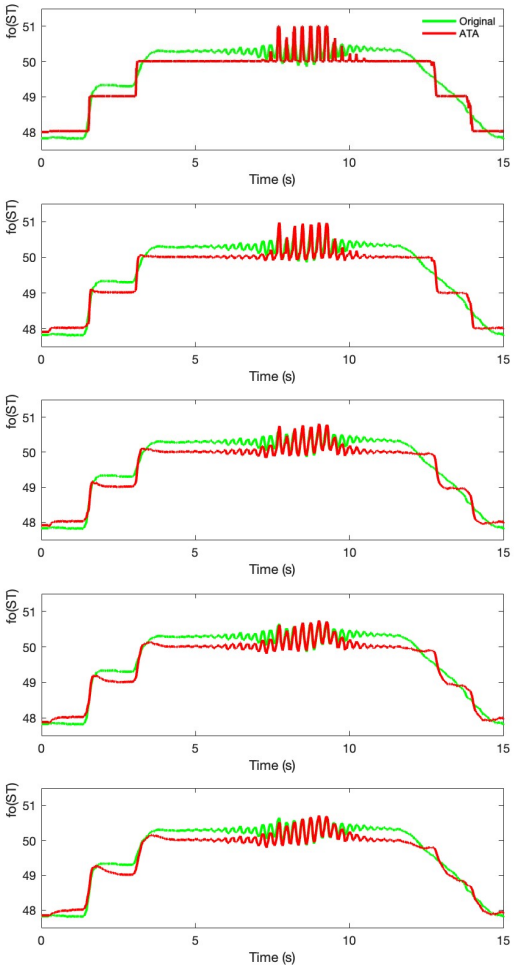


Figure 6: *Correction using different values of retune speed on ATA, RS= 0, 15, 50, 100, 200 ms (up to down)*

Now we will examine the functionality of the ATA flextune parameter. For a more general case, let's now observe what happens when we vary the retune speed while maintaining a specific value for flextune. We have done a configuration with zero retune speed and two values of flextune: zero (red) and 40 cents (violet), figure 7. As we can see, the flextune parameter allows for movement within the range defined by the flextune value after the correction, resulting in the production of smaller ornaments at the output.

In the following example, we will use a non zero value of retune speed, 15ms, and flextune values of zero (red) and 30 cents (violet), as shown in figure 8. As we can see, like the previous example, some ornaments smaller than the flextune value can be preserved at the output.

Now, we present a study with a transition time of 50ms and flextune values of 30 and 60 cents. As we can see in figure 9, a

[6] https://www.fon.hum.uva.nl/praat/

[7] https://on.soundcloud.com/b5NDp last checked: 25 may 2023

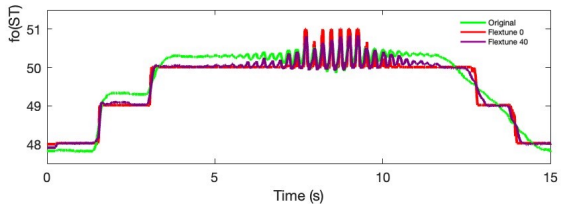


Figure 7: *Correction with ATA, at zero retune speed and flextune: 0 (red) and 40 cents (violet))*

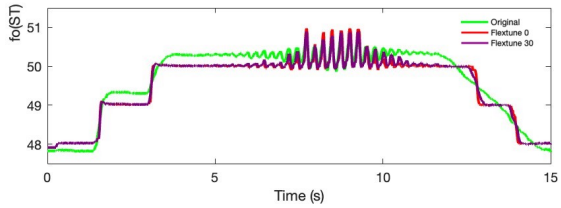


Figure 8: *Correction with ATA, at retune speed equal to 15 ms and flextune: 0 (red) and 30 cents (violet))*

larger value for flextune results in a lack of reactivity. This means the vibrato is not corrected but the path after time equal to 12 s is better preserved than in the other cases. In other words when the notes are well corrected, the general path may be more or less lost depending on the parameters.

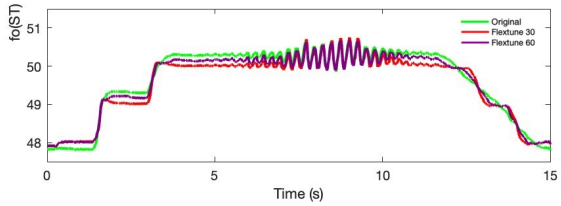


Figure 9: *Correction with ATA, at retune speed equal to 50 ms and flextune: 30 (red) and 60 cents (violet))*

Finally, we show what happens when varying the retune speed for the same flextune parameter. We have chosen a moderate flextune value of 40 cents, while the retune speed varies as follows: 50ms, 100ms, and 200ms. The result can be seen in figure 10. There is always a trade-off between preservation of the main path (free path) and vibrato correction. This means that ATA better preserves the vibrato, but regions such as the one after 12 seconds become staircase-like, resulting in the loss of the original pitch trajectory. In the other hand, parameter values that preserve the shape in that zone, does not correct the vibrato. As we can see in figure 10, the vibrato is not corrected for a retune speed higher than 50ms. On the other hand, when we use flextune at 40 cents and keep zero retune speed (figure 8) the vibrato is corrected but the path after time 12 s is flattened.

3.3. Expressive Correction with DPW

We will show several examples variations of the DPW parameter: critical time and transition time. For the first example, we do choose 100 ms as T_c, then we vary T_t, giving the results in figure

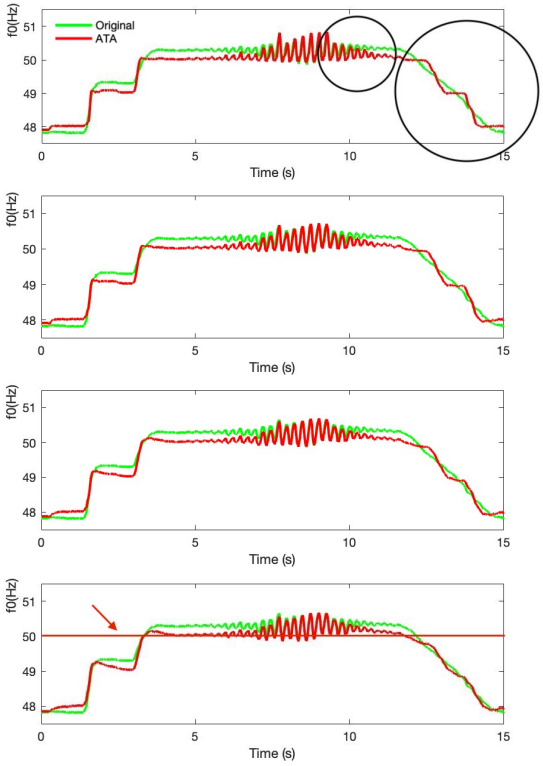


Figure 10: *Correction using different values of retune speed on ATA, RS= 0, 50, 100, 200 ms (up to down) for the same flextune value (40 cents)*

11. As we can see, varying T_t parameter allow us to "smooth" the pitch correction.

Also we have done a correction using a larger critical time equal to 250 ms (optimal according to [11]). It gives the results in figure 12. As we see, the critical time acts as trigger of the correction and the transition time acts as a smoother. The critical time (as parameter) adds an ornament at the beginning of each note step in the staircase region and the transition time modifies the shape of the ornament.

Finally, we have performed a correction using the same transition time (50 ms) while varying the critical time parameter (100 ms, 150 ms, 250 ms). It gives the results in figure 13. As we can see the critical time parameter acts like a trigger for the pitch correction algorithm and the transition time acts as the smoother.

Now we can compare the best configuration for each method. In the case of ATA, it is not possible to achieve good vibrato correction and good preservation of the free path simultaneously. Therefore, we preferred a moderate configuration that performs reasonably well for both purposes. A suitable ATA configuration is a retune speed of 100 ms and flextune of 40 cents (figure 13). For DPW the most suitable correction is done by choosing the critical time as 200 ms (DPW) and then we can choose for example a transition time equal to 50 ms (figure 10) . For simplicity we have put these two cases in the figure 14). This shows that DPW performs a better correction: Firstly the vibrato is well centered in DPW correction while not in ATA; and secondly the DPW preserve better the f_o-path after time 12 s, while ATA flatten it. In contrast, ATA seems visually better in the segment before 5 s while DPW

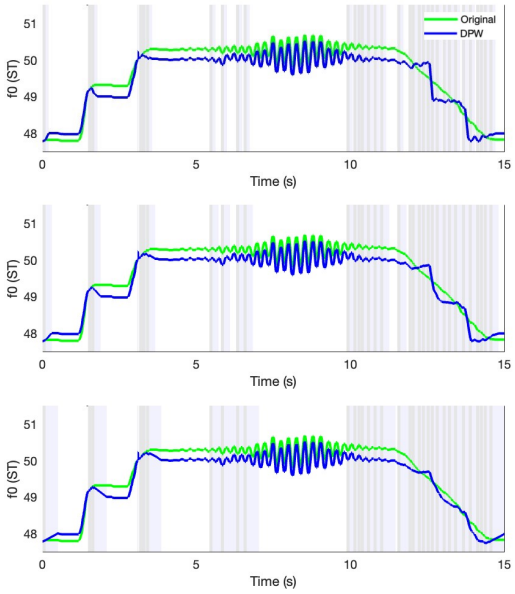


Figure 11: *Correction using different values of transition time in DPW (from up to down: 100,200,400 ms), for the same critical time (100 ms)*

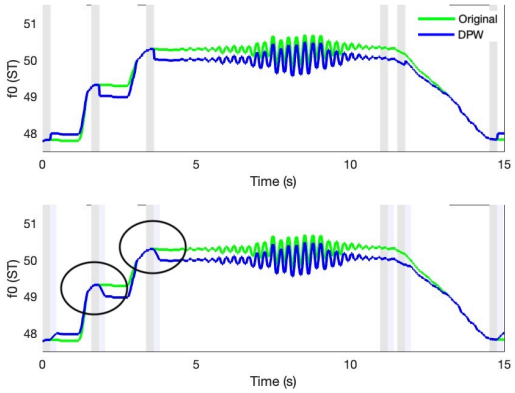


Figure 12: *Correction using different values of T_t in DPW (from up to down: 25,200 ms), for the same T_c (250 ms)*

present an more visible expressive ornament. In the subsequent subsection, we will showcase the measurements that are directly linked to the aforementioned observations, as we will see DPW is closer to the original f_o curve for all the regions.

3.4. Comparison through MSE and MAE

The difference between two curves can be measured in various ways, here we presented two. Firstly, the Mean Squared Error (MSE) that measures the sensitivity to quadratic errors; it is calculated through the difference of squares, which gives larger errors a greater impact on the overall result. MSE also provides a measure of variance between the curves. Secondly, the Mean Absolute Error (MAE) that provides a measure of the average difference in magnitude between the curves, unlike MSE, MAE does not amplify larger errors. We use the follwoing equations:

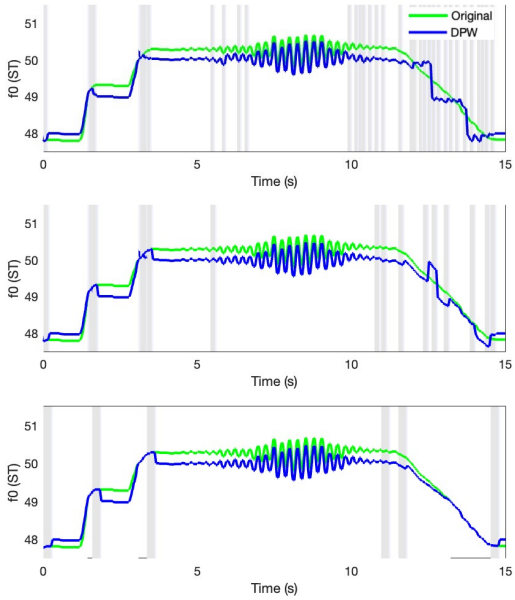


Figure 13: *Correction using different values of T_c in DPW (from up to down: 100,150,250 ms), for the same T_t (50 ms)*

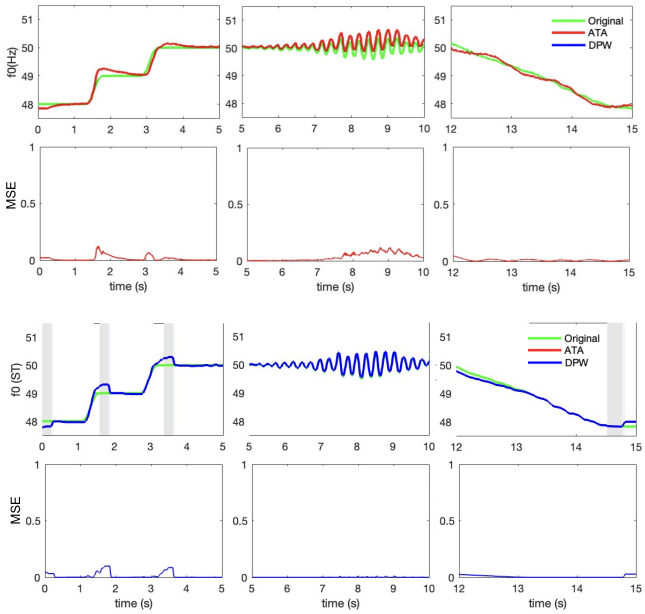


Figure 14: *Correction for the same T_t (50 ms) using flextune at 40 cents for ATA and T_c at 200 ms for DPW and the corresponding MSE.*

$$\text{Mean of MSE} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{n_j}\sum_{i=1}^{n_j}(y_{ij}-\hat{y}_{ij})^2\right) \tag{6}$$

$$\text{Mean of MAE} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{n_j}\sum_{i=1}^{n_j}|y_{ij}-\hat{y}_{ij}|\right) \tag{7}$$

Where N represents the number of samples, n_j is equal to 1, cause there always a comparison of one curve with the reference, j represents the curve to compare (ATA or DPW), y_{ij} are the values of the original curve j, and $\hat{y}_{ij}$ are the values of the comparison curve j.

Our example is helpful to highlight three types of pitch modification. The first part in $0 < t < 5$ where signal is like a staircase between the notes 48,49 and 50. The second part in $5 < t <$ 0represents the correction of a poorly intonated frequency modulation, similar to the human vibrato. And the third part is a soft path of a f_o trajectory that should not be corrected, the free path represents the case where the singer do not have the intention to play any specific note. Each part must be compared to the desirable pitch curve, which is different for each region. For example for the staircase part, the desired signal is a staircase. For the vibratory part the ideal pitch would be the same vibration but well centered. And for the third part, the original signal would be the ideal pitch, rather than a correction we want to preserve it. These assumptions are illustrated on figure 6, the calculation of MSE is done point by point. The mean over each region is reported in Table 1. As it is shown and mentioned before, DPW perform better correction of vibratos while preserving the free path of the note, and ATA is better for the staircase part while losing more of the vibrato and free path parts.

Table 1: *MSE and MAE between input and corrected f_0 for the different regions*

	MSE		MAE	
Region	DPW	ATA	DPW	ATA
1	0.0146	0.0146	0.0747	0.0914
2	0.0415	0.0642	0.1304	0.2103
3	0.0539	0.0280	0.2015	0.1463

Please note that all the comparison are focused on the pitch correction curves. For DPW we use the pitch correction method that is different than the full algorithm audio. The implementation of the vocoder, described in section 4, is a complex process and the vocoder we have use in making the audio DPW tracks is not as advanced as the vocoder of ATA. As a result, some imprecision may be present in the generated f_0 paths for the DPW audio examples. Despite these limitations, it is worth highlighting the valuable insights gained from this comparison, which shed light on the respective strengths and weaknesses of each method.

4. IMPLEMENTATION OF AN OFF-LINE AUDIO PITCH CORRECTION

This section talks about the off-line implementation of DPW. DPW works in an analogous way to Cantor Digitalis. However, instead of an incoming f_0 given by a table, we use an f_0 value obtained from a pitch tracker on a pre-recorded vocal audio track. The general structure for a autotune system is conformed by: a pitch tracker, a pitch correction algorithm, and a pitch warping algorithm (vocoder). DPW can follow a similar approach using a pitch tracker to acquire f_0.

4.1. Development of the off-line retuner

We developed a methodology for off-line vocal retuning using the DPW method; this process requires obtaining F0 data and a

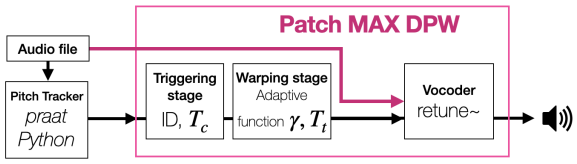


Figure 15: *Configuration of the offline retuner*

transparent vocoder as shown in 15. For pitch tracking, we utilized Praat[8] (software to analyze audio prosody), the *To PitchTier* method allows us to obtain F0 curves for the original audios within a Praat file sampled at Praat time intervals. We did a Python code (with package *wave*) to extract the file's relevant data and to create arrays for time and f_0 information; the arrays were re-sampled at the original audio files sampling rate. The *pathlib* package was employed to process multiple sound library files simultaneously, resulting in a library of the original audios and the f_0 files. The Max/MSP environment was used to process the f_0 information (on Semi tones and Hz) and write retuned audio files using the different vocoders (retune∼, freqshift∼, pitchshift∼, supervp∼, etc). Our goal was to identify the most transparent vocoder that generated a voice signal closest to the input F0, using the original f_0 data, the *retune∼* object was selected as the most transparent modification for the entire library; this ensured that the vocoder avoided introducing sound artifacts that could affect the perception of quality and retuning. However, the overall quality of the presented vocoder, retune∼, is not as precise and good as the ATA vocoder. Therefore, the resulting audio tracks using retune∼ may not be as good as those using the ATA vocoder. Therefore, we dispose of an alternative option, with an wrapper of the *World* [16]vocoder, provided by the research engineers of Lutherie-Acoustique-Musique Group, the audio obtained with *World* is done through a non-real-time transposition through python. The resulting audio has a better quality than the MAX implementation. The sound library for DPW correction using both vocoders can listen on the soundcloud playlist noted in section 3.2.

5. CONCLUSIONS

Through our research, we studied DPW algorithm for audio pitch correction. It is possible to control and trigger a pitch correction thanks to three degrees of freedom that preserves low-amplitude vibratos and ornaments in the neighborhood of the target note. We have also shown how the pitch correction methods are composed of two stages (triggering and warping), and how the modification of the control parameters can lead to equivalent configurations for different systems. We have identified a scenario where ATA and DPW exhibit similarity: extreme correction. Moreover, we have identified three types of correction: staircases, vibratos, and free paths, and have illustrated that DPW performs better for vibratos and free paths, while also being adequate for staircase correction. DPW also exhibits less trade-off between its parameters compared to ATA.

In addition, we have developed an audio application that includes the DPW method. Compared to ATA, its control parameters allow for a smooth pitch trajectory transition towards the nearest

[8] https://www.fon.hum.uva.nl/praat/

notes on a defined scale, minimizing distortion of melodic ornaments between the notes. However, it is important to note that the vocoder used in our application (retune∼) may not provide the same level of quality, precision and accuracy as the ATA vocoder.

We plan to undertake a comprehensive perceptual evaluation of the two systems in a formal setting. This evaluation aims to assess the perceptual salience of the pitch effects introduced by the DPW method, as well as their potential musical relevance.

6. ACKNOWLEDGMENTS

This research was funded through ANR National Research Agency projects: Analysis and Transformation of Singing Style (ANR-19-CE38-0001) and Gepeto: GEsture and PEdagogy of inTOnation (ANR-19-CE28-0018)

7. REFERENCES

[1] C. Vincent, *La voix chanteé*, chapter De l'antipop à l'Autotune, pp. 123–142, N. Henrich & De Boeck Solal, 2013.

[2] P. Boulez and A. Gerzso, "Computers in music," *Scientific Amer.*, vol. 258, no. 4, pp. 44–51, April 1998.

[3] A. Wilson and B. Fazenda, "Perception and Evaluation of Audio Quality in Music Production," in *Proc. of the 16th Int. Conf. on Digit. Audio Effects*, Maynooth, Ireland, September 2013, pp. 68–77.

[4] J.M Chowning, "Digital sound synthesis, acoustics and perception: A rich intersection," in *Proc. of the Int. Conf. on Digit. Audio Effects*, Verona, Italy, December 2000, pp. 1–6.

[5] H. Hildebrand, "Pitch detection and intonation correction apparatus and method," Auburn Audio technologies, Auburn, AL, USA Patent US5973252A, G10H-007/00, Oct. 14, 1992, pp 10-18.

[6] L. Haken, "Position correction for an electronic musical instrument," Champaign, IL, US Patent 76191562009, Int GIOH 1/22, Apr. 19, 2007, pp 6–12.

[7] L. Haken, E. Tellman, and P.Wolfe, "An indiscrete music keyboard," *Comput. Music J.*, vol. 22, no. 1, pp. 30–48, Spring 1992.

[8] L. Feugère, C. d'Alessandro, B. Doval, and O. Perrotin, "Cantor digitalis: chironomic parametric synthesis of singing," *EURASIP J. on Audio, Speech, and Music Process.*, vol. 2, pp. 98 1–19, December 2017.

[9] Sebastian Rosenzweig, Simon Schwär, Jonathan Driedger, Meinard MüllerA. Wilson, and B. Fazenda, "Adaptive pitch-shifting with applications to intonation adjustment in a cappella recordings," in *Proc. of the 24th Int. Conf. on Digit. Audio Effects*, Vienne, Austria, September 2021, pp. 121–128.

[10] N. Orio, N. Schnell, and M.M. Wanderley, "Input devices for musical expression: Borrowing tools from hci," in *Proc. of the Int. Conf. on New Interfaces for Musical Expression*, Seattle, USA, April 2001, pp. 62–76.

[11] O. Perrotin and C. D'alessandro, "Target acquisition vs. expressive motion: Dynamic pitch warping for intonation correction," *ACM Transactions on Computer-Human Interaction*, vol. 23, no. 3, pp. 17 1–21, June 2016.

[12] S. Rossignol, P. Depalle, J. Soumagne, X. Rodet, and J.-L. Collette, "Vibrato: Detection, estimation, extraction, modification," in *Proc. of the Int. Conf. on Digit. Audio Effects*, Verona, Italy, December 1999, pp. 1–6.

[13] R. Lamb and A.N. Robertson, "Seaboard: a new piano keyboard-related interface combining discrete and continuous control," in *Proc. of the Int. Conf. on Digit. Audio Effects*, Oslo, Norway, June 2011, pp. 503–506.

[14] A.P. McPherson, A. Gierakowski, and A.M. Stark, "The space between the notes: Adding expressive pitch control to the piano keyboard," in *Proc. of the SIGCHI Conf. on Human Factors in Comput. Syst.*, New York, NY, USA, June 2013, p. 2195–2204.

[15] O. Perrotin and C. d'Alessandro, "Quel ajustement de hauteur mélodique pour les instruments de musique numériques?," in *Journées d'Informatique Musicale (JIM 2015)*, Montréal, Canada, May 2015, pp. 186–189.

[16] K. Ozawa M. Morise, F. Yokomori, "World: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems*, vol. E99.D, no. 7, pp. 1877–1884, July 2016.

[17] V. Verfaille, U. Zölzer, and D. Arfib, "Adaptive digital audio effects (a-dafx): a new class of sound transformations," *IEEE Transactions on Speech and Audio Processing 14 (5)*, pp. 1817– 1831, 2006.

[18] N. Zacharov, *Sensory Evaluation of Sound*, CRC Press, Boca Raton, FL, USA, first edition, 2019, pp. 60–99, 107-134.

[19] J. Chowning, *Music, Cognition, and Computerized Sound*, MIT Press, Cambridge, Massachusetts, London, England, 1999, pp. 261–276.

[20] S. Bernsee and D. Gökdag, "Methods for extending frequency transforms to resolve features in the spatio-temporal domain," Zynaptiq GmbH , Hannover, Germany, USA Patent 11079418 B2, Aug. 23 , 2018, pp 26–41.

[21] Cycling '74. CA, USA, "Max online documentation," Accessed: 19.09.2022. [Online]. Available: https://docs.cycling74.com/.

[22] Ircam. Paris, France, "Supervp for max max reference pages 11/2012," [Online]. Available: https://forum.ircam.fr/media/uploads/software/SuperVP%20for%o20Max/supervp-for-max.pdf, published Nov-2012, pp 1–14.

VOCAL TRACT AREA ESTIMATION BY GRADIENT DESCENT

David Südholt
Centre for Digital Music
Queen Mary University of London
London, UK
d.sudholt@qmul.ac.uk

Mateo Cámara *
Information Processing & Telecomm. Center
Universidad Politécnica de Madrid
Madrid, Spain
mateo.camara@upm.es

Zhiyuan Xu, Joshua D. Reiss
Centre for Digital Music
Queen Mary University of London
London, UK
zhiyuan.xu@qmul.ac.uk
joshua.reiss@qmul.ac.uk

ABSTRACT

Articulatory features can provide interpretable and flexible controls for the synthesis of human vocalizations by allowing the user to directly modify parameters like vocal strain or lip position. To make this manipulation through resynthesis possible, we need to estimate the features that result in a desired vocalization directly from audio recordings. In this work, we propose a white-box optimization technique for estimating glottal source parameters and vocal tract shapes from audio recordings of human vowels. The approach is based on inverse filtering and optimizing the frequency response of a waveguide model of the vocal tract with gradient descent, propagating error gradients through the mapping of articulatory features to the vocal tract area function. We apply this method to the task of matching the sound of the Pink Trombone, an interactive articulatory synthesizer, to a given vocalization. We find that our method accurately recovers control functions for audio generated by the Pink Trombone itself. We then compare our technique against evolutionary optimization algorithms and a neural network trained to predict control parameters from audio. A subjective evaluation finds that our approach outperforms these black-box optimization baselines on the task of reproducing human vocalizations.

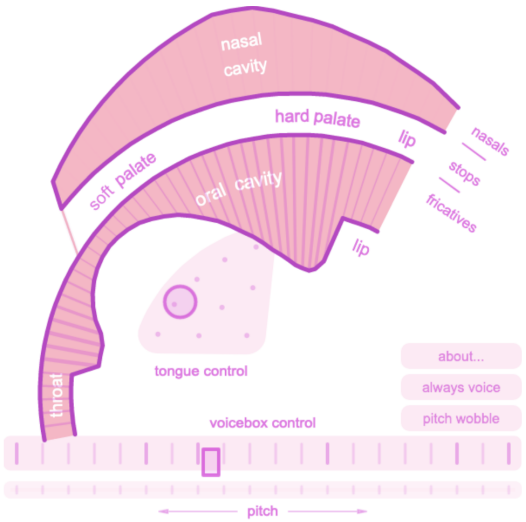


Figure 1: *The user interface of the Pink Trombone articulatory synthesizer.*

1. INTRODUCTION

Articulatory synthesis is a type of speech synthesis in which the position and movement of the human articulators, such as the jaw, lips or tongue, are used as control parameters. Because of their inherent interpretability, articulatory features lend themselves well towards fine-grained and flexible user control over the speech synthesizer [1]. Articulatory Synthesis is typically implemented as a physical model, which simulates the propagation of air pressure waves through the human vocal tract. A large number of such models have been developed over the years [2].

Obtaining the articulatory features that control the physical model is not a trivial problem. Area functions of the vocal tract can be directly measured with magnetic resonance imaging (MRI) [3] or electromagnetic articulography (EMA) [4]. However, these procedures are time-consuming, susceptible to noise and variations, and require access to specialized equipment. It is therefore desirable to recover the articulatory features directly from a given speech signal. In general, this task is known as Acoustic-to-Articulatory Inversion (AAI). Two main strands of research can be identified: one is data-driven AAI, which seeks to develop statistical methods based on parallel corpora of speech recordings and corresponding MRI or EMA measurements [5, 6]. The other takes an analysis-by-synthesis approach to AAI, in which numerical methods are developed to both obtain acoustic features from articulatory configurations, and to invert that mapping to perform AAI [7, 8, 9].

In this work, we focus on the analysis-by-synthesis approach and consider the specific articulatory features that make up the control parameters of an articulatory synthesizer. The AAI task is then framed as obtaining control parameters such that the synthesizer reproduces a target recording. This allows a user to reproduce that vocalization with the articulatory synthesizer, and then modify parameters such as vocal tract size, pitch, vocal strain, or vowel placement.

Attempts to solve this problem of *sound matching*, for articulatory synthesis or other types of synthesis, can generally be classified into *black-box* and *white-box* methods.

Black-box methods do not rely on information about the structure of the synthesizer. A popular approach is to use derivative-free optimization techniques such as genetic algorithms [10, 11, 12, 13, 14] or particle swarm optimization [15]. These methods are computationally expensive and can take many iterations to converge

* Work performed as part of an academic visit to the Centre for Digital Music, Queen Mary University of London

to a solution. Various deep neural network (DNN) architectures have also been proposed to predict control parameters that match a given sound [16, 17, 18, 19, 20]. They require constructing high-quality datasets for training that cover the space of acoustic outputs.

White-box methods can improve the sound matching of specific synthesizers by incorporating knowledge of their internal structure. This can be done by reasoning about their underlying physical processes [21, 22] or, more recently, making use of auto-differentiation and gradient descent techniques [23, 24, 25, 26].

In this work, we propose a gradient-based white-box optimization technique for sound matching vowel sounds with the articulatory synthesizer known as the Pink Trombone (PT)[1]. The PT is a web application that uses well-known models of the glottal source and the vocal tract to implement an intuitively controllable vocal synthesizer. Its user interface is depicted in Figure 1.

Our technique works as follows. First, we decompose a recording into a glottal source signal and an IIR filter with existing inverse filtering methods. We then obtain a vocal tract configuration by minimizing the difference between an analytical formulation of the tract's transfer function [27] and the IIR filter with gradient descent. A differentiable implementation of the mapping between control parameters and the vocal tract configuration allows propagation of the error gradient directly to the control parameters. Section 2 describes the details of our approach.

We find that this approach can accurately recover the vocal tract area function on vowel sounds generated by the PT itself. A subjective listening test shows that without requiring any training procedures, the approach outperforms black-box baselines on the task of reproducing real human vocalization. The results of the objective and subjective evaluations are presented in section 3. Section 4 concludes the paper.

2. METHOD

The PT is based on the widely used source-filter model of speech production. The speech output $S(z) = G(z)V(z)L(z)$ is assumed to be the combination of three linear time-invariant (LTI) systems: the glottal flow G, the vocal tract V, and the lip radiation L. The lip radiation is approximated as a first-order differentiator $L(z) = 1 - z^{-1}$ and combined with G to form a model of the *glottal flow derivative* (GFD). Speech is then synthesized by generating a GFD signal (the source) and filtering it through the vocal tract V.

In our sound matching approach, a target sound is first decomposed into the GFD source waveform and coefficients for an all-pole filter, using the inverse filtering technique proposed in [28]. The control parameters for the PT glottal source are then obtained directly from the GFD waveform. We propose an objective function based on the magnitude response of the all-pole filter that allows estimating the control parameters of the vocal tract with gradient descent. The overall method is illustrated in Figure 2. The source code is available online[2].

2.1. Inverse Filtering

To separate target audio into a GFD waveform and a vocal tract filter, we use the Iterative Adaptive Inverse Filtering method based on a Glottal Flow Model (GFM-IAIF) [28].

[1] `https://dood.al/pinktrombone`

[2] `https://github.com/dsuedholt/vocal-tract-grad`

IAIF methods in general obtain gross estimates of G, V and L with low-order LPC estimation, and then iteratively refine the estimates by inverse filtering the original audio with the current filter estimates, and then repeating the LPC estimation at higher orders.

GFM-IAIF makes stronger assumptions about the contribution of the glottis G, and uses the same GFD model as the PT synthesizer (compare section 2.2), making it a good choice for our sound matching task.

From GFM-IAIF, we obtain an estimate for the vocal tract filter V in the form of $N + 1$ coefficients $a_0, \dots a_N$ for an all-pole IIR filter:

$$V(z) = \frac{1}{\sum_{i=0}^{N} a_i z^{-i}} \tag{1}$$

This also gives us an estimate of the GFD waveform by inverse filtering the original audio through V, i.e. applying an all-zero FIR filter with feed-forward coefficients $b_i = a_i$.

2.2. Glottal Source Controls

The PT uses the popular Liljencrants-Fant (LF) model to generate the GFD waveform. Originally proposed with four parameters [29], the LF model is usually restated in terms of just a single parameter R_d, which is known to correlate well with the perception of vocal effort [30].

R_d can be obtained from the spectrum of the GFD. Specifically, [31] finds the following linear relationship between R_d and $H_1 - H_2$, the difference between the magnitudes of the first two harmonic peaks of the GFD spectrum (measured in dB):

$$H_1 - H_2 = -7.6 + 11.1 R_d \tag{2}$$

We estimate the fundamental frequency F_0 using the YIN algorithm [32], and measure the magnitudes of the GFD spectrum at the peaks closest to F_0 and $2 \cdot F_0$ to calculate $H_1 - H_2$ and thus R_d.

However, the PT does not use R_d as a control parameter directly. Instead, it exposes a "Tenseness" parameter T, which relates to R_d as $T = 1 - R_d/3$.

T is clamped to values between 0 and 1, with higher values corresponding to higher perceived vocal effort. Additionally, the PT adds white noise with an amplitude proportional to $1 - \sqrt{T}$ to the GFD waveform, to give the voice a breathy quality at lower vocal efforts. Figure 3 shows the glottal source at varying Tenseness values.

The estimated control parameters F_0 and Tenseness correspond to the horizontal and vertical axes in the PT's "voicebox" UI element, respectively (see Figure 1).

2.3. Vocal Tract

While the glottal source affects voice quality aspects like breathiness and perceived effort, the vocal tract is responsible for shaping the source into vowels and consonants.

In the PT, the vocal tract is treated as a sequence of $M + 1$ cylindrical segments, with $M = 43$. The shape of the vocal tract is then fully described by its *area function*, i.e. the individual segment cross-sectional areas $A_0, \dots, A_M$. Noting that $A = \pi(d/2)^2$, the area function may equivalently be described by the segment diameters $d_0, \dots, d_M$.

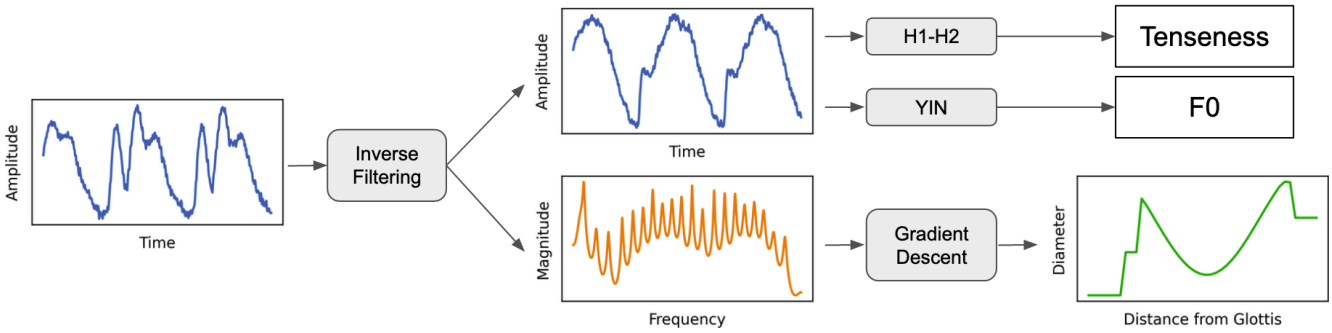


Figure 2: *Illustration of the proposed sound matching method. Target audio is inverse filtered to obtain a source waveform and the transfer function of a filter. For resynthesis, the glottal control parameters F_0 and Tenseness are estimated from the source waveform. The vocal tract area function is optimized with gradient descent to match the filter's transfer function.*

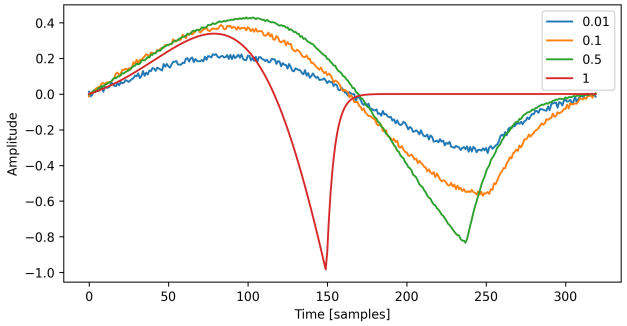


Figure 3: *A single cycle of the glottal source waveform of the Pink Trombone, which combines the LF model with white noise, at varying values of the Tenseness parameter.*

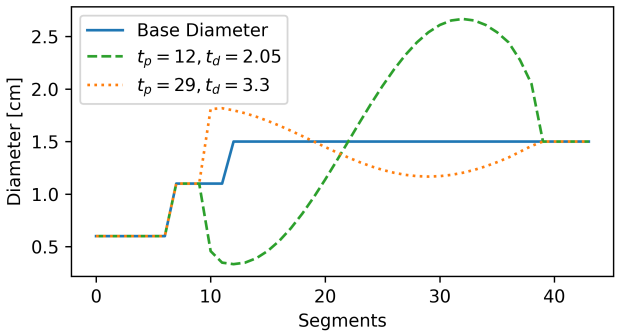


Figure 4: *Example plots of the rest diameter, i.e. the result of applying the tongue model to the base diameter, at different tongue positions t_p and tongue diameters t_d.*

An additional, similar model of the nasal tract is coupled to the vocal tract at the soft palate. However, for the open vowel sounds that we are considering, the soft palate is closed and the coupling effect is negligible. In the PT implementation, the soft palate only opens when parts of the vocal tract are fully constricted, therefore here we focus only on the vocal tract itself.

2.3.1. Control Model

Directly specifying each segment diameter individually does not make for an intuitive user experience and could easily result in very unrealistic, strongly discontinuous area functions. Instead, the PT implements a tiered control model over the vocal tract based on the model proposed in [33].

The control model consists of two tiers. The first tier is a tongue defined by a user-specified diameter t_d and position t_p. The tongue shape is modeled as sinusoid shape and modifies a *base diameter*, representing a neutral area function, into the *rest diameter*. Figure 4 illustrates this.

The second control tier are *constrictions* that the user can apply to the rest diameter at any position along the vocal tract. Similarly to the tongue, constrictions are defined by an index, a diameter, and a model of how they affect the rest diameter. There are however two differences between the tongue and the constrictions: Firstly, constrictions are optional, while the tongue is always present. Secondly, constrictions can fully close the vocal tract, at

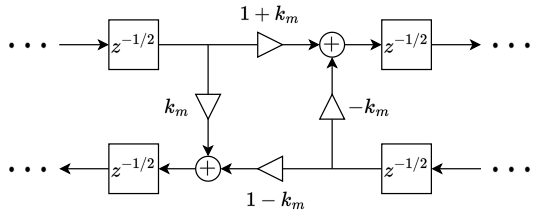


Figure 5: *Block diagram of a scattering junction in the Kelly-Lochbaum model, with scattering coefficient k_m.*

which point noise is inserted to model plosives and fricatives. For this work, we consider only open area functions, meaning that we do not allow constrictions to reduce the diameter below a certain threshold.

2.3.2. Estimating the Area Function

Propagation of the glottal source through the vocal tract is modeled by implementing each cylindrical segment as a bidirectional, half-sample delay. The half-sample delay is achieved by processing the signal at twice the audio sampling rate and adding up adjacent pairs of samples. At the M inner junctions, the change in cross-sectional area leads to reflection and refraction, described by

scattering coefficients calculated from the segment areas as

$$k_m = \frac{A_m - A_{m-1}}{A_m + A_{m-1}} \text{ for } m = 1, \dots M. \tag{3}$$

This is the well-known Kelly-Lochbaum (KL) model [34]. An illustration of a scattering junction is shown in Figure 5.

The length of the simulated vocal tract results from the number of segments and the sampling rate. Considering a speed of sound in warm air of $c \approx 350$ m/s and an audio sampling rate of $f_s = 48000$ Hz, implementing half-sample delays as unit delays processed at $2 \cdot f_s$, $M + 1 = 44$ segments result in a vocal tract length of $44 \cdot 350/(2 \cdot 48000) \approx 0.16$ m. This corresponds to the vocal tract of an average adult male [33], giving the PT a male voice. The number of segments and the unit delays are fixed in the PT. The KL model can be implemented more flexibly through e.g. the use of fractional delays [35].

An analytical transfer function for the piecewise cylindrical model using unit delays was derived in [27]. The formulation can be straightforwardly adapted to half-sample delays by replacing every delay term z^{-n} with $z^{-n/2}$, and then applying an additional factor of $1 + z^{-1}$ to account for the summing of adjacent samples. The transfer function H_{KL} can then be stated as:

$$H_{\text{KL}}(z) = \frac{(1 + z^{-1}) z^{-(M+1)/2} \prod_{m=1}^{M} (1 + k_m)}{K_{1,1} + K_{1,2} R_L - R_0 (K_{2,1} + K_{2,2} R_L) z^{-1}} \tag{4}$$

R_0 and R_L are the amount of reflection at the glottis and lips, respectively, and $K \in \mathbb{R}^{2\times 2}$ is defined as follows:

$$K = \begin{bmatrix} K_{1,1} & K_{1,2} \\ K_{2,1} & K_{2,2} \end{bmatrix} = \prod_{m=1}^{M} \begin{bmatrix} 1 & k_m z^{-1} \\ k_m & z^{-1} \end{bmatrix} \tag{5}$$

We now wish to find the tongue controls and constrictions such that $|H_{\text{KL}}|$ approximates $|V|$, the magnitude response of the vocal tract recovered by inverse filtering.

In an approach inspired by [24], we now consider the squared error between the log of the magnitude responses for a given angular frequency $0 \leq \omega < \pi$:

$$E(\omega) = \left(\log_{10} |H_{\text{KL}}(e^{i\omega})| - \log_{10} |V(e^{i\omega})| \right)^2 \tag{6}$$

We can then define a loss function that measures how closely a given vocal tract area function matches the recovered vocal tract filter by evaluating the mean squared error over a set of F linearly spaced frequencies:

$$\mathcal{L} = \frac{1}{F} \sum_{f=0}^{F-1} E(\frac{f}{F}\pi) \tag{7}$$

We can then find the set of controls that minimizes $\mathcal{L}$, meaning that the corresponding area function approximates $|V|$. A schematic overview of the computation graph is shown in Figure 6.

3. EXPERIMENTS AND RESULTS

We first evaluated the performance of our approach on recovering control parameters for sounds generated by the PT itself. These *in-domain* sounds are guaranteed to be within the possible output space of the PT, and the ground truth parameters are known.

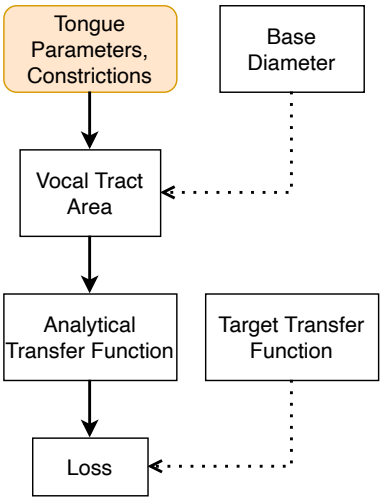


Figure 6: *Schematic overview of the computation graph. In the forward pass, an area function is calculated from the control parameters. The corresponding transfer function is then computed and used to calculate the loss. Solid arrows denote that the operations are implemented to support auto-differentiation. This allows updating the estimate of control parameters (tongue and constrictions) using the gradient of the loss.*

We then applied our approach to estimating control parameters for *out-of-domain* sounds that were not generated by the PT itself. Ground truth parameters that provide an exact match are not known and likely do not exist due to limitations of the model, which makes evaluation challenging. We performed a listening test to compare the quality of our method to previously proposed, model-agnostic black-box sound matching approaches.

For all evaluations, parameter ranges were normalized to $[0, 1]$. Gradient descent was performed for 100 steps, with a step size of 10^{-4} and a momentum of 0.9.

3.1. Reconstructing PT-generated Audio

3.1.1. Setup

For the in-domain evaluation, we generated 3000 total sets of control parameters and attempted to recover the vocal tract area. For all examples, F_0 was uniformly sampled from $[80, 200]$, the tenseness from $[0, 1]$, the tongue position t_p from $[12, 29]$ (measured in segments along the tract), and the tongue diameter t_d from $[2.05, 3.5]$. The range of F_0 roughly covers the pitch range of adult male speech, while the other control parameter ranges cover the range of possible values defined by the PT interface.

The parameters were divided in three sets of 1000 examples each. The first set was taken as-is. A random constriction, with position sampled from $[0, 43]$ and diameter sampled from $[0.3, 2]$, was applied to the vocal tract in the second set. Two such independently sampled constrictions were applied in the third set.

For each example, we performed the gradient descent optimization twice with different targets: First, with the target response $|V|$ taken directly from the ground truth frequency response (FR) of the original vocal tract. Since this FR is guaranteed to be within the domain of the KL vocal tract model, it should be able to be matched very closely.

Table 1: *MAE values for recovering control parameters when the target transfer function of the vocal tract (VT) is either given from the ground truth area function, or obtained by inverse filtering (IF).* $t_p \in [12, 29]$ *is the (continuous) position of the tongue along the vocal tract.* $t_d \in [2.05, 3.5]$ *is the tongue diameter.*

# of Constrictions	0		1		2	
VT Transfer Function	Given	IF	Given	IF	Given	IF
t_p [-]	0.19	1.42	1.21	1.93	1.74	2.15
t_d [cm]	0.02	0.26	0.12	0.28	0.19	0.32
Total Diameter [cm]	0.01	0.23	0.07	0.24	0.11	0.24
Frequency Response [dB]	0.13	2.09	0.60	2.33	0.87	2.50

Second, with the target response $|V|$ recovered by the GFM-IAIF method. This is no longer guaranteed to have an exactly matching vocal tract configuration, so higher deviation is expected. However, since GFM-IAIF and the PT are based on similar assumptions about the source-filter model, the obtained target responses match the ground truth closely enough to be useful in recovering the original control parameters.

3.1.2. Results

Table 1 shows the mean absolute error (MAE) for the tongue parameters t_p and t_d for each condition. Additionally, the MAE values for the total area function (i.e. the diameter of each individual segment) and the recovered FR are given.

In the simple case of optimizing the true FR with no constrictions applied, the original vocal tract area could be recovered with very high accuracy, often to an exact match. Constrictions introduce more degrees of freedom and result in a less accurately recovered area function, although the FR was still matched very closely. Figure 7 illustrates how visibly different area functions can have very similar frequency responses. This relates to the transfer function in equation (4) not depending on the area directly, but rather on the resulting reflection coefficients in equation (3). The locations of the area function's extrema, i.e. the segments at which the area changes from growing wider to growing more narrow or vice versa, therefore affect the transfer function more strongly than the specific value of a given area segment.

Since the FR obtained by GFM-IAIF might not be able to be matched exactly by the KL model, some constrictions might be used during the estimation even if there were none applied to the original vocal tract, leading to deviations from the true area function. An example of this is shown in Figure 8. The range of frequencies most affected by this depend on the choice of LPC estimation in GFM-IAIF; as noted in [28], modeling the glottal contribution as a 3^{rd} order filter is well-motivated by the LF model and gives balanced results in practice.

Due to the presence of this error introduced through inverse filtering, applying constrictions to the ground truth area function had a considerably less pronounced effect on the error metrics when the FR obtained by GFM-IAIF is used as the optimization target.

Inverse filtering also noticeably affected the estimation of the glottal source parameters. The MAE for the prediction of the tenseness $T \in [0, 1]$ was 0.013 when the original GFD waveform was used, but rose to 0.057 when the GFD waveform was recovered by inverse filtering. Even the accuracy of the YIN fundamental frequency estimator dropped slightly: the MAE for $F_0 \in [80, 200]$ was 0.04 on the original GFD waveform, and 0.44 on the recovered GFD waveform.

Applying constrictions had no effect on the glottal source parameter estimation. Grouping the MAE values by the number of constrictions result in values deviating less than 0.5% from the reported global MAE values for both T and F_0.

3.2. Sound Matching Human Vocalizations

3.2.1. Black-Box Baselines

To assess the out-of-domain performance, we performed a subjective evaluation comparing our gradient-based approach against three black-box optimization methods that have previously been used for the task of sound matching.

Genetic algorithms [10, 11, 12, 13, 14] employ a population of candidate solutions, which evolve through generations by applying genetic operators such as selection, crossover, and mutation. The fittest individuals, evaluated through a fitness function, are more likely to reproduce and pass on their traits to offspring.

Particle Swarm Optimization (PSO) [15] involves a group of candidate solutions, called particles, that move through the search space to find the global optimum. Each particle's position is updated based on its own best-known position, the best-known position within its neighborhood, and a random component, with the goal of balancing exploration and exploitation.

For both the genetic algorithm and PSO, scores for a given set of parameters were calculated as the mean squared error between the mel-spectrogram of the target audio, and the audio generated by the PT with the current parameters.

Neural parameter prediction [16, 17] uses a neural network to predict parameters from audio. We train a convolutional neural network (CNN) architecture with two convolutional layers separated by a max-pooling layer and followed by three fully connected layers on a dataset of 1,000,000 randomly sampled parameter sets and their corresponding mel-spectrograms.

While the in-domain evaluation focused on static vocal tract configurations, the speech samples used in the out-of-domain evaluation are time-varying. For all baselines and the gradient-based approach, this is handled by estimating the parameters on a frame-by-frame basis. To avoid sudden jumps in the area, the predictions of the baselines were smoothed over time by applying a Savitzky-Golay filter [36]. For our gradient approach, the estimation of each frame was initialized with the previous frame's prediction.

3.2.2. Listening Test

We reproduced 6 short recordings of human vocalizations with each method. The originals and the reproductions, and the individual ratings are available online.[3] The pitch, breathiness, and vowel shape of the recordings is time-varying. Each recording came from

[3] `https://dsuedholt.github.io/vocal-tract-grad/`

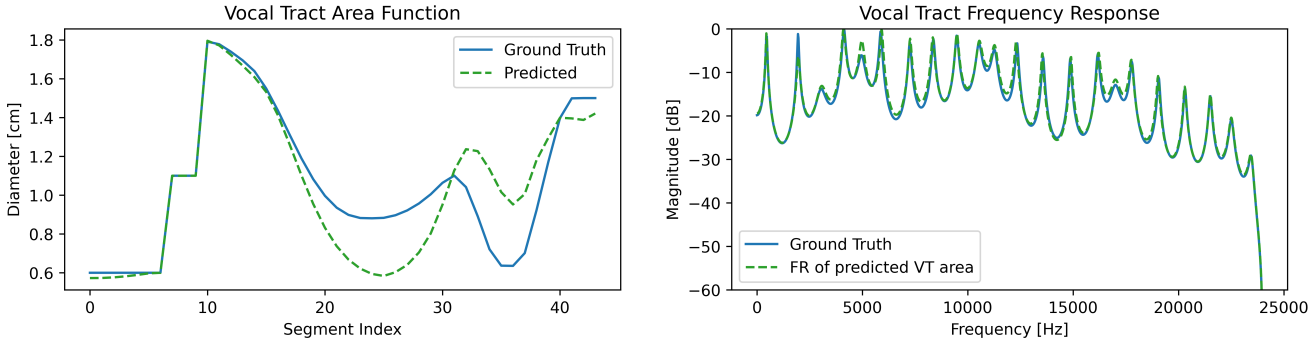


Figure 7: *Visibly different area functions can have very similar frequency responses.*

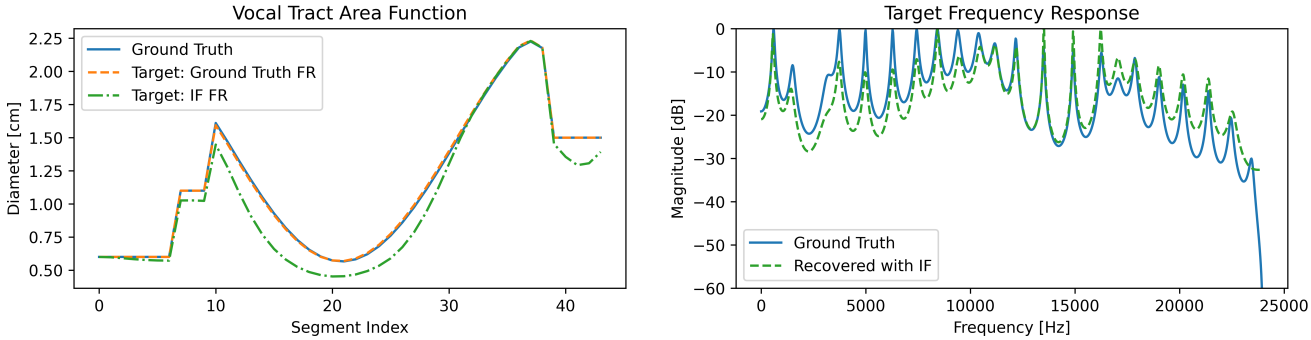


Figure 8: *Area estimation results when either the frequency response (FR) of the true vocal tract or the result of inverse filtering (IF) are used as the target. The two different target frequency responses are shown on the right.*

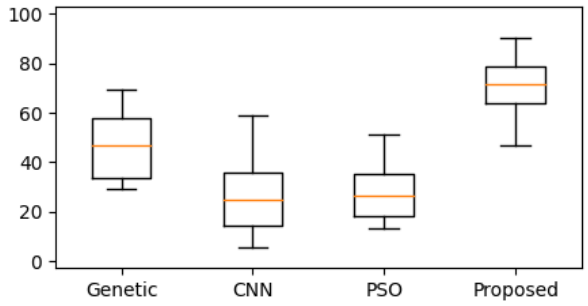


Figure 9: *Boxplots showing the average rating across all stimuli of our gradient-based approach and black-box baselines.*

a different male speaker, since the PT's fixed vocal tract length limits its output to voices that are read as male (see section 2.3.2). We set up an online multiple-stimulus test on the Go Listen platform [37] asking participants to compare the four reproductions to the original recording and rate the reproduction on a scale of 0–100. We included an additional screening question in which we replaced one of the reproductions with the original recording to ensure participants had understood the instructions and were in a suitable listening environment.

22 participants took part in the listening test. Of those, 4 gave the original recording in the screening question a rating lower than 80, so their results were discarded.

The results of the listening test are shown in Figure 9. Friedman's rank sum test indicates that the ratings differ significantly ($p < 0.001$), and post-hoc analysis using Wilcoxon's signed-rank test confirms that the reproductions obtained by our proposed approach are rated significantly ($p < 0.001$) higher than the three baselines, indicating that our method is well-suited for the sound matching task.

4. CONCLUSION

We presented a white-box optimization technique for sound matching vowel sounds with the articulatory synthesizer. We obtained a vocal tract frequency response through inverse filtering and estimated corresponding articulatory control parameters with gradient descent optimization, propagating error gradients through the mapping of control parameters to the vocal tract area function.

We showed that our approach can accurately match frequency responses for audio generated by the synthesizer itself. Reproductions of time-varying human vocalizations generated with our approach outperformed black-box baselines in a subjective evaluation.

By showing that articulatory features can be estimated with a gradient-based method, our work lays the foundation for further research into end-to-end sound matching of articulatory synthesizers using neural networks, which require the propagation of gradients. Additionally, our method can be expanded to explore the sound matching of more complex synthesizers, including those with two- and three-dimensional vocal tract models and varying vocal tract lengths that are not limited to adult male voices.

5. ACKNOWLEDGMENTS

This work was supported by UK Research and Innovation [grant number EP/S022694/1]. The authors would like to thank Benjamin Hayes, Yisu Zong, Christian Steinmetz and Marco Comunità for valuable feedback.

6. REFERENCES

[1] P. Birkholz, "Modeling Consonant-Vowel Coarticulation for Articulatory Speech Synthesis," *PLoS ONE*, vol. 8, Apr. 2013.

[2] B. J. Kröger, "Computer-Implemented Articulatory Models for Speech Production: A Review," *Frontiers in Robotics and AI*, vol. 9, 2022.

[3] B. H. Story, I. R. Titze, and E. A. Hoffman, "Vocal tract area functions from magnetic resonance imaging," *The Journal of the Acoustical Society of America*, vol. 100, pp. 537–554, July 1996.

[4] A. Toutios and S. Narayanan, "Articulatory synthesis of French connected speech from EMA data," in *Interspeech*, pp. 2738–2742, Aug. 2013.

[5] J. Dang and K. Honda, "Estimation of vocal tract shapes from speech sounds with a physiological articulatory model," *Journal of Phonetics*, vol. 30, pp. 511–532, July 2002.

[6] P. Liu, Q. Yu, Z. Wu, S. Kang, H. Meng, and L. Cai, "A deep recurrent approach for acoustic-to-articulatory inversion," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4450–4454, Apr. 2015.

[7] B. S. Atal, J. J. Chang, M. V. Mathews, and J. W. Tukey, "Inversion of articulatory-to-acoustic transformation in the vocal tract by a computer-sorting technique," *The Journal of the Acoustical Society of America*, vol. 63, pp. 1535–1555, May 1978.

[8] V. N. Sorokin, A. S. Leonov, and A. V. Trushkin, "Estimation of stability and accuracy of inverse problem solution for the vocal tract," *Speech Communication*, vol. 30, pp. 55–74, Jan. 2000.

[9] K. Richmond, *Estimating Articulatory Parameters from the Acoustic Speech Signal*. PhD thesis, University of Edinburgh, 2001.

[10] J. Riionheimo and V. Välimäki, "Parameter Estimation of a Plucked String Synthesis Model Using a Genetic Algorithm with Perceptual Fitness Calculation," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, Dec. 2003.

[11] C. Cooper, D. Murphy, D. Howard, and A. Tyrrell, "Singing synthesis with an evolved physical model," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 4, pp. 1454–1461, 2006.

[12] O. Schleusing, T. Kinnunen, B. Story, and J.-M. Vesin, "Joint Source-Filter Optimization for Accurate Vocal Tract Estimation Using Differential Evolution," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, pp. 1560–1572, Aug. 2013.

[13] Y. Gao, S. Stone, and P. Birkholz, "Articulatory Copy Synthesis Based on a Genetic Algorithm," in *Interspeech*, pp. 3770–3774, Sept. 2019.

[14] N. Masuda and D. Saito, "Quality Diversity for Synthesizer Sound Matching," in *24th International Conference on Digital Audio Effects (DAFx)*, Sept. 2021.

[15] M. A. Ismail, "Vocal Tract Area Function Estimation Using Particle Swarm," *Journal of Computers*, vol. 3, pp. 32–38, June 2008.

[16] L. Gabrielli, S. Tomassetti, S. Squartini, and C. Zinato, "Introducing deep machine learning for parameter estimation in physical modelling," in *20th International Conference on Digital Audio Effects (DAFx)*, Sept. 2017.

[17] M. J. Yee-King, L. Fedden, and M. d'Inverno, "Automatic Programming of VST Sound Synthesizers Using Deep Networks and Other Techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, 2018.

[18] P. Saha and S. Fels, "Learning joint articulatory-acoustic representations with normalizing flows," in *Interspeech*, pp. 3196–3200, 2020.

[19] H. Shibata, M. Zhang, and T. Shinozaki, "Unsupervised Acoustic-to-Articulatory Inversion Neural Network Learning Based on Deterministic Policy Gradient," in *2021 IEEE Spoken Language Technology Workshop*, (Shenzhen, China), pp. 530–537, Jan. 2021.

[20] M. A. Martínez Ramírez, O. Wang, P. Smaragdis, and N. J. Bryan, "Differentiable Signal Processing With Black-Box Audio Effects," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 66–70, June 2021.

[21] V. Chatziioannou and M. van Walstijn, "Estimation of Clarinet Reed Parameters by Inverse Modelling," *Acta Acustica*, vol. 98, pp. 629–639, July 2012.

[22] W. J. Wilkinson, J. D. Reiss, and D. Stowell, "Latent force models for sound: Learning modal synthesis parameters and excitation functions from audio recordings," in *20th International Conference on Digital Audio Effects (DAFx)*, pp. 56–63, 2017.

[23] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *International Conference on Learning Representations*, 2020.

[24] J. T. Colonel, C. J. Steinmetz, M. Michelen, and J. D. Reiss, "Direct design of biquad filter cascades with deep learning by sampling random polynomials," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Feb. 2022.

[25] F. Caspe, A. McPherson, and M. Sandler, "DDX7: Differentiable FM Synthesis of Musical Instrument Sounds," in *23rd International Society for Music Information Retrieval Conference*, 2022.

[26] R. Diaz, B. Hayes, C. Saitis, G. Fazekas, and M. Sandler, "Rigid-Body Sound Synthesis with Differentiable Modal Resonators." `http://arxiv.org/abs/2210.15306`, Oct. 2022.

[27] T. Smyth and D. Zurale, "On The Transfer Function of the Piecewise-Cylindrical Vocal Tract Model," in *18th Sound and Music Computing Conference*, 2021.

[28] O. Perrotin and I. McLoughlin, "A Spectral Glottal Flow Model for Source-filter Separation of Speech," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019.

[29] G. Fant, J. Liljencrants, and Q.-G. Lin, "A four-parameter model of glottal flow," *STL-QPSR*, vol. 26, no. 4, 1985.

[30] H.-L. Lu and J. O. Smith, "Glottal source modeling for singing voice synthesis," in *International Computer Music Conference*, 2000.

[31] G. Fant, "The LF-model revisited. Transformations and frequency domain analysis," *STL-QPSR*, vol. 2, no. 3, 1995.

[32] A. de Cheveigné and H. Kawahara, "YIN, a Fundamental Frequency Estimator for Speech and Music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[33] B. H. Story, "A parametric model of the vocal tract area function for vowel and consonant simulation," *The Journal of the Acoustical Society of America*, vol. 117, no. 5, pp. 3231–3254, 2005.

[34] J. L. Kelly and C. C. Lochbaum, "Speech Synthesis," in *Stockholm Speech Communication Seminar*, 1962.

[35] V. Välimäki and M. Karjalainen, "Improving the Kelly-Lochbaum Vocal Tract Model using Conical Tube Sections and Fractional Delay Filtering Techniques.," in *International Conference on Spoken Language Processing (ICSLP)*, 1994.

[36] Abraham. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures.," *Analytical Chemistry*, vol. 36, pp. 1627–1639, July 1964.

[37] D. Barry, Q. Zhang, P. W. Sun, and A. Hines, "Go Listen: An End-to-End Online Listening Test Platform," *Journal of Open Research Software*, vol. 9, July 2021.

Oral Session 3:
Virtual Analog & Hardware

POWER-BALANCED DYNAMIC MODELING OF VACTROLS: APPLICATION TO A VTL5C3/2

Judy Najnudel and Rémy Müller

UVI
Paris, France
j.najnudel@uvi.net | r.muller@uvi.net

Thomas Hélie and David Roze

IRCAM-CNRS-SU
Paris, France
thomas.helie@ircam.fr | david.roze@ircam.fr

ABSTRACT

Vactrols, which consist of a photoresistor and a light-emitting element that are optically coupled, are key components in optical dynamic compressors. Indeed, the photoresistor's program-dependent dynamic characteristics make it advantageous for automatic gain control in audio applications. Vactrols are becoming more and more difficult to find, while the interest for optical compression in the audio community does not diminish. They are thus good candidates for virtual analog modeling. In this paper, a model of vactrols that is entirely physical, passive, with a program-dependent dynamic behavior, is proposed. The model is based on first principles that govern semi-conductors, as well as the port-Hamiltonian systems formalism, which allows the modeling of nonlinear, multiphysical behaviors. The proposed model is identified with a real vactrol, then connected to other components in order to simulate a simple optical compressor.

1. INTRODUCTION

A vactrol (or resistive opto-isolator) consists of a photoresistor (labelled LDR for Light-Dependent Resistor in the following) and a light-emitting element (usually a LED) that are optically coupled: the photoresistor's resistance decreases (nonlinearly) with the light it receives from the LED.

Vactrols were widely used from the 1960s to the early 2000s due to their low fabrication costs, important dynamic range, and low noise distortion (below - 80 dB). They could be found in cameras (in exposure meters and autofocus) and security systems (for object detection) to name a few applications.

Apart from their nonlinear response to light, a remarkable feature of photoresistors is their relatively long response times (compared to e.g. transistors in transistor opto-isolators). Indeed, these response times vary from a few tens of microseconds (for the *turn-on*, or attack, when light is switched on) to a few hundreds of milliseconds (for the *turn-off*, or release, when light is switched off). Moreover, the attack time decreases with the received light. These characteristics made vactrols much appreciated for automatic gain control in audio applications, in which adaptive treatment and transient preservation are essential. An emblematic example of optical dynamic compressors from the late 1960s is the LA-2A built by Teletronix [1], which was used in prominent broadcast studios such as CBS and RCA. A more recent example is the Langevin ELOP built by Manley [2].

Vactrols were manufactured by Perkin Elmer [3] among others, until 2010. They are still manufactured by Advanced Photonix, but their availability has severely diminished since the 2000s due to a EU ban on cadmium sulfide, which is one of the main components of photoresistors. As vintage optical dynamic compressors are priced at tens of thousands of dollars, an accurate simulation is a convenient and much cheaper way to access optical dynamic compression.

Models of vactrols for audio applications have been proposed in the literature [4, 5]. Based on a signal representation, they associate static characteristics obtained from measurements, and a combination of low-pass filters to account for the photoresistor's dynamic behavior. More recent models based on Recurrent Neural Networks [6, 7] allow the joint inference of static and dynamic characteristics from data.

Although these models demonstrate interesting features from both qualitative and computational points of view, they are tailored to a specific circuit; therefore, they offer much less modularity than purely physical models. In particular, they are difficult to connect to other components modeled as port-Hamiltonian systems, which provide a unified formalism for the modeling of multiphysical systems with passivity guarantees, and has proven relevant for audio applications [8, 9]. Moreover, a key specificity of vactrols, namely, the inherent *program-dependence* of the attack and release times, remains elusive in signal-based models.

In this paper, which is a condensed version of the work presented in [19], we propose a nonlinear model of vactrols that is entirely physical, passive, with a program-dependent dynamic behavior by construction. To this end, we rely on first principles that govern semi-conductors, and the port-Hamiltonian systems (PHS) formalism.

This paper is structured as follows. In Section 2, we give a brief reminder on PHS. In Section 3, we recall the main doping mechanisms in photoresistors and derive a model for their internal dynamics. Then in Section 4, we propose a law for the optical coupling between the LED and the photoresistor, and obtain a complete PHS model for the vactrol by connecting all subcomponents through multiphysical ports. Finally, the model's parameters are estimated from measurements of a real vactrol, and then used to simulate a simplified optical compressor in Section 5.

2. REMINDER ON PORT-HAMILTONIAN SYSTEMS

Any physical system can be divided into parts that interact with each other via energy exchanges. Detailed presentations of PHS are available in [10, 11, 12]. In this paper, we rely on a differential-algebraic formulation adapted to multiphysical systems [13, 14]. This formulation allows the representation of a dynamical system as a network of

1. storage components of state $\boldsymbol{x}$ and energy $E(\boldsymbol{x})$;
2. passive memoryless components described by an effort law $z : \boldsymbol{w} \mapsto z(\boldsymbol{w})$, such that the dissipated power $P_{\text{diss}} = z(\boldsymbol{w})^\intercal \boldsymbol{w}$ is non-negative for all flows $\boldsymbol{w}$;
3. connection ports conveying the *outgoing* power $P_{\text{ext}} = \boldsymbol{u}^\intercal \boldsymbol{y}$ where $\boldsymbol{u}$ are inputs and $\boldsymbol{y}$ are outputs.

The system flows $\boldsymbol{f}$ and efforts $\boldsymbol{e}$ are coupled through a (possibly dependent on $\boldsymbol{x}$) skew-symmetric interconnection matrix $\boldsymbol{S} = -\boldsymbol{S}^\intercal$, so that

$$\underbrace{\begin{bmatrix}\dot{\boldsymbol{x}}\\ \boldsymbol{w}\\ \boldsymbol{y}\end{bmatrix}}_{\boldsymbol{f}} = \boldsymbol{S} \underbrace{\begin{bmatrix}\nabla E(\boldsymbol{x})\\ z(\boldsymbol{w})\\ \boldsymbol{u}\end{bmatrix}}_{\boldsymbol{e}}. \tag{1}$$

In the context of electronic circuits, flows can either be currents (e.g. for capacitors) or voltages (e.g. for inductors), and vice versa for efforts.

Such systems satisfy the power balance

$$P_{\text{stored}} + P_{\text{diss}} + P_{\text{ext}} = 0 \tag{2}$$

where $P_{\text{stored}} = \nabla E(\boldsymbol{x})^\intercal \dot{\boldsymbol{x}}$ denotes the stored power.

Proof By skew-symmetry of $\boldsymbol{S}$ (which we recall diagonal is zero),

$$P_{\text{stored}} + P_{\text{diss}} + P_{\text{ext}} = \boldsymbol{e}^\intercal \boldsymbol{f} = \boldsymbol{e}^\intercal \boldsymbol{S} \boldsymbol{e} = 0. \quad \square$$

Note that throughout this paper, we adopt the *receiver convention* for all components, including external sources. This means that the current is defined positive when entering the component through the positive voltage terminal [15].

3. PHOTORESISTOR

A photoresistor consists of a thin layer of photoconductive material (typically, cadmium sulfide) deposited on a ceramic substrate. In the following, we assume that the photoconductive material is spatially homogeneous, so that no pn-junction can be formed and that diffusion of free carriers is negligible. As a consequence, the photoresistor internal dynamics can be modeled by ODEs.

A photoresistor is in fact a 2-port component. The first port is electrical, and allows connections to other electronic components like any other resistor. The second port is optical, and allows interactions with light. It is responsible for internal dynamics in the micro-electrical domain. In the next sections, the (hidden) flow and effort that result from optical interactions are explicitly referred to as f_{opt} and e_{opt}, to avoid confusion with the (directly observable) electrical flow i_{LDR} (current), and electrical effort v_{LDR} (voltage). Note that both flows are in Amperes, and both efforts are in Volts.

3.1. Photoconductivity and doping

Photoconductive materials are semiconductors, that is, they become conductive under certain conditions. Indeed, the photoabsorption of a small amount of additional energy (denoted e_g) generates a pair of free carriers: an electron in the conduction band (of energy E_C), and a hole in the valence band (of energy E_V). The presence of free carriers increases the photoresistor conductivity. Note that the overall conductivity depends on the quantity of electrons, but also on the quantity of holes (which, as we shall see, do not necessarily remain equal).

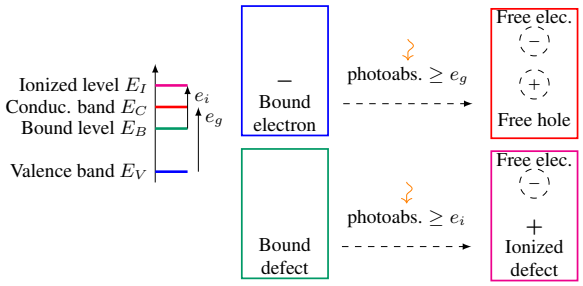


Figure 1: *Energy levels and free carriers before and after photoabsorption for an n-doped semi-conductor.*

The conductivity of a photoconductive material can be artificially increased (or "doped") through the inclusion of defects, which lower the amount of energy needed to generate free carriers via ionization [16]. Here, we consider n-doped materials only, as cadmium sulfide (which is present in Vactrols) is most generally n-doped [17]. For efficient n-doping, the bound defect energy (denoted E_B) must lie between the valence band and the conduction band, and the ionized defect energy (denoted E_I) must lie in the conduction band, so that the ionization energy e_i is smaller than e_g and electrons reach the conduction band more easily (Fig. 1).

3.2. Internal dynamics

The internal dynamics of the photoresistor is due to carrier recombination processes. The most important recombination process in doped materials is the Shockley-Read-Hall recombination [16], in which free carriers recombine with defects. Therefore, other kinds of recombination (e.g., radiative and Auger) are neglected.

The Shockley-Read-Hall recombination can be summed up as follows (Fig. 2). Assume an initial state with no free carriers, denoted s_0. Photoabsorption can lead either to ionization (state s_i), or to electron-hole pair generation (state s_g). The ionized defect can then return to state s_0, followed by the dissipation of excess energy e_i. In this case, the ionized defect acts as a "trap" for electrons. Likewise, an electron-hole pair can recombine to state s_0, followed by the dissipation of excess energy e_g.

For convenience, and since this is transparent energy-wise, we choose to model a unique source of photoabsorption (that leads to electron-hole pair generation), and replace a transition from state s_g to state s_0, followed by a transition from state s_0 to state s_i, with a direct transition from state s_g to state s_i, that dissipates the energy $e_g - e_i$. In this case, the bounded defect acts as a "trap" for holes.

Free carriers dynamics modeling To model the free carriers dynamics in the micro-electrical domain, we rely on the homogeneous Iverson model [18]. Denote f_{opt} the optical flow responsible for the generation of free carriers (in receiver convention), q^- the absolute charge of electrons and $\dot{q}^-$ its time variation, q^+ the absolute charge of holes and $\dot{q}^+$ its time variation, q_τ^+ the absolute charge of defects in ionized state, q_τ^0 the absolute charge of defects in bound state[1], and q_τ the absolute charge of defects. The photoresistor internal dynamics are governed by the following

[1] Here, the absolute charge of a species is to be understood as the number of instances multiplied with the elementary charge for homogeneity. Therefore, it can be non-zero even for bound defects.

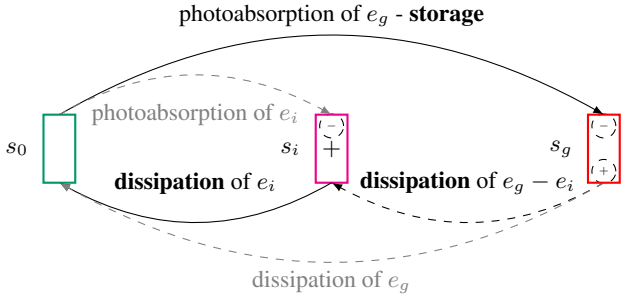


Figure 2: *Possible states and transitions for a n-doped photoresistor with Shockley-Read-Hall recombination. For convenience, a transition from state s_g to state s_0 followed by a transition from state s_0 to state s_i (gray, dashed), is replaced by a transition from state s_g to state s_i (black, dashed).*

equations [16]:

$$\text{Electron charge variation} \qquad \dot{q}^- = -f_{\text{opt}} - \nu_0^- \, q_\tau^+ \, q^-, \tag{3a}$$

$$\text{Hole charge variation} \qquad \dot{q}^+ = -f_{\text{opt}} - \nu_0^+ \, q_\tau^0 \, q^+, \tag{3b}$$

$$\text{Constant quantity of defects} \qquad q_\tau = q_\tau^+ + q_\tau^0, \quad \dot{q}_\tau = 0, \tag{3c}$$

$$\text{Charge neutrality} \qquad q_\tau^+ + q^+ - q^- = 0, \tag{3d}$$

where the electron (resp. hole) absolute charge time variation corresponds to the free carriers generation rate minus the de-ionization (resp. ionization) rate. The constants ν_0^- and ν_0^+ (in $\text{C}^{-1}{\cdot}\text{s}^{-1}$) relate to the electron and hole velocity, respectively. The constraints expressed by Eqs. (3c)-(3d) allow the reduction of Eqs. (3a)-(3b) to

$$\dot{q}^- = -f_{\text{opt}} - \nu_0^- \left(q^- - q^+\right) q^-, \tag{4a}$$

$$\dot{q}^+ = -f_{\text{opt}} - \nu_0^+ \left(q_\tau + q^+ - q^-\right) q^+. \tag{4b}$$

PHS formulation and equivalent circuit The system described by Eq. (4) admits a PHS formulation. To give this system a physical interpretation in terms of currents and voltages, we introduce capacities C^+ and C^-, as well as conductances $G_\tau^-(q^+, q^-) := \nu_0^- \, C^- \left(q^- - q^+\right)$ and $G_\tau^+(q^+, q^-) := \nu_0^+ \, C^+ \left(q_\tau + q^+ - q^-\right)$. The internal dynamics of the photoresistor can thus be described by two equivalent capacitors C^- and C^+ that model the storage of electrons and holes respectively, with flow $\dot{\boldsymbol{x}}$ (in Amperes) and effort $\nabla E(\boldsymbol{x})$ (in Volts)

$$\begin{aligned} \dot{\boldsymbol{x}} &= \left[\dot{q}^+, \dot{q}^-\right]^\intercal =: \left[f_{\text{C}+}, f_{\text{C}-}\right]^\intercal, \\ \nabla E(\boldsymbol{x}) &= \left[\frac{q^+}{C^+}, \frac{q^-}{C^-}\right]^\intercal =: \left[e_{\text{C}+}, e_{\text{C}-}\right]^\intercal, \end{aligned} \tag{5}$$

and two conductors G_τ^- and G_τ^+ that model the dissipation caused by de-ionization and ionization respectively, with flow $\boldsymbol{w}$ (in Volts) and (state-modulated) effort $z(\boldsymbol{w})$ (in Amperes)

$$\boldsymbol{w} = \left[e_{G_\tau^+}, e_{G_\tau^-}\right]^\intercal, \quad z(\boldsymbol{w}) = \left[G_\tau^+ \, e_{G_\tau^+}, G_\tau^- \, e_{G_\tau^-}\right]^\intercal =: \left[f_{G_\tau^+}, f_{G_\tau^-}\right]^\intercal. \tag{6}$$

The schematics of the corresponding equivalent circuit is shown on Fig. 3a, with corresponding PHS equations in Fig. 3b.

Remark 1. *The dynamics described by Eq. (4) depends on $C^\pm$ through the reduced parameters $\nu_0^\pm = \nu_0^\pm \, C^\pm \cdot \frac{1}{C^\pm}$. As only $\nu_0^\pm$ have to be identified and $\nu_0^\pm \, C^\pm$ and $C^\pm$ are unknown, we arbitrarily set $C^+ = C^- = 1$ F.*

Photoresistor electrical flow and effort The resistance of the photoresistor is given by [16]

$$R(q^+, q^-) = \frac{1}{\mu_0^+ \, q^+ + \mu_0^- \, q^-}, \tag{7}$$

where μ_0^+ (resp. μ_0^-) is the surface mobility (in $\text{V}^{-1}{\cdot}\text{s}^{-1}$) of holes (resp. electrons) and relates to the dimensions of the photoresistor and the mobility of holes.

In practice, the photoresistor exhibits a large but finite positive resistance R_d when it is not exposed to light (d for *dark*), and a small but non-zero positive resistance R_ℓ when it is exposed to maximal light (ℓ for *light*). The total resistance $R_{\text{LDR}}(q^+, q^-)$ is thus modeled as a parallel/series interconnection governed by

$$R_{\text{LDR}}(q^+, q^-) = \frac{R_\text{d}\left(R(q^+, q^-) + R_\ell\right)}{R_\text{d} + R(q^+, q^-) + R_\ell}. \tag{8}$$

As R_d is several orders of magnitude larger than R_ℓ, it is immediately verified that $\lim_{R \to +\infty} R_{\text{LDR}} = R_\text{d}$ and $\lim_{R \to 0} R_{\text{LDR}} = R_\ell$. We deduce the electrical flow $\boldsymbol{w}$ and (state-modulated) effort $z(\boldsymbol{w})$ of the photoresistor:

$$\boldsymbol{w} = i_{\text{LDR}}, \quad z(\boldsymbol{w}) = R_{\text{LDR}}\left(q^+, q^-\right) i_{\text{LDR}} =: v_{\text{LDR}}. \tag{9}$$

4. OPTICAL COUPLING

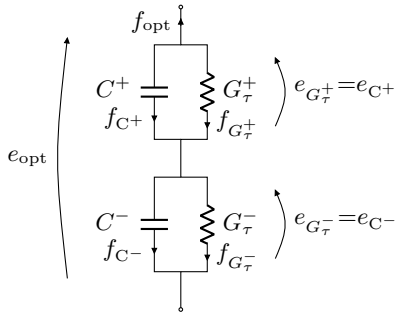
4.1. Power conversion between the LED and the photoresistor

Two types of power conversion take place between the LED and the photoresistor. The first one is a conversion from electrical to optical and occurs during photoemission by the LED. The second one is a conversion from optical to micro-electrical and occurs during photoabsorption by the photoresistor.

However, as the photoresistor spectral sensitivity does not match exactly with the LED spectral output, some of the power emitted by the LED is not transmitted to the photoresistor. Since the LED and photoresistor are enclosed in opaque resin, we assume that the difference of power is absorbed by the resin and converted into heat. Therefore, for convenience, the conversion from electrical to optical to micro-electrical domain is modeled as a dissipative quadripole (Fig.4). Denote v_D the LED voltage, i_D the LED current, and $P_\text{D} := i_\text{D} \, v_\text{D}$ the LED electrical power. Assume that the LED dissipative law $\mathcal{I} : v_\text{D} \mapsto i_\text{D}$ is known (e.g. from measurements or datasheets). Denote $P_{\text{opt}} := f_{\text{opt}} \, e_{\text{opt}}$ the optical power outgoing from the photoresistor. The power conversion must be passive, that is, the incoming optical power cannot be greater than the power delivered by the LED. Therefore, we choose to model the power conversion with a function f such that

$$f(P_\text{D}) + P_{\text{opt}} = 0, \quad \text{with } 0 \leq f(P_\text{D}) \leq P_\text{D}. \tag{10}$$

Assuming that $e_{\text{opt}} \neq 0$, the quadripole flow $\boldsymbol{w}$ and effort $z(\boldsymbol{w})$



(a) *Photoresistor internal dynamics equivalent circuit schematics.*

	e_{C^+}	e_{C^-}	$f_{G_\tau^+}$	$f_{G_\tau^-}$	f_{opt}
f_{C^+}	.	.	-1	.	-1
f_{C^-}	.	.	.	-1	-1
$e_{G_\tau^+}$	1	.	.	.	.
$e_{G_\tau^-}$	.	1	.	.	.
e_{opt}	1	1	.	.	.

(b) *Corresponding PHS. Dots represent zeros.*

Figure 3: *Photoresistor internal dynamics equivalent circuit schematics and corresponding PHS.*

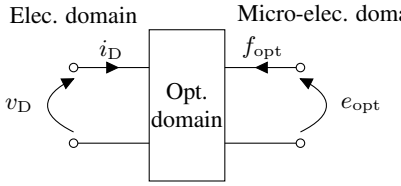


Figure 4: *Multiphysical power conversion between the LED and the photoresistor.*

can then be defined as

$$\boldsymbol{w} = [v_{\text{D}},\, e_{\text{opt}}]^\intercal\,, \quad z(\boldsymbol{w}) = \boldsymbol{\Gamma}(\boldsymbol{w})\,\boldsymbol{w} =: [i_{\text{D}},\, f_{\text{opt}}]^\intercal\,,$$

$$\text{with } \boldsymbol{\Gamma}(\boldsymbol{w}) := \underbrace{\begin{bmatrix} 0 & \frac{\mathcal{I}(v_{\text{D}})}{e_{\text{opt}}} \\ -\frac{\mathcal{I}(v_{\text{D}})}{e_{\text{opt}}} & 0 \end{bmatrix}}_{\boldsymbol{\Gamma}_{\text{antisym}}(\boldsymbol{w})} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & \frac{\mathcal{I}(v_{\text{D}})\,v_{\text{D}} - f\left(\mathcal{I}(v_{\text{D}})\,v_{\text{D}}\right)}{e_{\text{opt}}^2} \end{bmatrix}}_{\boldsymbol{\Gamma}_{\text{sym}}(\boldsymbol{w})}. \tag{11}$$

The matrix $\boldsymbol{\Gamma}_{\text{antisym}}(\boldsymbol{w})$ encodes the (modulated) gyration from electrical to micro-electrical domain, while the (non-negative) matrix $\boldsymbol{\Gamma}_{\text{sym}}(\boldsymbol{w})$ encodes the actual power loss during the conversion. With Eq. (10), such a formulation guarantees the passivity of the quadripole, since it verifies $z(\boldsymbol{w})^\intercal \boldsymbol{w} \geq 0$ for all $\boldsymbol{w}$.

In practice, we have neither access to photoemission (from the LED), nor photoabsorption (into the photoresistor) characteristics. However, "ground truth" for the optical power P_{opt} can be obtained from measurements of the photoresistor's resistance, as will be shown in Section 5. Regarding the function f, we propose an empirical model of the form

$$f(P_{\text{D}}) = P_0\, P_{\text{D}}^{\alpha_0} + P_1\, P_{\text{D}}^{\alpha_1}, \tag{12}$$

with α_0, α_1 positive, to account for the dual-slope of the optical power as a function of the LED power observed in the logarithmic domain.

4.2. Vactrol complete model

By connecting the power converter between the LED and the photoresistor with the equivalent circuit that models the internal dynamics of the photoresistor, the vactrol (Fig. 5a) can be modeled as the equivalent circuit in Fig. 5b. The corresponding PHS equations are given in Fig. 5c.

5. IDENTIFICATION OF A VACTROL FROM MEASUREMENTS AND SIMULATION OF A SIMPLE OPTICAL COMPRESSOR

The characteristics of a VTL5C3/2 are measured to obtain the LED dissipative law, the photoresistor static resistance as a function of LED power, as well as the photoresistor's turn-on and turn-off (see [19] for a schematic of the experimental setup).

The parameters R_ℓ and R_{d} are taken from the VTL5C3/2 datasheet, and the parameters μ_0^+ and μ_0^- are set according to [20]. The remaining vactrol parameters are then estimated from measurements, in three steps.

5.1. Parameter identification for the LED

The Shockley model [21] does not capture the behavior of LEDs that are present in Vactrols accurately. Indeed, once it has reached its threshold, the LED behaves more closely as a linear resistance. Therefore, we propose a more suitable model, passive by construction, in which the LED threshold appears explicitly as a parameter. Denote v_{D} the LED voltage and i_{D} the LED current. The LED is a dissipative component of flow $\boldsymbol{w}$ and effort $z(\boldsymbol{w})$ given by

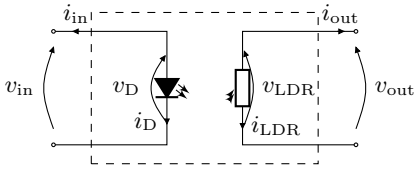
$$\boldsymbol{w} = v_{\text{D}},$$
$$z(\boldsymbol{w}) = I_s \left(\text{sp}\left(\frac{v_{\text{D}} - V_t}{V_s}\right) - \text{sp}\left(-\frac{V_t}{V_s}\right)\right) =: \mathcal{I}(v_{\text{D}}) = i_{\text{D}}, \tag{13}$$

where $\text{sp} : \; x \mapsto \ln(1 + \exp x)$ denotes the softplus function, and V_t, V_s, and I_s are positive model parameters. The voltage V_t is the LED threshold (from which it starts emitting light), and the ratio I_s/V_s corresponds to the LED conductance. Due to the fact that the sp function is increasing, and that $z(0) = 0$, this model is passive since $z(\boldsymbol{w})^\intercal \boldsymbol{w} \geq 0$ for all $\boldsymbol{w}$.

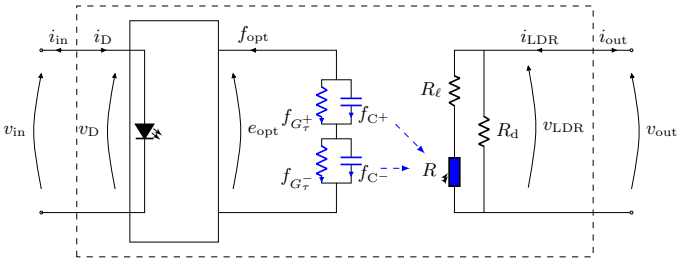
The set of parameters $\theta_{\text{D}} = [V_t,\, V_s,\, I_s]$ is estimated through a least-squares optimization. The estimated parameters are shown in Table 1. Figure 6 shows the results of the estimation compared with measurements.

5.2. Parameter identification for the photoresistor internal dynamics

To decouple the identification process, the set of parameters $\theta_{\text{LDR}} = \left[\nu_0^+,\, \nu_0^-,\, q_\tau\right]$ is estimated in isolation during turn-off, which avoids



(a) *Voltage-controlled vactrol schematics.*



(b) *Voltage-controlled vactrol equivalent circuit schematics. The photoresistance R is modulated by the states of C^+ and C^-.*

(c) *Corresponding PHS. Dots represent zeros.*

$$
\begin{array}{cc|cccccccc|cc}
 & & \multicolumn{8}{c|}{\nabla E(\boldsymbol{x})} & \multicolumn{2}{c}{z(\boldsymbol{w})} & \multicolumn{2}{c}{\boldsymbol{u}} \\
 & & e_{C^+} & e_{C^-} & f_{G_\tau^+} & f_{G_\tau^-} & i_{\text{D}} & f_{\text{opt}} & v_{\text{LDR}} & v_{\text{in}} & i_{\text{out}} \\
\dot{\boldsymbol{x}} & f_{C^+} & . & . & -1 & . & . & -1 & . & . & . \\
 & f_{C^-} & . & . & . & -1 & . & -1 & . & . & . \\
\hline
\boldsymbol{w} & e_{G_\tau^+} & 1 & . & . & . & . & . & . & . & . \\
 & e_{G_\tau^-} & . & 1 & . & . & . & . & . & . & . \\
 & v_{\text{D}} & . & . & . & . & . & . & . & 1 & . \\
 & e_{\text{opt}} & 1 & 1 & . & . & . & . & . & . & . \\
 & i_{\text{LDR}} & . & . & . & . & . & . & . & . & -1 \\
\hline
\boldsymbol{y} & i_{\text{in}} & . & . & . & . & -1 & . & . & . & . \\
 & v_{\text{out}} & . & . & . & . & . & . & 1 & . & .
\end{array}
$$

Figure 5: *Voltage-controlled vactrol schematics and corresponding PHS.*

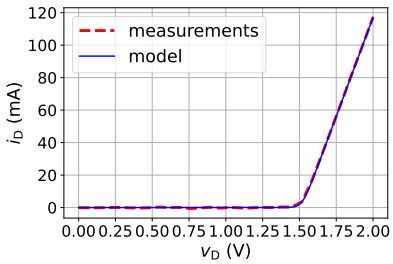


Figure 6: *LED dissipative law measurements and model.*

dependencies to the optical flow f_{opt}. Denote $R_{\text{sim}}(\theta)$ the photoresistor resistance simulated for a given set θ, using Eq. (4) with $f_{\text{opt}} = 0$ and Eq. (8). The set θ_{LDR} is obtained by fitting $R_{\text{sim}}(\theta)$ to the measured resistance through a least-squares optimization. The estimated parameters are shown in Table 1. Figure 7a shows that the resistance simulated with these parameters matches closely with the measured resistance.

5.3. Parameter identification for the power conversion law in steady state

The estimated set θ_{LDR} as well as the measured static resistance R_{LDR} as a function of LED power P_{D} can be used to provide experimental data for the optical power P_{opt}. The complete method is detailed in [19]. In this subsection, we recall the main steps but do not provide complete expressions and proofs, for brevity.

The first step is to reinject θ_{LDR} in Eq. (4) and solve for steady state $\dot{q}^+ = \dot{q}^- = 0$. This yields the relation

$$q_{st}^- = \mathcal{Q}^-(q_{st}^+), \tag{14}$$

where q_{st}^- and q_{st}^+ are the electron charge and hole charge in steady state, and $\mathcal{Q}^-$ is a function parametrized by θ_{LDR}. The second step is to substitute Eq. (14) in Eq. (8) and invert Eq. (8). We obtain the relation

$$q_{st}^+ = \mathcal{Q}^+(R_{\text{LDR}}), \tag{15}$$

where $\mathcal{Q}^+$ is a function parametrized by θ_{LDR}, μ_0^+, μ_0^-, R_{d} and R_ℓ. Finally, from Fig. 3b, we have in steady state (for the choice $C^\pm$ = 1 F, see Remark 1)

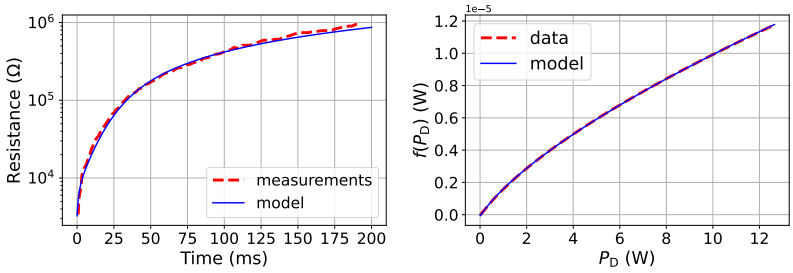
$$P_{\text{opt}} := f_{\text{opt}}\, e_{\text{opt}} = -\nu_0^- \left(q_{st}^- - q_{st}^+\right) q_{st}^- \left(q_{st}^+ + q_{st}^-\right). \tag{16}$$

By reinjecting Eqs. (14)-(15) in Eq. (16), we obtain the measured optical power P_{opt} as a function of static resistance R_{LDR}, which is itself a function of LED power P_{D}.

The set of parameters $\theta_{\text{f}} = [P_0,\ P_1,\ \alpha_0,\ \alpha_1]$ is then estimated by fitting Eq. (12) through a least-squares optimization. The estimated parameters are shown in Table 1, and Fig. 7b shows the estimation results. The simplified model matches well with the original model within the range of LED powers observed under normal use. Moreover, the passivity of the model expressed by Eq. (10) is indeed verified since $0 \leq f(P_{\text{D}}) \leq P_{\text{D}}$.

5.4. Model validation

To confirm the parameter estimation results, the photoresistor's resistance is simulated again with a non-zero optical flow, for several



(a) *Photoresistor's turn-off measurements and model.*

(b) *Photoresistor's absorbed optical power vs LED power.*

(c) *Photoresistor's turn-on measurements and model for P_D = 70 mW.*

(d) *Photoresistor's resistance vs LED power after 800 ms.*

Figure 7: *Photoresistor characteristics: turn-off (top left), power conversion (top right), turn-on (bottom left) and resistance vs LED power after 800 ms (bottom right).*

Table 1: *Parameters for the vactrol VTL5C3/2.*

	Parameter	Value	Unit
Given	μ_0^+	4	$\mathrm{V}^{-1}{\cdot}\mathrm{s}^{-1}$
	μ_0^-	35	$\mathrm{V}^{-1}{\cdot}\mathrm{s}^{-1}$
	R_ℓ	2	Ω
	R_d	10×10^6	Ω
Estimated	q_τ	9.77×10^{-1}	C
	ν_0+	1.35×10^2	$\mathrm{C}^{-1}{\cdot}\mathrm{s}^{-1}$
	ν_0^-	1.79×10^8	$\mathrm{C}^{-1}{\cdot}\mathrm{s}^{-1}$
	P_0	-5.47×10^{-5}	W
	P_1	5.63×10^{-5}	W
	α_0	0.54	dimensionless
	α_1	0.55	dimensionless
	V_t	1.52	V
	V_s	23.16×10^{-3}	V
	I_s	5.65×10^{-3}	A

values of LED powers. The turn-on time response matches with measurements (Fig. 7c, here for P_D = 70 mW). After 800 ms of simulation (the theoretical turn-on time being 3 ms), the simulated resistance matches very closely with the measured static resistance (Fig. 7d).

5.5. Simulation of an optical compressor

The estimated parameters for the VTL5C3/2 are used to simulate a minimal optical compressor, shown in Fig. 8a. This compressor consists in a voltage divider, in which the output resistor is the photoresistor. If the output voltage is greater than the LED threshold, the LED emits light, and the photoresistor resistance drops, decreasing the output voltage in a feedback control loop [24]. The operational amplifier-based voltage follower removes the electrical coupling between the photoresistor and the LED.

For this application, we treat the voltage follower power supplies as infinite constants, so that the operational amplifier never reaches saturation. Denote v^+ the input voltage, i^+ the input current, v_o the output voltage, and i_o the output current. The voltage follower has flow $\boldsymbol{w}$ and effort $z(\boldsymbol{w})$ given by (see [22] for a complete derivation)

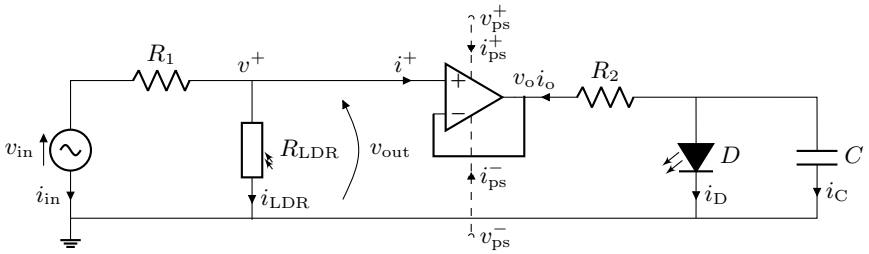
$$\boldsymbol{w} = \begin{bmatrix} v^+, \, i_\mathrm{o} \end{bmatrix}^\intercal, \quad z(\boldsymbol{w}) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \boldsymbol{w} =: \begin{bmatrix} i^+, \, v_\mathrm{o} \end{bmatrix}^\intercal. \tag{17}$$

Kirchhoff's laws yield the (reduced) PHS equations in Fig. 8b.

The compressor is driven with a sinusoidal voltage of the form $v_\mathrm{in} = U(t)\, \sin(2\pi\, f_0\, t)$, with $U(t)$ defined as

$$U(t) = \begin{cases} 1 & \text{if } t \leq t_0 \\ U_0 & \text{if } t_0 \leq t \leq 2\, t_0 \\ 1 & \text{if } 2\, t_0 \leq t. \end{cases} \tag{18}$$

Here, the simulation is computed with an iterative solver, namely a Newton-Raphson method [23]. Simulation results for vactrol parameters in Table 1 and simulation parameters in Table 2, are shown in Fig. 9a to Fig. 9c. We observe that the higher the input voltage, the shorter the attack and the longer the release, in agreement with the photoresistor time responses. In all cases, the attack is much sharper than the release. The compression ratio and knee can be controlled with the resistances R_1 and R_2: the higher

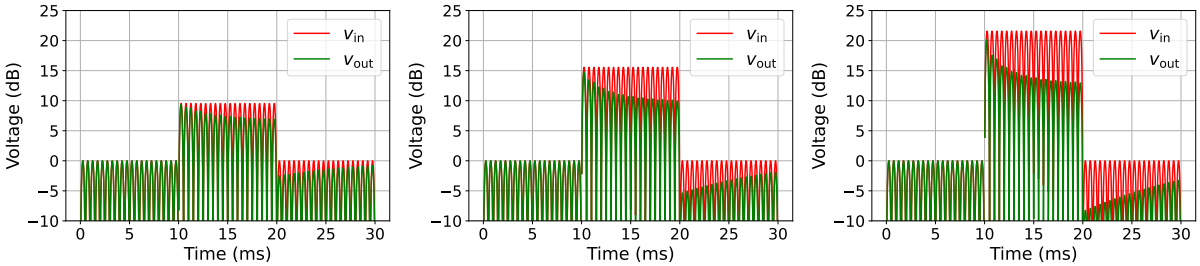


(a) *Schematics of a minimal optical compressor (OPA power supplies are dashed).*

		$\nabla E(\boldsymbol{x})$						$z(\boldsymbol{w})$				$\boldsymbol{u}$
		v_C	e_{C+}	e_{C-}	$f_{G_\tau^+}$	$f_{G_\tau^-}$	i_D	f_{opt}	i^+	v_o	v_{LDR}	v_{in}
$\dot{\boldsymbol{x}}$	i_C	$-1/R_2$	.	.	.	.	-1	.	.	$1/R_2$	.	.
	f_{C+}	.	.	.	-1	.	.	-1	.	.	.	.
	f_{C-}	.	.	.	.	-1	.	-1	.	.	.	.
$\boldsymbol{w}$	$e_{G_\tau^+}$	.	1	.	.	.	.	.	.	.	.	.
	$e_{G_\tau^-}$	.	.	1	.	.	.	.	.	.	.	.
	v_D	1	.	.	.	.	.	.	.	.	.	.
	e_{opt}	.	1	1	.	.	.	.	.	.	.	.
	v^+	.	.	.	.	.	.	.	.	.	1	.
	i_o	$1/R_2$	.	.	.	.	.	.	.	$-1/R_2$	.	.
	i_{LDR}	.	.	.	.	.	.	.	-1	.	$-1/R_1$	$1/R_1$
$\boldsymbol{y}$	i_{in}	.	.	.	.	.	.	.	.	.	$1/R_1$	$-1/R_1$

(b) *Corresponding PHS. Dots represent zeros.*

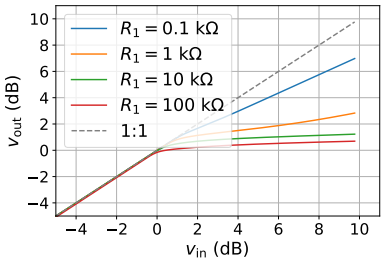
Figure 8: *Simple optical compressor schematics and corresponding PHS.*



(a) *Simulation with U_0 = 3 V.*

(b) *Simulation with U_0 = 6 V.*

(c) *Simulation with U_0 = 12 V.*



(d) *Compression ratio for different values of R_1 and fixed R_2.*

Figure 9: *Simple optical compressor simulation results.*

Table 2: *Optical compressor simulation parameters.*

Parameter	R_1 (kΩ)	R_2 (Ω)	C (μF)	U_0 (V)	f_0 (kHz)	t_0 (ms)	f_s (kHz)
Value	1	5	4.7	3-6-12	1	10	96

the ratio R_1/R_2, the higher the compression ratio and sharper the knee (Fig. 9d).

6. CONCLUSION

In this paper, we proposed a dynamic, multiphysical and power-balanced model for the vactrol, as a port-Hamiltonian system. First, we modeled the photoresistor. Its internal dynamics were obtained from the study of doping mechanisms in semiconductors. Then, we addressed the nonlinear optical coupling between the LED and the photoresistor. A law for this coupling was derived from the photoresistor's static resistance as a function of the LED's electrical power.

The model's parameters were successfully estimated from measurements of a real vactrol. Simulations using the estimated parameters closely match with measured dynamic and static characteristics. Finally, the model was implemented in order to simulate a minimal optical compressor. The simulated attack and release times are program-dependent, as expected in such compressors.

Regarding ongoing work, the model is being implemented in complex circuits that are closer to real optical compressors, and tested against real audio signals for a more complete assessment.

7. REFERENCES

[1] Lynn Fuston. A history of the Teletronix LA-2A leveling amplifier.

[2] Manley. `https://www.manley.com/pro/melpp`.

[3] *Photoconductive cells and analog optoisolators (Vactrols)*. Perkin Elmer Optoelectronics, 2001.

[4] Julian Parker and Stephano D'Angelo. A digital model of the Buchla lowpass-gate. In *Proc. Int. Conf. Digital Audio Effects (DAFx-13), Maynooth, Ireland*, pages 278–285, 2013.

[5] Felix Eichas and Udo Zölzer. Modeling of an optocoupler-based audio dynamic range control circuit. In *Novel Optical Systems Design and Optimization XIX*, volume 9948, page 99480W. International Society for Optics and Photonics, 2016.

[6] Alec Wright, Vesa Välimäki, et al. Grey-box modelling of dynamic range compression. In *Proc. Int. Conf. Digital Audio Effects (DAFx-22), Vienna, Austria*, pages 304–311, 2022.

[7] Riccardo Simionato and Stefano Fasciani. Deep learning conditioned modeling of optical compression. In *Proc. Int. Conf. Digital Audio Effects (DAFx-22), Vienna, Austria*, pages 288–295, 2022.

[8] Nicolas Lopes and Thomas Hélie. Energy balanced model of a jet interacting with a brass player's lip. *Acta Acustica united with Acustica*, 102(1):141–154, 2016.

[9] Antoine Falaize and Thomas Hélie. Passive simulation of the nonlinear port-Hamiltonian modeling of a Rhodes piano. *Journal of Sound and Vibration*, 390:289–309, 2017.

[10] B. M. Maschke, A. J. van der Schaft, and P. Breedveld. An intrinsic Hamiltonian formulation of network dynamics: Non-standard Poisson structures and gyrators. *Journal of the Franklin institute*, pages 923–966, 1992.

[11] Vincent Duindam, Alessandro Macchelli, Stefano Stramigioli, and Herman Bruyninckx. *Modeling and control of complex physical systems: The port-Hamiltonian approach*. Springer Science & Business Media, 2009.

[12] A. J. van der Schaft, Dimitri Jeltsema, et al. Port-Hamiltonian systems theory: An introductory overview. *Foundations and Trends in Systems and Control*, 1(2-3):173–378, 2014.

[13] Antoine Falaize and Thomas Hélie. Passive guaranteed simulation of analog audio circuits: A port-Hamiltonian approach. *Applied Sciences*, 6(10):273, 2016.

[14] Rémy Müller. *Time-continuous power-balanced simulation of nonlinear audio circuits: Realtime processing framework and aliasing rejection*. PhD thesis, Sorbonne Université, 2021.

[15] Timothy A. Bigelow. Power and energy in electric circuits. In *Electric Circuits, Systems, and Motors*, pages 105–121. Springer, 2020.

[16] Marius Grundmann. *Physics of semiconductors*, volume 11. Springer, 2010.

[17] U.V. Desnica. Doping limits in II–VI compounds—Challenges, problems and solutions. *Progress in crystal growth and characterization of materials*, 36(4):291–357, 1998.

[18] Arthur Evan Iverson. *The mathematical modeling of time-dependent photoconductive phenomena in semiconductors*. PhD thesis, The University of Arizona, 1987.

[19] Judy Najnudel. *Power-Balanced Modeling of Nonlinear Electronic Components and Circuits for Audio Effects*. PhD thesis, Sorbonne Université, 2022.

[20] WE Spear and J Mort. Electron and hole transport in CdS crystals. *Proceedings of the Physical Society (1958-1967)*, 81(1):130, 1963.

[21] William Shockley. The theory of p-n junctions in semiconductors and p-n junction transistors. *Bell System Technical Journal*, 28(3):435–489, 1949.

[22] Rémy Müller and Thomas Hélie. A minimal passive model of the operational amplifier: Application to Sallen-Key analog filters. In *Proc. of the 22nd Int. Conference on Digital Audio Effects*, 2019.

[23] Peter Deuflhard. *Newton methods for nonlinear problems: Affine invariance and adaptive algorithms*, volume 35. Springer Science & Business Media, 2011.

[24] Jonathan S Abel and David P Berners. On peak-detecting and rms feedback and feedforward compressors. In *Audio Engineering Society Convention 115*. Audio Engineering Society, 2003.

UPCYCLING ANDROID PHONES INTO EMBEDDED AUDIO PLATFORMS

Victor Zappi

Northeastern University
Boston, MA, US
v.zappi@northeastern.edu

Carla Sophie Tapparo

Independent Researcher
Buenos Aires, Argentina
sophiecarlatapparo@gmail.com

ABSTRACT

There are millions of sophisticated Android phones in the world that get disposed of at a very high rate due to consumerism. Their computational power and built-in features, instead of being wasted when discarded, could be repurposed for creative applications such as musical instruments and interactive audio installations. However, audio programming on Android is complicated and comes with restrictions that heavily impact performance. To address this issue, we present LDSP, an open-source environment that can be used to easily upcycle Android phones into embedded platforms optimized for audio synthesis and processing. We conducted a benchmark study to compare the number of oscillators that can be run in parallel on LDSP with an equivalent audio app designed according to modern Android standards. Our study tested six phones ranging from 2014 to 2018 and running different Android versions. The results consistently demonstrate that LDSP provides a significant boost in performance, with some cases showing an increase of more than double, making even very old phones suitable for fairly advanced audio applications.

1. INTRODUCTION

With its billions of users, Android is one of the most widely adopted technologies existing today [1, 2]. Even the more affordable Android phones have CPU and memory specifications that compare with or even top those of many platforms commonly used by academics, researchers and creatives to design audio applications, including the Raspberry Pi[1], Bela [3] and the Daisy Seed board[2]. Yet, the mobile phone market is characterized by a constant evolution of both software and hardware, with new updates and models released frequently. Although current Android phones boast impressive technical specifications, they are often abandoned by users due to software incompatibility or the desire to own a newer, more advanced device. This consumeristic approach to technology creates significant environmental and ethical issues.

Firstly, the regular replacement of mobile phones contributes to a throwaway culture that values disposability over sustainability [4]. Electronic waste (e-waste) generated by discarded phones is a growing concern as it contains hazardous substances such as lead, mercury and cadmium, which can pollute the environment and harm human health. Moreover, this consumeristic approach to technology—that spreads way beyond mobile phones—also perpetuates a cycle of social and economic inequality [5]. Not everyone can afford to upgrade their technologies regularly, and the constant release of new models creates an unnecessary pressure to keep up with the latest trends, contributing to a sense of inadequacy and exclusion among those who cannot afford to do so.

In contrast to these unsustainable trends, this paper describes LDSP, a technology that enables extending the lifespan of older Android phones by repurposing them as embedded platforms for audio application development. LDSP provides an environment that allows developers to leverage the full potential of the phones' hardware and avoid the limitations imposed by Android's runtime environment. With LDSP, phones that would have otherwise become obsolete can be given new life, decreasing the need for the purchase of new programmable audio technology and reducing e-waste. We discuss the implementation of LDSP, its capabilities and how it can provide a significant boost in performance for audio applications. Additionally, we present the results of our experiment using an oscillator bank to compare the performance of LDSP with that of a typical Android audio app, addressing the open problem of effectively employing Android phones for audio synthesis and processing.

2. BACKGROUND

2.1. Technology, E-waste and Upcycling

In the field of audio and music hardware design, there is a larger issue at play regarding the obsolescence and progress mindset surrounding technological products. This is in part enabled by manufacturers seeking to drive trends and increase consumption [6] and strongly resonates with humans' innate curiosity and will to experiment. This mindset is economically and environmentally harmful, given the decreased lifespan of equipment and the increased production of e-waste [7]. This issue has underlying epistemological roots, where technology-based or electronic arts are tied to the notion of progress, 'new is better', and consumerism, which is unsustainable [4]. The process of creating new technologies and their discard negatively affects the land, water, air, and all living beings.

Among consumer technologies, mobile phones are notorious for their short product cycles, with an average use time of around two years [8]. In many cases, phones are replaced with newer models even if they are still functioning, as a result of the expiration of support for essential apps or the operating system itself. In the case of Android phones, open software can be leveraged to extend the life span of the device beyond the end of support and continue its intended. Examples includes: Lineage OS [3], a mostly open-source operative system based on Android and maintained by a large community of developers; /e/ OS [4] and iodé OS

[1] https://www.raspberrypi.com/

[2] https://www.electro-smith.com/daisy

[3] https://lineageos.org/

[4] https://e.foundation/

[5], both open-source mobile operative systems forked from or powered by Lineage OS that focus on privacy; or the Free Software Foundation Europe's initiative Upcycling Android [6] that intends on teaching the possibilities given by free and open software to improve user agency against technological obsolescence. However, the applicability of these solutions tends to depend on the specific model of the phone and support for functionalities can only be extended for a finite period of time. For example, one of the two LG G2 Mini phones used in this work to test LDSP (see Section 4) is running Lineage OS 14.1 that is equivalent to Android 7.0, while the last official Android version that specific model can run is 5.0.2. This extended the overall life span of the phone of two years, from 2016 to 2018—i.e., when support for Android 5.0.2 and 7.0 officially ended, respectively. Furthermore, phones frequently encounter significant damage to their screens, batteries, or other crucial components that render them unusable for daily use, irrespective of software support. Repairing such devices is increasingly expensive and intricate due to the miniaturized design of the products, the presence of glued-in parts, and the overall concept of *economical obsolescence*, where the cost of repairs often outweighs the value of the device itself [8]. Nonetheless, in many cases damaged/broken phones can still be turned into different but functional devices.

There is a tradition of technological disobedience found worldwide through different approaches that involve recycling and reusing materials, from *hackerspaces* [9] to Gambiarra [10] and others [11, 12]. These practices recover components and devices that would otherwise be discarded, finding novel uses for them by surpassing limitations in innovative ways, which resist consumerism and planned obsolescence practically. Such practices shift our understanding of when an object becomes useless or expendable if it is not related to its functionality or lack thereof, forcing us to think critically about how we use, buy, and discard technology.

Various art-related projects and instruments are specifically aligned with the upcycling of materials and the political implications of such endeavors. For instance, the Echo project [13] explores creative and alternative uses of outdated and damaged technologies, fostering an atmosphere where audiences can engage in critical and aesthetic debates surrounding the possibilities of these technologies, away from their intended purposes. Similarly, the Gatorra instrument [14] was created through a hobbyist approach to circuitry, repurposing electronic and non-electronic components to create a unique final product, emphasizing the autonomy of the creator and promoting innovative ways of engaging with hardware.

Certain composers and musicians, including Yasunao Tone, Nicolas Collins, and the group Oval [15], use the glitching and skipping of compact disks to generate new sounds, chance-based compositions, and indeterminate performances. Though their approaches to technology may differ, they share an interest in using seemingly broken technology to encourage novel sounds.

Other instruments, such as the Concentric Sampler [16], repurpose outdated technology, like floppy disks and floppy disk drivers, with additional circuitry that loops and uses time-based granular synthesis for live performances of lo-fi noise. The author of this project discusses their motivation for fostering creativity through physical limitations and misuse of audio technologies. Similarly, Disky [17] is a D.I.Y. USB turntable that utilizes the mechanical parts found in obsolete hard disk drives, providing an accessible, reliable, and low-cost project for audio control. In both cases, the authors emphasize their motivations driven against the novelty-driven discard of technology due to its current way of production encouraged by consumerism. They see their upcycling methodology as a creative way of dealing with technology that is considered obsolete while fostering creativity and community.

[5] `https://iode.tech/`

[6] `https://fsfe.org/activities/upcyclingandroid/` [Accessed on 2023/05/26]

2.2. Android Audio Programming

There have been various efforts to turn Android phones into platforms for audio processing and synthesis, with applications like Nexus [18] and MoMubPlat [19] using web technologies and Pure Data/libpd, respectively. However, *faust2api* appears to be the most comprehensive project to date, offering optimized Faust audio/sensors processing code and graphical user interfaces designed to explore the acoustic features of handheld devices [20].

Despite their capabilities, all these platforms and environments are limited by the Android audio stack, which consists of several layers and buffers that can introduce significant computational overhead and latency in audio processing and synthesis applications. As a result, it is difficult to achieve the satisfactory audio performance, especially on outdated phones. This problem has been known to audio processing and synthesis communities for some time, as discussed in research such as [21].

These issues arise from the structure of the Android application framework that allows for hardware-agnostic development, even for code written in native languages like Faust or C++ using the Android NDK. Such code has to pass through multiple layers of the audio stack before it can exchange samples with the audio driver in kernel space. These layers include the application layer, Android's mixer and the audio HAL, each of which introduces some level of buffering and scheduling that can increase CPU workload and cause inconsistent latency. Figure 1 represents the typical audio stack in modern Android app architecture. Another crucial detail emphasized in this figure is that developing a high-performance Android app often requires combining at least two programming languages: Java for the overall app structure and C++ for the performance-critical components. Additionally, in most cases, a graphical user interface (UI) is necessary, which, in modern Android development, is typically implemented using yet another language, Kotlin.

Researchers have proposed various solutions to address this problem, including the technique described in [1], which leverages the Exclusive Mode of the AAudio API introduced by Google in 2017 to bypass many layers of the audio stack. However, this solution is available only on relatively modern devices (running Android 8 or above). And, in general, more work is needed to fully address these issues in Android audio processing and synthesis, particularly for interactive applications.

3. LDSP

LDSP is an open-source cross-platform environment designed to enable developers to create native C++ audio applications for Android devices. Unlike traditional Android apps that run within the Android Run-Time Environment/Dalvik Virtual Machine, LDSP generates executables that are dealt with directly by the kernel and can directly access memory and hardware resources. Essentially, LDSP turns Android phones into generic Linux embedded boards, with the only requirement being that the phone is *rooted*. This

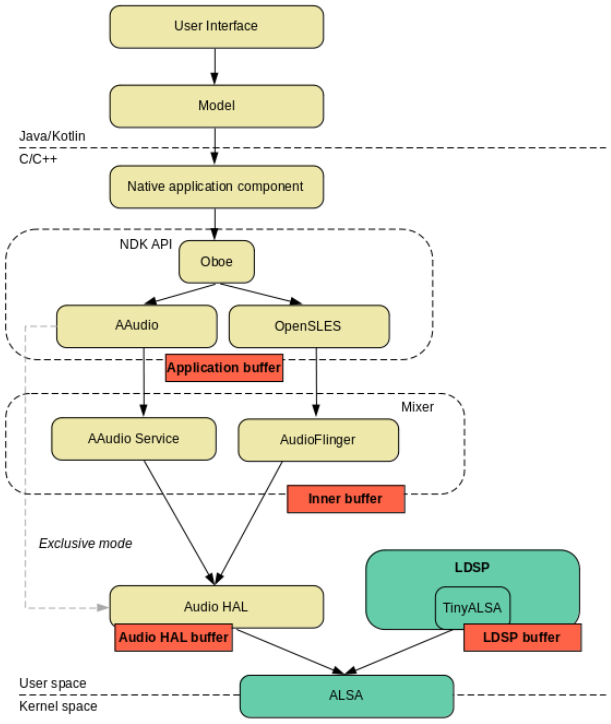


Figure 1: Audio stack comparison: LDSP stack (on the right) and modern Android audio app stack (on the left).

approach has a significant impact on code performance, as demonstrated later in this work. Currently, LDSP can be downloaded to a host computer, and developers can use it to cross-compile their native application and deploy it to their Android phone. The framework includes a C++ API, libraries and examples tailored to mobile audio development, with also direct access to the phone's sensors, buttons, touchscreen, LED lights and vibration motor. More details about the compilation process, libraries, features and examples can be found here [22], while the full source code is available at LDSP official GitHub repository[7].

LDSP is designed with portability in mind, and the low-level development and deployment workflow allow LDSP applications to bypass any resource allocation restrictions that standard apps may encounter. Moreover, LDSP is widely compatible across phones. Retro-compatibility with older Android versions is one of the most challenging aspects of Android development. The Android development framework is continually advancing, to support upcoming devices, better streamline general purpose app design and comply with security and privacy regulations. These are important issues, but contribute to the quick obsolescence of phones that are otherwise still functional. LDSP offers a solution to give new life to older devices regardless of the Android version they run or the hardware features they have. Thanks to the pure C++ implementation of LDSP, the same code can be built and run on most official Android versions, including phones with installed custom ROMs[8] based on Android.

At the core of the LDSP C++ framework lies a custom audio engine that is built around the TinyALSA library[9] and designed to directly control the Advanced Linux Audio Architecture (ALSA) kernel drivers. The audio engine provides an API to open any of the phone's capture and playback devices, synchronize them and set up a user-defined audio callback function—called 'render'. This render function runs on a dedicated thread and has direct access to the audio buffers used by the hardware ALSA driver (i.e., the ALSA *period*). Similarly to the API of Bela, a 'setup' and a 'cleanup' function are called before the start and after the termination of the render loop. LDSP's simple audio implementation optimizes the use of the phone's resources, enabling advanced audio algorithms and buffer sizes that would typically be prohibitive for Android apps (see next section). Additionally, LDSP can change the mode of operation of the CPU's scaling governor to keep the clock speed at maximum. And as many Android ROMs run preemptive kernels, the framework is designed to try to assign real-time priority to the audio thread, hence further improving timing and performance on supported phones. Figure 1 depicts LDSP audio stack and compares it with that of the modern Android audio app architecture.

4. COMPUTATIONAL PERFORMANCE ANALYSIS

4.1. Software Benchmark

We developed a C++ oscillator bank class to evaluate the computational performance of LDSP. Oscillators are essential components of traditional synthesis techniques and are also used in unconventional DMIs, as demonstrated in previous works (e.g., [23, 24]). While the number of oscillators that can be run in parallel is not a universal metric of an audio application's sonic potential, it provides valuable insight into the musical capabilities of Android phones running LDSP. To further assess the performance gain of LDSP compared to the 'standard' Android audio programming environment, we ran the same code within a custom mobile audio app and quantified the results. Additionally, we benchmarked the oscillator bank on a Bela board for reference.

The oscillator bank is initialized with the number of oscillators and a frequency range when instantiated. The frequencies of the oscillators are linearly spaced within the specified range. Each oscillator consists of a sinusoidal wavetable with linear sample interpolation. When a new sample is requested, the oscillator bank advances all the oscillators, retrieves their samples and sums them into a single value. The total amplitude is normalized to prevent clipping. The source code can be found in the LDSP GitHub repository, under *examples*.

Although the code could be optimized and tailored to individual hardware features, such as multicores or vector floating-point units, we purposely kept the focus on the audio environments and ran identical code on each device. Moreover, our goal was to measure the performance of an application designed by a creative with moderate audio programming skills, as we believe this represents a valuable test case for LDSP, which is designed with accessibility in mind.

When used within LDSP, the oscillator bank is initialized in the setup function and the oscillators' samples are constantly retrieved in the render loop to fill the output audio buffer, following the same code structure used on Bela. In contrast, running the code within an Android app required additional work. We designed a

[7] `https://github.com/victorzappi/LDSP.git`

[8] 'ROM' refers to the combination of firmware and operative system.

[9] `https://github.com/tinyalsa/tinyalsa` [Accessed on 2023/05/26]

simple application following the latest Android app architecture guidelines[10], comprising a minimalistic UI and a small audio engine. The audio engine runs the oscillator bank and sends samples to the phone's audio device, while the UI is used to pass initialization parameters to the oscillator bank, select the audio buffer size and start/stop the audio stream (Figure 2). This app was inspired by the tutorial on wavetable synthesizer by Jan Wilczek[11]. As discussed in Section 2.2, Android app development involves a combination of different programming languages. We built the UI in Kotlin, utilizing the Jetpack Compose framework, which has become a standard for modern Android UI development. For optimal performance, we implemented the audio engine in C++ and used the Oboe audio library, which serves as a wrapper for both the modern AAudio library and the legacy OpenSLES library. Oboe is designed to ensure high performance and provide backward compatibility with older phones and Android versions, which is relevant to our project's aim.

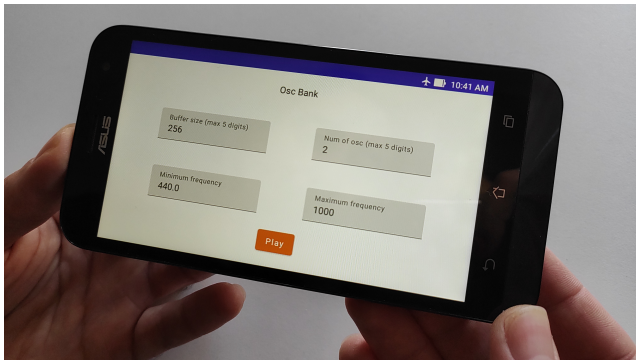


Figure 2: Phone running the oscillator bank Android audio app.

The audio engine includes a setup function that initializes and starts the playback stream, as well as a user-defined callback that is invoked whenever the application's audio buffer needs to be filled. This structure allows the oscillator bank class to be used in a similar fashion as in LDSP. The parameters and functions of the audio engine are mirrored in a Kotlin model, which can be directly accessed from the UI to set the number of oscillators and start/stop playback. The Java Native Interface in Android is used to call the C++ functions directly from the Kotlin model. The source code of the Android application can be found here: `https://github.com/victorzappi/android-osc-bank.git`.

4.2. Methodology

We measured the maximum number of oscillators that a selection of phones could play in parallel using different buffer sizes, using both LDSP and the Android app. In real-time digital audio, increasing the number of samples buffered in memory can improve computational capabilities. However, larger buffer sizes can also increase action-to-sound latency, which tends to limit the usability of the application in interactive scenarios [25]. To obtain a comprehensive view of phone performance, we tested a range of buffer sizes, focusing on powers of two, which is a common set of values for real-time audio applications. Our tests started from the lowest buffer size supported by each phone (typically 32) and increased up to 1024. Each oscillator bank configuration was tested 10 times for a minimum of 45 seconds. If any underruns occurred during this time, the configuration was deemed unreliable for real-time use, and the number of oscillators had to be decreased. We used steps of five oscillators. The first five seconds of each run were discarded from the test window, as considered a warm-up period for the application. All phones were set to airplane mode and no other apps were running during the tests.

It's worth noting that the choice of buffer size is limited by the architecture of Android audio apps. This in turn depends on both the specific hardware and the Android version running on it. For this reason, our tests included also values that depart from the initial pool and comply with each phone's Android Audio HAL as well as the inner workings of the Oboe library. We selected the *low-latency* audio device, when available on each phone, for testing. This is the audio device capable of supporting smaller buffer sizes, and all tests were run using its native sample rate and default number of channels (48 kHz and 2 channels on all tested devices). These settings were retrieved via one of the helper scripts available in LDSP.

In LDSP, we disabled audio capture, sensors, and control inputs/outputs using the appropriate command-line flags. This was done to match the features implemented in the Android app's audio engine and avoid unfair computational overhead. We passed the tested buffer size and the current number of oscillators as command-line arguments to the LDSP executable. As illustrated in Figure 1, the buffer size set within LDSP corresponds with the period size requested from the ALSA driver. We fixed the ALSA ring buffer to two periods.

Within the Android app, we paid particular attention to the audio engine setup and the parameters used to build an efficient audio stream[12]. The engine works with a high-priority callback, and the stream requests exclusive access to the audio device for optimal performance. We also explicitly set the Oboe performance mode to low-latency, which is expected to improve Android's mixer responsiveness.

Figure 1 shows that the audio signal synthesized by Android apps has to pass through several buffering stages before reaching the ALSA driver, namely: the application buffer, which is filled at every Oboe callback; the mixer's buffer, often referred to as the 'internal buffer'; and the Audio HAL buffer, which is filled by the mixer and then passed to the driver. All data transfers between these buffers happen in *bursts*, whose size depends on the audio device. Our app allows requesting the application buffer size from the UI, along with the number of oscillators. Then, the audio engine automatically tries to set the internal buffer to the same size as the application buffer, which is the lowest value possible. In some scenarios, this also allows for samples to be transferred in single bursts (see Section 4.4.1). However, the ALSA period cannot be modified with any Android audio library and is set by the Audio HAL along with the size of the ring buffer, which was fixed to twice the period size on all tested phones. Section 4.4 details how these constraints were taken into account on each phone to assure a fair comparison between LDSP and the app. As a general rule, we started by checking what buffer sizes Oboe managed to set on

[10] `https://developer.android.com/topic/architecture` [Accessed on 2023/05/26]

[11] `https://thewolfsound.com/android-synthesizer-1-app-architecture/` [Accessed on 2023/05/26]

[12] `https://developer.android.com/ndk/guides/audio/aaudio/aaudio#optimizing-performance` [Accessed on 2023/05/26]

each phone, and then we passed to LDSP a single buffer size that combined all Android buffering stages.

4.3. Tested Devices

We tested a total of six phones, which varied in release year, Android version, CPU specifications and overall tier. This heterogeneous pool of devices was selected for two reasons. First, we wanted to assess the performance of LDSP across hardware with different capabilities. Second, we wanted to gather some preliminary data on the actual portability of the environment. This is of particular importance when targeting the upcycling of obsolete devices and was one of the bigger design challenges during the initial stage of development of the project.

The first phone we tested is a Xiaomi Mi 8 Lite. It was released in 2018 and is a high-end device running a stock Android 9 ROM. The second phone is a Huawei P8 Lite from 2015 with Android 5.0 installed. At launch, it was considered a mid-tier device. The third and fourth phones are different models of the Asus ZenFone line. One is a ZenFone 2 Laser (Asus1) released in 2016, equipped with Android 6.0. The other is a ZenFone Go (Asus2) from 2015, running Android 5.1; it was considered a 'budget phone' upon release, but has a slightly higher clock speed than Asus1. The last two phones are both old LG G2 Mini models released in 2014 (LG1 and LG2). LG1 runs Lineage OS 14.1, a custom ROM equivalent to Android 7.0, while LG2 has a stock Android 4.4 ROM. The details and hardware specifications of the tested devices, including Bela's, are listed in Table 1.

Table 1: *Tested devices' details.*

Device	**Year**	**CPU**	**RAM**	**Android**
Xiaomi	2018	octacore 1.8-2.2 GHz	6 GB	9
Huawei	2015	octacore 1.0-1.2 GHz	3 GB	5.0
Asus1	2016	quadcore 1.2 GHz	2 GB	6.0
Asus2	2015	quadcore 1.3 GHz	1 GB	5.1
LG1	2014	quadcore 1.2 GHz	1 GB	7.0
LG2	2014	quadcore 1.2 GHz	1 GB	4.4
Bela	2013	singlecore 1 GHz	512 MB	-

4.4. Results

4.4.1. Android 9

The Xiaomi phone runs Android 9, enabling Oboe to utilize the modern AAudio library. AAudio allows for exclusive access to the audio device, bypassing the internal buffer, whose size was therefor not taken into account when calculating the total buffers tested via LDSP. The app reported a size of 192 samples for both the Audio HAL buffer (i.e., the ALSA period) and the burst size. Therefore, only total buffer sizes larger than 192 samples could be tested on both the app and LDSP. We increased the application buffer size in steps of 192 samples to optimize data transfer to the HAL within the app and avoid unnecessary overhead. The only exception is the starting value of 192 samples, which was approximated by setting a symbolic application buffer size of one sample. Table 2 (left) shows the values we tested, expressed as the combination of Android's application and HAL buffers, as well as the maximum number of oscillators measured in the two environments.

Despite careful choice of audio parameters that could benefit the app, results are largely in favor of LDSP. For sizes that are integer multiples of the HAL buffer, LDSP showed a performance gain that ranged from slightly above 25% to 81%. At 192 samples, the impact of removing the application buffer stage is visible in Android, and the gain reaches almost 700%. Entries below 192, accessible only to LDSP, showcase that the phone is capable of running large numbers of oscillators even with typically small buffer sizes. This reflects the overall high-end specifications of the Xiaomi.

4.4.2. Android 7.0–5.0

The Huawei, the two Asus phones and LG1 all run versions of Android that do not support AAudio[13]. On these devices, Oboe falls back to using the OpenSLES library for audio processing, leading to a series of important limitations on the audio settings and overall performance. Firstly, the audio device cannot be accessed in exclusive mode, meaning that the samples synthesized by the application have to transit through the internal buffer of Android's mixer before reaching the Audio HAL. This was taken into account when computing the total buffer sizes passed to LDSP. Secondly, the buffer size requested by audio apps is ignored as OpenSLES is set to use the most optimal configuration for both the application and internal buffer, as reported in the Audio HAL. The values in use can be checked by inspecting the mixer's status using the command `dumpsys media.audio_flinger` from an Android Debug Bridge shell. Finally, depending on the implementation of the HAL, there is no guarantee that the library matches the actual size of the bursts employed by the audio device, leading to possible overhead during data transfer.

Table 2 (right) displays the results of the tests run on the Huawei. The audio device on this phone only supports a single ALSA period size of 960 samples, as smaller and even larger sizes result in continuous underruns in the audio stream. When running the app, the mixer reported the expected 960 samples for the Audio HAL buffer (matching the supported ALSA period), plus a total of 1924 samples for the application and internal buffers—a surprisingly large value given the overall specifications of the phone. Despite running with a buffer three times smaller, LDSP showcased a performance gain of almost 340%.

Table 3 presents the results for Asus1 (top), Asus2 (middle) and LG1 (bottom). In spite of their lower technical specifications, these phones exhibit good overall audio performance and greater flexibility than the Huawei. They all support Android's *fast mixer*, which enables the use of smaller buffer sizes and requires a lower computational footprint. During app runtime, the fast mixer reported 240 samples for all three buffering stages, resulting in a total size of 720 samples.

On Asus1, OpenSLES effectively matches the optimal buffering configuration and manages to sustain 200 oscillators in real-time. LDSP provides a moderate 22.5% gain at the equivalent buffer size, but with a third of the app's optimal buffer size, it can still run 200 oscillators. Overall, Asus1's audio hardware and firmware are well-suited for interactive audio applications.

The performance of the app on Asus2 falls short compared to other device, with a maximum of only 50 oscillators. This suggests that the configuration employed by OpenSLES is sub-optimal in this case. Nevertheless, LDSP demonstrates the real potential of

[13] AAudio was introduced with Android 8.0 and is not retro-compatible.

Table 2: *Results - Xiaomi Mi 8 Lite (left), Huawei P8 Lite (right) (*native ALSA period size).*

Env. \ Buff.	**1024**	**960**	**768**	**576**	**384**	**192***	**128**	**64**	**32**
LDSP	355	350	335	320	290	235	225	190	85
Android app	-	275	265	255	160	30	-	-	-

Env. \ Buff.	**2884**	**960***
LDSP	-	175
Android app	40	-

Table 3: *Results - Asus ZenFone 2 Laser (top), Asus ZenFone GO (middle), first LG G2 Mini (bottom) (*native ALSA period sizes).*

Env. \ Buff.	**1024**	**720**	**512**	**240***	**128**	**64**	**32**
LDSP	255	245	235	200	180	165	40
Android app	-	200	-	-	-	-	-

LDSP	185	180	175	165	150	135	120
Android app	-	50	-	-	-	-	-

LDSP	180	180	180	175	175	160	110
Android app	-	165	-	-	-	-	-

the phone. When using the same total buffer size as the app, performance improves by 260%. Other buffer sizes yield good numbers of oscillators, albeit lower than those measured on Asus1. Notably, the phone stably runs 120 oscillators at the small buffer size of 32 samples. None of the other tested phones reach this count when using the lowest buffer size supported.

The test results for LG1 reveal good audio capabilities and overshadow the age of the phone. LDSP's performance gain compared to the Android app is at just under 10%. This is likely due to the fact that the CPU has already reached a plateau at a buffer size of 720 samples, where further increase in buffer size does not lead to significant improvements in the maximum count of oscillators. However, LDSP still manages to run a significant number of oscillators at lower buffer sizes, including as low as 32 samples. In fact, at this end of the scale, LG1 outperforms the high-end Xiaomi phone.

4.4.3. Android 4.4 and Bela

LG2 was the last phone to be benchmarked. Unfortunately, we discovered that it is not possible to build the Android app for its outdated Android 4.4 operating system, which is not compatible with the Jetpack Compose framework. Although alternative UI design packages are available for Android versions below 5.0, implementing such a change would have required a massive redesign of the app architecture, as well as a complete re-run of the previous tests. We deemed the app redesign beyond the scope of this work and we decided to only run LDSP on LG2. Table 4 presents the results from LG2 and Bela, both tested using the same buffer sizes, but neither having a direct comparison with the Android app.

LG2's maximum number of oscillators is identical to LG1, except for the value reported at 32 samples. This may be due to the differences in the ROMs loaded on the two phones. Android 4.4 is likely less optimized for real-time audio than Android 7.0; furthermore, LG1 runs a custom ROM, based on Android 7.0 but much more lightweight. Despite these disadvantages, the similar results between the two phones suggest that LDSP's optimizations still manage to harness most of the device's computational power for audio synthesis.

Bela's performance is limited by the BeagleBone Black's lower specifications, but its buffer size scale reaches values that are inaccessible to all the other devices, showcasing its unique ultra-low-latency capabilities.

5. DISCUSSION

LDSP outperforms the Android app in terms of the number of oscillators that can be run on all tested devices and buffer sizes. This suggests that LDSP can better utilize the computational power of the devices for audio synthesis. While this is in line with our expectations due to LDSP's optimized audio stack structure, the degree of improvement is sometimes beyond what we anticipated, even exceeding 100%. As a result, even phones that were previously considered unsuitable for audio applications using standard Android app development, such as ASUS1 and LG2, reveal the actual potential of their underlying hardware when using LDSP. This may open up new musical possibilities for already discarded technology and reminds us that we often underestimate the nature and the origin of the objects we interface with [26, 5]. In fact, one may be surprised to discover that a 2014 Android phone can reliably run more than 150 real-time oscillators using an audio buffer of only 64 samples.

However, when viewed through the lens of sustainability, this seemingly favorable scenario may pose some risks. Like circuit-bent devices [27, 5], upcycled LDSP phones may not age linearly. LDSP's unconstrained access to CPU and hardware capabilities enables the design of audio and musical applications that may put components under significant strain, such as CPU overheating, battery draining and constant high memory data rates. This can lead to a quicker decrease of the phone's lifespan. Nonetheless, we designed LDSP as a tool to repurpose phones that have already reached the end of their product life cycle, at least by modern consumeristic standards. From this perspective, we believe that even reckless phone usage via LDSP would result in an almost neutral environmental impact.

The oscillator bank experiment shows how LDSP empowers developers to optimize the balance between performance and responsiveness of their applications, by fine-tuning buffer size. While this may seem like a minor detail to experienced audio programmers and creatives, our tests expose the limitations of Android in this regard. Modern Android versions offer little flexibility when it comes to adjusting the overall buffering mechanism, while older versions such as Android 7.0 and lower straight remove this possibility.

Our experiments with various Android devices have helped us understand the rationale behind these architectural constraints. For instance, some devices like Huawei have limited audio codecs that support fixed periods only. To overcome this, Android relies on a multi-stage buffering mechanism that sits atop the Audio HAL and low-level audio driver, granting applications enough time to complete audio synthesis or processing even when the native period

Table 4: *Results - second LG G2 Mini (LG2) and Bela.*

Buff. / Env.	**1024**	**512**	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**
LG2	180	180	180	175	160	100	-	-	-	-
Bela	100	105	105	80	80	80	80	75	70	25

may not be enough. Conversely, LDSP buffers are always constrained by the native period size of the audio device. While the Android multi-buffering mechanism is useful for general-purpose multimodal applications, it adds unnecessary overhead and can introduce buffers whose sizes are inappropriate for responsive audio applications, as our results suggest.

Portability is an important feature that emerges from LDSP's results. We could have possibly either upgraded the ROM or downgraded the UI package to run the Android app on LG2. Yet, the fact that LDSP seamlessly runs on all phones including this very old one is a way more valuable result. The presence of ALSA low-level drivers is the only strict requisite for LDSP to be supported on a phone. ALSA started being included in the Android kernel since Android 2.3 and it is now the most widely used audio driver across all brands and models of phones. This means that phones released as early as 2010 are very likely to support LDSP and run the same that code we tested on our 2018 Xiaomi. While harder to find and more modest in terms of hardware as well as software capabilities, such old phones can still be spotted in flee markets, garages and even in secluded drawers within our very homes. We believe they can be instrumental to unleashing creativity in spite of and because of their limitations [23], and we are looking forward to testing one.

Compared to standard Android development, LDSP offers a streamlined solution that eliminates the need for developers to learn and use different packages and frameworks based on the phone's age and setup. Instead, LDSP is based on standard C++, making it a one-size-fits-all solution that also requires less hardware and software for development. Whereas Android Studio is typically the only option for deploying an application on a phone, LDSP is development environment-agnostic and allows for the use of leaner editors, resulting in faster and less memory/power-intensive compilation.

Additionally, the comparison with Bela showcases how LDSP offers a low entry fee for creatives. Bela can leverage block-based processing on the onboard NEON vector floating point unit to reach 700 oscillators [23]. When combining C++ and Assembly, these results hold for buffers as small as 16 samples. While similar optimization techniques can be carried out on phones via LDSP[14], a person who wants to repurpose old technologies may not be familiar with the hardware specifics of a discarded phone, nor may they want to delve into low-level optimization audio practices. Nonetheless, our entry-level code yielded better results than Bela's even on budget/old phones, suggesting a larger audio application domain with minimal coding effort.

6. CONCLUSIONS

In this paper, we discussed how LDSP can be used to harness the potential of old Android phones and foster the design of creative audio applications. We ran an experiment using six different Android phones to test the performance of an oscillator bank application built using the LDSP C++ framework. Results suggest that LDSP outperforms the standard Android app in terms of the number of oscillators that can be run on all tested devices and buffer sizes. This is likely due to LDSP's optimized audio stack structure, which better utilizes the computational power of the devices for audio synthesis.

While often referring to the impact that the size of the buffer has on the responsiveness of the application, more tests are necessary to measure the actual latency of audio applications designed with LDSP and compare them with results obtained with equivalent Android applications.

In our judgement, the central emphasis placed by LDSP as a project on upcycling and reclaiming conventional technology invites innovative approaches to engage with it, both practically and politically. This engagement entails acknowledging our agency and responsibility towards the technologies we have created, used and discarded. By reusing and exploring new ways to interact with off-the-shelf devices, we shift our focus towards sustaining them in a manner that nurtures creativity rather than solely pursuing the allure of the latest technology [28].

The utilization of ready-made technologies also holds the potential for maintenance through community-driven practices and shared knowledge [29]. Given the abundance of Android technology expertise available in varying degrees, the likelihood of continued support is somewhat assured. Therefore, it becomes paramount for us to collaborate with musicians and developers, as such partnerships would expand the project's capabilities to cater to diverse needs, interests, and skill sets while fostering a sense of community across different spheres of action.

LDSP was designed with accessibility in mind, as evident through its collection of examples and libraries, as well as its overall simplicity. However, it does require basic C++ or equivalent coding skills to fully explore its potential. In this study, we highlighted the advantages of working with low-level C++ for achieving optimal audio performance. However, it is important to note that this comes at the cost of a less accessible environment compared to other Android audio frameworks. To address this issue, we have recently introduced support for Pure Data patches by integrating libpd directly into the core low-level audio engine of LDSP. This enhancement offers users the flexibility to build their LDSP applications using Pure Data exclusively or to combine Pure Data with C++, allowing for a tailored balance between code complexity and performance. In fact, it is worth mentioning that the use of libpd introduces some computational overhead that may impact the performance of audio applications (this is seen in Bela too). We plan to conduct further tests to quantify this impact within the LDSP environment.

[14] All the tested devices come equipped with the NEON.

7. REFERENCES

[1] Alessio Balsini, Tommaso Cucinotta, Luca Abeni, Joel Fernandes, Phil Burk, Patrick Bellasi, and Morten Rasmussen,

"Energy-efficient low-latency audio on android," *Journal of Systems and Software*, vol. 152, pp. 182–195, 2019.

[2] Margaret Butler, "Android: Changing the mobile landscape," *IEEE Pervasive Computing*, vol. 10, no. 1, pp. 4–7, 2011.

[3] Andrew McPherson, "Bela: An embedded platform for low-latency feedback control of sound," *The Journal of the Acoustical Society of America*, vol. 141, no. 5, pp. 3618–3618, 2017.

[4] Zak Argabrite, Jim Murphy, Sally Jane Norman, and Dale Carnegie, "Technology is Land: Strategies towards decolonisation of technology in artmaking," in *NIME 2022*, 2022.

[5] Enrico Dorigatti and Raul Masu, "Circuit bending and environmental sustainability: Current situation and steps forward," in *NIME 2022*, 2022.

[6] Joe Cantrell, "The timbre of trash: Rejecting obsolescence through collaborative new materialist sound production," *JAAAS: Journal of the Austrian Association for American Studies*, vol. 1, no. 2, pp. 217–229, 2020.

[7] Vítor N Palmeira, Graziela F Guarda, Luiz Fernando Whitaker Kitajima, et al., "Illegal international trade of e-waste–europe," *Detritus*, vol. 1, no. 1, pp. 48, 2018.

[8] Marina Proske, Janis Winzer, Max Marwede, Nils F Nissen, and Klaus-Dieter Lang, "Obsolescence of electronics-the example of smartphones," in *2016 Electronics Goes Green 2016+(EGG)*. IEEE, 2016, pp. 1–8.

[9] Parvez Alam, "Hacking into e-waste," *Material World*, vol. 28, no. 7/8, pp. 24–27, 2020.

[10] João Tragtenberg, Gabriel Albuquerque, and Filipe Calegario, "Nime 2021," in *Gambiarra and Techno-Vernacular Creativity in NIME Research*, 2021.

[11] Rômulo Vieira and Flávio Schiavoni, "Fliperama: An affordable arduino based midi controller," in *NIME 2020*, 2020.

[12] Ignacio Orobitg, Adolfo Subieta, Federico Uslenghi, and Federico Wiman, "El desarrollo de la música electroacústica en buenos aires," *Revista del Instituto de Investigacion Musicologica Carlos Vega*, 2003.

[13] Laewoo Kang, "Echo(): Listening to the reflection of obsolete technology," *Proceedings of the 2017 ACM Conference Companion Publication on Designing Interactive Systems*, 2017.

[14] JGA Lima, "The gatorra: a technologically disobedient instrument for protest music," in *xCoAx 2018: Proceedings of the Sixth Conference on Computation, Communication, Aesthetics & X*, 2018, pp. 55–64.

[15] Caleb Stuart, "Damaged sound: Glitching and skipping compact discs in the audio of yasunao tone, nicolas collins and oval," *Leonardo Music Journal*, vol. 13, no. 1, pp. 47–52, 2003.

[16] Timothy Tate, "The Concentric Sampler: A musical instrument from a repurposed floppy disk drive," in *NIME 2022*, 2022.

[17] Karl Yerkes, Greg Shear, and Matthew James Wright, "Disky: a diy rotational interface with inherent dynamics," in *NIME 2010*, 2010.

[18] Benjamin Taylor, Jesse T Allison, William Conlin, Yemin Oh, and Daniel Holmes, "Simplified expressive mobile development with nexusui, nexusup, and nexusdrop.," in *NIME 2014*, 2014, pp. 257–262.

[19] Daniel Iglesia and Iglesia Intermedia, "The mobility is the message: The development and uses of mobmuplat," in *Pure Data Conference (PdCon16). New York*, 2016.

[20] Romain Michon, Yann Orlarey, Stéphane Letz, Dominique Fober, and Catinca Dumitrascu, "Mobile music with the faust programming language," in *Perception, Representations, Image, Sound, Music: 14th International Symposium, CMMR 2019, Marseille, France, October 14–18, 2019, Revised Selected Papers*, 2021, pp. 307–318.

[21] Ben Cahill and Stefania Serafin, "Guitar simulator: An audio-haptic instrument for android smartphones.," in *The Seventh International Workshop on Haptic and Audio Interaction Design August 23-24 2012 Lund, Sweden*, 2012, pp. 19–20.

[22] Carla Sophie Tapparo, Brooke Chalmers, and Victor Zappi, "Leveraging android phones to democratize low-level audio programming.," in *NIME*, 2023.

[23] Victor Zappi and Andrew McPherson, "Design and use of a hackable digital instrument," in *Proceedings of the International Conference on Live Interfaces*, 2014.

[24] Adnan Marquez-Borbon, "Perceptual learning and the emergence of performer-instrument interactions with digital music systems," in *Proceedings of A Body of Knowledge - Embodied Cognition and the Arts Conference 2016*, 2016.

[25] Andrew P McPherson, Robert H Jack, Giulio Moro, et al., "Action-sound latency: Are our tools fast enough?," in *NIME 2016*, 2016.

[26] Alexandra Rieger and Spencer Topel, "Driftwood: Redefining sound sculpture controllers.," in *NIME 2016*, 2016, pp. 158–159.

[27] M Premalatha, Tabassum-Abbasi, Tasneem Abbasi, and SA Abbasi, "The generation, impact, and management of e-waste: State of the art," *Critical Reviews in Environmental Science and Technology*, vol. 44, no. 14, pp. 1577–1678, 2014.

[28] Steven J Jackson, "11 rethinking repair," *Media technologies: Essays on communication, materiality, and society*, pp. 221–39, 2014.

[29] Mela Bettega, Raul Masu, Nicolai Brodersen Hansen, and Maurizio Teli, "Off-the-shelf digital tools as a resource to nurture the commons," in *Proceedings of the Participatory Design Conference 2022-Volume 1*, 2022, pp. 133–146.

[30] Ana Delgado and Blanca Callén, "Do-it-yourself biology and electronic waste hacking: A politics of demonstration in precarious times," *Public Understanding of Science*, vol. 26, no. 2, pp. 179–194, 2017.

[31] Juan Pablo Martinez Avila, João Tragtenberg, Filipe Calegario, Ximena Alarcon, Laddy Patricia Cadavid Hinojosa, Isabela Corintha, Teodoro Dannemann, Javier Jaimovich, Adnan Marquez-Borbon, Martin Matus Lerner, Miguel Ortiz, Juan Ramos, and Hugo Solís García, "Being (A)part of NIME: Embracing Latin American Perspectives," in *NIME 2022*, 2022.

NEURAL GREY-BOX GUITAR AMPLIFIER MODELLING WITH LIMITED DATA

Štěpán Miklánek[1,2,], Alec Wright[1], Vesa Välimäki[1,†] and Jiří Schimmel[2]*

[1]Acoustics Lab, Department of Information and Communications Engineering, Aalto University, Espoo, Finland
[2]Department of Telecommunications, Brno University of Technology, Brno, Czech Republic
stepan.miklanek@vut.cz

ABSTRACT

This paper combines recurrent neural networks (RNNs) with the discretised Kirchhoff nodal analysis (DK-method) to create a grey-box guitar amplifier model. Both the objective and subjective results suggest that the proposed model is able to outperform a baseline black-box RNN model in the task of modelling a guitar amplifier, including realistically recreating the behaviour of the amplifier equaliser circuit, whilst requiring significantly less training data. Furthermore, we adapt the linear part of the DK-method in a deep learning scenario to derive multiple state-space filters simultaneously. We frequency sample the filter transfer functions in parallel and perform frequency domain filtering to considerably reduce the required training times compared to recursive state-space filtering. This study shows that it is a powerful idea to separately model the linear and nonlinear parts of a guitar amplifier using supervised learning.

1. INTRODUCTION

Virtual analogue (VA) modelling [1, 2, 3] is a broad topic that still offers room to combine different approaches to obtain faithful digital models of real devices. Approaches to VA modelling are often divided into "black-box" methods, which require almost no knowledge of the target device's inner workings [4]. These methods are limited because they typically do not allow for simulating user controls, which are present in most analogue devices.

On the other hand, there are "white-box" VA modelling techniques, which create discretised versions of the actual electrical circuits and allow for simulation of the behaviour of variable components such as potentiometers [5, 6, 7]. Typically, these methods are based on wave digital filters [2, 8, 9] or nonlinear state-space representations [10, 11, 12]. Nonlinear white-box models often require the utilisation of approximation or look-up tables in order to run efficiently in real time [10]. The overall modelling accuracy is also affected by the exactness of physical models of nonlinear components, namely diodes, transistors or vacuum tubes, which often require fitting to measurements from real components [13]. Furthermore, component values listed in schematics may need to be optimised using data recorded from the device to account for inaccurate schematics and component tolerances [14].

With the emergence of the concept of differentiable digital signal processing [15], recent works have shown that it is possible to adapt white-box modelling methods in a deep learning scenario to either discover unknown values of components in analogue circuit models [16] or to replace some of the computationally demanding nonlinear circuit elements with small neural networks [17, 18].

Finally, we can define the "grey-box" approach to VA modelling. The methods that fall into this category require some knowledge of the inner structure of given devices but also rely on measurements [19], such as the state trajectory network method [20]. Various grey-box techniques were previously successfully applied to create models of guitar amplifiers [21], time-varying effects [22] or dynamic range compressors [23].

Simulation of vacuum tube guitar amplifiers is among the popular VA modelling subfields. Numerous works utilising deep learning methods for black-box guitar amplifier modelling have been published in the last few years [24, 25, 26, 27, 28]. One way to adapt these models to emulate user controls is to capture audio datasets of various control settings and allow the model to learn the dependencies from data using conditioning [29]. However, simulating multiple controls with conditioning calls for a reliable automated method for sampling numerous control combinations, making this task unsuitable for manual data collection [30].

In this paper, we propose a grey-box model that combines recurrent neural networks (RNNs) for the nonlinear preamplifier and power amplifier simulation, with a white-box linear state-space model of a guitar amplifier equaliser section, commonly referred to as the tone stack [31]. The main contributions of this paper are as follows. We show that our model needs only a fraction of the data to learn the tone stack behaviour accurately, when compared to an RNN baseline model, whilst also generalising far better to unseen data. Although the differentiable tone stack model was already presented in [16], we utilise the frequency sampling of the tone stack state-space model for frequency domain filtering, which results in considerably faster training times when compared to recursive time domain filtering. In addition, we adapt the discretised Kirchhoff nodal analysis (DK-method) to efficiently derive multiple state-space filters in a deep learning framework.

The rest of the paper is structured as follows. Sec. 2 describes the modelled guitar amplifier and the data acquisition process. In Sec. 3, we describe the differentiable tone stack model, the frequency sampling of the tone stack filter for deep learning purposes, and a method used for frequency domain filtering. Sec. 4 describes the proposed neural network model and the hyperparameters used for training. Sec. 5 presents our experiments. Sec. 6 summarises the objective and subjective results. Finally, Sec. 7 concludes.

2. MODELLED DEVICE

The device in question is a Marshall JVM 410H vacuum tube amplifier. For the channel setting modelled in the work, the circuit topology consists of a preamplifier followed by a tone stack and a

[*] Work carried out during a research visit to the Aalto Acoustics Lab in Feb.–Mar. 2023.

[†] This research is part of the activities of the Nordic Sound and Music Computing Network—Nordic SMC (NordForsk project number 86892).

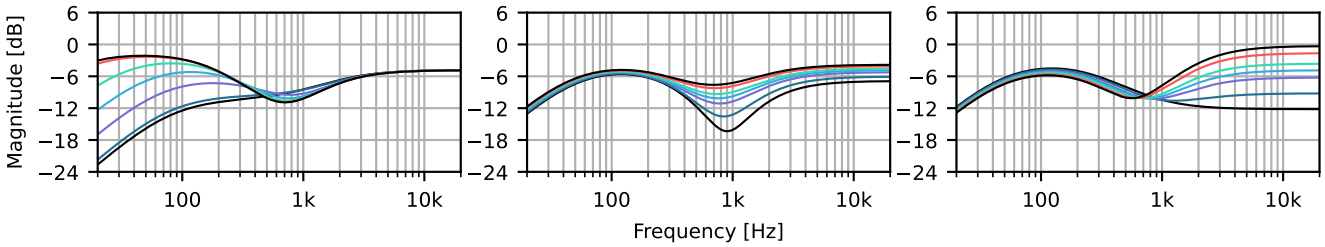


Figure 1: *Frequency responses of the JVM tone stack model with each control (bass, middle, and treble) varied from 0 to 10.*

Table 1: *Tone stack settings for evaluation on parameter values not seen in training. The values are normalised to the range* $[0, 1]$ *and denoted by* c_b *for the bass control,* c_m *for the middle control, and* c_t *for the treble control.*

c_b	c_m	c_t	c_b	c_m	c_t	c_b	c_m	c_t
0.1	0.3	0.7	0	0	1	1	0	0
0.3	0.7	0.1	0	1	0	1	0	1
0.65	0.85	0.35	0	1	1	1	1	0

power amplifier. The preamplifier is controlled by a "gain" knob, which adjusts the signal level before the tube clipping stages. The tone stack has a well known topology used in Fender, Vox, and Marshall amplifiers (FMV), described in Sec. 3.1. Its frequency response can be altered by three potentiometers, labelled as "bass", "middle", and "treble". Fig. 1 shows frequency responses of the JVM tone stack model as each control is varied. After the tone stack, the signal level is adjusted by a "volume" knob followed by the power amplifier, which has three controls. A "master" knob adjusts the signal level before the power tubes, and "resonance" and "presence" knobs alter the low- and high-frequency content.

2.1. Dataset Description

It has previously been shown that the amount of training data required for black-box neural guitar amplifier modelling is relatively low [28]. Thus, we composed a 6-min-long dataset of guitar and bass audio files of different playing styles and genres. We split the dataset into three parts: the first 4 min are used for training, and the remaining 2 min are split in half for validation and testing. All the sounds were taken from IDMT datasets described in [32, 33].

For the sake of simplicity, we model a single channel of the amplifier, which is labelled as "OD1" and produces a relatively high amount of distortion. We further limit ourselves to only making the tone stack knobs fully controllable. Therefore, we set the channel volume to maximum (10) and the gain, master, resonance and presence to midpoint (5) in all cases. Then, the entire dataset was processed through the amplifier multiple times with different settings of the bass, middle, and treble controls. We varied each knob from minimum (0) to 10 with a step of 2. Each time a single control was varied, the remaining two were set to 5. This resulted in 6 different output signals for each control. In addition, we also recorded 3 output signals, where all knobs were set to 0, 5, and 10.

We captured 9 additional output signals whilst processing only the 1-min test subset to further evaluate the model performance on unseen parameter settings. These tone stack settings, normalised to the range of $[0, 1]$, are shown in Table 1. For the first three output signals, the tone stack was set to values between the steps used for capturing the training subset. In the remaining cases, the tone stack was set to various combinations of extreme values to assess how well the models were able to generalise the non-orthogonal control behaviour described in [31].

The recording was carried out in the same fashion as in [29]. The output signals were recorded at a sampling rate of 44.1 kHz from the speaker output of the amplifier using a Two Notes Torpedo Captor 8 reactive load connected to a line input of an RME UCX USB audio interface.

3. DIFFERENTIABLE TONE STACK MODEL

We identified the DK-method [5] as a suitable analogue circuit discretisation technique to implement a differentiable version of the tone stack model. The DK-method allows an automated derivation of the state-space model from a list of virtual electronic components. We use the version of the DK-method proposed in [12]. In comparison with the earlier version [5], it allows more efficient handling of the variable components (potentiometers), which is useful not only for inference but also for adapting it for deep learning using PyTorch [34] as shown in Sec. 3.1. In this work, we use the DK-method only to derive a linear model. For more information about modelling nonlinear systems with the DK-method, we refer to other sources [10, 11, 35].

3.1. Derivation of the Tone Stack Model

The first step is to construct so-called incidence matrices that specify to which circuit nodes the individual components are connected. The number of rows in the incidence matrix is equal to the number of components, and the number of columns is equal to the number of circuit nodes (excluding the ground node). Entries in each row are given by positive and negative poles of the components, and are marked by $(+1)$ and (-1), respectively. The negative pole is omitted for components connected to the ground node. In the case of the tone stack model, we need to create five incidence matrices where $\mathbf{N}_\mathrm{R}$ is for resistors, $\mathbf{N}_\mathrm{V}$ for variable resistors, $\mathbf{N}_\mathrm{x}$ for capacitors, $\mathbf{N}_\mathrm{u}$ for the voltage source v_in, and $\mathbf{N}_\mathrm{o}$ for the output voltage v_out.

Next, we can build the system matrix $\mathbf{S}_0$, excluding the variable resistors, defined as

$$\mathbf{S}_0 = \begin{pmatrix} \mathbf{N}_\mathrm{R}^\mathsf{T}\mathbf{G}_\mathrm{R}\mathbf{N}_\mathrm{R} + \mathbf{N}_\mathrm{x}^\mathsf{T}\mathbf{G}_\mathrm{x}\mathbf{N}_\mathrm{x} & \mathbf{N}_\mathrm{u}^\mathsf{T} \\ \mathbf{N}_\mathrm{u} & \mathbf{0} \end{pmatrix}, \tag{1}$$

where $\mathbf{G}_\mathrm{R}$ and $\mathbf{G}_\mathrm{x}$ are diagonal matrices containing the parameter values of each resistor and capacitor respectively. Following the discretisation scheme from [12], the values in $\mathbf{G}_\mathrm{R}$ are conductances given by $\frac{1}{R_i}$, and the values in $\mathbf{G}_\mathrm{x}$ are computed by $\frac{2C_i}{T}$, where T is the sampling period.

To ensure the system matrix $\mathbf{S}_0$ is invertible, we augment the matrices $\mathbf{N}_\mathrm{R}$ and $\mathbf{G}_\mathrm{R}$ with virtual constant resistors [10], which

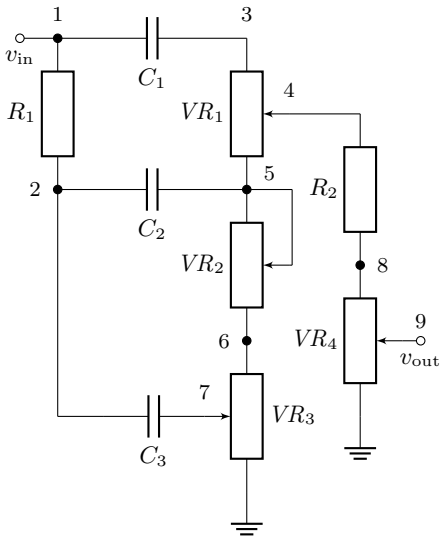


Figure 2: *Schematic of the JVM tone stack circuit, where the potentiometers VR_i are adjusted by control knobs.*

are connected in parallel to the resistors of the potentiometers VR_3 and VR_4. Consequently, we derive the coefficient matrices

$$\mathbf{A_0} = \begin{pmatrix} 2\mathbf{G}_\text{x}\mathbf{N}_\text{x} & \mathbf{0} \end{pmatrix} \mathbf{S_0}^{-1} \begin{pmatrix} \mathbf{N}_\text{x} & \mathbf{0} \end{pmatrix}^\mathsf{T} - \mathbf{I} \tag{2}$$

$$\mathbf{B_0} = \begin{pmatrix} 2\mathbf{G}_\text{x}\mathbf{N}_\text{x} & \mathbf{0} \end{pmatrix} \mathbf{S_0}^{-1} \begin{pmatrix} \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{3}$$

$$\mathbf{D_0} = \begin{pmatrix} \mathbf{N}_\text{o} & \mathbf{0} \end{pmatrix} \mathbf{S_0}^{-1} \begin{pmatrix} \mathbf{N}_\text{x} & \mathbf{0} \end{pmatrix}^\mathsf{T} \tag{4}$$

$$\mathbf{E_0} = \begin{pmatrix} \mathbf{N}_\text{o} & \mathbf{0} \end{pmatrix} \mathbf{S_0}^{-1} \begin{pmatrix} \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{5}$$

and the helper matrices

$$\mathbf{Q} = \begin{pmatrix} \mathbf{N}_\text{V} & \mathbf{0} \end{pmatrix} \mathbf{S_0}^{-1} \begin{pmatrix} \mathbf{N}_\text{V} & \mathbf{0} \end{pmatrix}^\mathsf{T} \tag{6}$$

$$\mathbf{U}_\text{x} = \begin{pmatrix} \mathbf{N}_\text{x} & \mathbf{0} \end{pmatrix} \mathbf{S_0}^{-1} \begin{pmatrix} \mathbf{N}_\text{V} & \mathbf{0} \end{pmatrix}^\mathsf{T} \tag{7}$$

$$\mathbf{U}_\text{o} = \begin{pmatrix} \mathbf{N}_\text{o} & \mathbf{0} \end{pmatrix} \mathbf{S_0}^{-1} \begin{pmatrix} \mathbf{N}_\text{V} & \mathbf{0} \end{pmatrix}^\mathsf{T} \tag{8}$$

$$\mathbf{U}_\text{u} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \end{pmatrix} \mathbf{S_0}^{-1} \begin{pmatrix} \mathbf{N}_\text{V} & \mathbf{0} \end{pmatrix}^\mathsf{T} \tag{9}$$

where $\mathbf{0}$ and $\mathbf{I}$ are zero and identity matrices of appropriate dimensions. Finally, we compute the state-space matrices $\mathbf{A} \in \mathbb{R}^{3\times3}$, $\mathbf{B} \in \mathbb{R}^{3\times1}$, $\mathbf{D} \in \mathbb{R}^{1\times3}$, and $\mathbf{E} \in \mathbb{R}^{1\times1}$ as

$$\mathbf{A} = \mathbf{A_0} - 2\mathbf{G}_\text{x}\mathbf{U}_\text{x}(\mathbf{R_V} + \mathbf{Q})^{-1}\mathbf{U}_\text{x}^\mathsf{T} \tag{10}$$

$$\mathbf{B} = \mathbf{B_0} - 2\mathbf{G}_\text{x}\mathbf{U}_\text{x}(\mathbf{R_V} + \mathbf{Q})^{-1}\mathbf{U}_\text{u}^\mathsf{T} \tag{11}$$

$$\mathbf{D} = \mathbf{D_0} - \mathbf{U}_\text{o}(\mathbf{R_V} + \mathbf{Q})^{-1}\mathbf{U}_\text{x}^\mathsf{T} \tag{12}$$

$$\mathbf{E} = \mathbf{E_0} - \mathbf{U}_\text{o}(\mathbf{R_V} + \mathbf{Q})^{-1}\mathbf{U}_\text{u}^\mathsf{T} \tag{13}$$

where $\mathbf{R}_\text{V} \in \mathbb{R}^{7\times7}$ is a diagonal matrix, which contains the resistances of the variable resistors given by $b\,VR_2$ for the bass control, $\frac{2(1-m)}{2-(1-m)}\,VR_3$ and $\frac{2m}{2-m}\,VR_3$ for the mid control, and $(1-t)\,VR_1$ and $t\,VR_1$ for the treble control. The coefficients $t \in [0,1]$, $m \in [0,1]$, and $b \in [0,1]$ denote the tone stack potentiometer settings. The resistances of the volume potentiometer VR_4 are derived in the same way as in the case of VR_3. However, they are left static and define a load for the tone stack circuit. The schematic of the modelled circuit with numbered nodes is shown in Fig. 2.

To adapt the model to be used in a deep learning framework, we can easily define trainable coefficients α_{R_i}, α_{VR_i} and α_{C_i} to optimise the component values from Table 2 during the model training similarly to [16]. We do this because component values taken directly from the schematic do not precisely correspond to the values of real components. Additionally, we use a scaled sigmoid function defined by $f(\alpha) = t_1 + \sigma(\alpha)t_2$, where t_1 and t_2 are fixed to limit the range in which the component values are adjusted. We set $t_1 = 0.8$ and $t_2 = 0.4$ to achieve $\pm20\%$ tolerance.

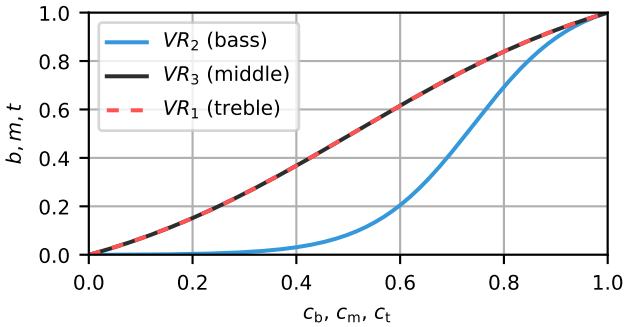


Figure 3: *Nonlinear potentiometer tapers.*

Table 2: *Component values of the JVM tone stack.*

Name	Value	Name	Value	Name	Value
R_1	33 kΩ	VR_2	1 MΩ	C_1	470 pF
R_2	39 kΩ	VR_3	20 kΩ	C_2	22 nF
VR_1	200 kΩ	VR_4	1 MΩ	C_3	22 nF

The resistances of the variable resistors in $\mathbf{R}_\text{V}$ are also dependent on conditioning vectors $\mathbf{c}_\text{b}$, $\mathbf{c}_\text{m}$, and $\mathbf{c}_\text{t}$, which describe how the controls were set on the amplifier. The number of elements in a single conditioning vector is given by the number of audio segments in the training batch of size s. The vectors $\mathbf{c}_\text{b}$, $\mathbf{c}_\text{m}$, and $\mathbf{c}_\text{t}$ cannot be used to directly alter the variable resistances. Thus, we add nonlinear trainable tapers for potentiometers VR_1, VR_2, and VR_3, shown in Fig. 3. The same tapers were used in [16] to form a two layer neural network defined by $f(x) = w_1\tanh(w_2x+b_2)+b_1$, where tanh is a hyperbolic function, x is the layer input, and $w_{1\text{–}2}$ and $b_{1\text{–}2}$ are trainable weights and biases of the respective layers. We found that leaving w_1 and b_2 unrestricted does not assure that the mapping output will stay in the range of $[0,1]$, which is needed to compute the variable resistances correctly. To solve this, we compute

$$w_1 = 1/[\tanh(w_2 + b_1) - \tanh(b_1)], \tag{14}$$

$$b_2 = -w_1\tanh(b_1), \tag{15}$$

which results in only two free parameters w_2 and b_1 as described in [36]. The same work also specifies $w_{1\text{–}2}$ and $b_{1\text{–}2}$ parameter values for fitting either linear or logarithmic potentiometer tapers that we use to initialise the parameters of the mapping neural networks.

Since we recorded the audio signals from the amplifier with various settings of controls, each audio segment in a training batch has different conditioning values assigned to it due to dataset shuffling. As a result, the number of state-space representations we need to compute is also equal to s. To achieve this, we exploit the tensor broadcasting semantics of PyTorch[1] and adapt the DK-method to derive multiple state-space filters efficiently.

If we return to the equations (10)–(13), it can be seen that the state-space matrices will be different each time the variable resistances in $\mathbf{R}_\text{V}$ change. Since each audio segment in the training

[1] https://pytorch.org/docs/stable/notes/broadcasting.html

batch needs to be filtered differently according to the conditioning vectors $\mathbf{c}_\mathrm{B}$, $\mathbf{c}_\mathrm{M}$, and $\mathbf{c}_\mathrm{T}$, we can create $\mathbf{R}_\mathrm{V} \in \mathbb{R}^{s\times 7\times 7}$, which is a tensor of variable resistances where the first dimension is equal to the batch size s. In other words, it is a tensor containing the variable resistances corresponding to the conditioning data for each item in the training batch, which are in turn used to obtain different state-space filters for each audio segment. If we use this tensor in computation of (10)–(13), we get tensors $\mathbf{A} \in \mathbb{R}^{s\times 3\times 3}$, $\mathbf{B} \in \mathbb{R}^{s\times 3\times 1}$, $\mathbf{D} \in \mathbb{R}^{s\times 1\times 3}$, and $\mathbf{E} \in \mathbb{R}^{s\times 1\times 1}$ as a result. This is relatively simple to implement using PyTorch, as the tensors will be correctly broadcasted as long as the remaining dimensions are compatible. Considering that we need to recompute the filters each time the neural network model parameters are updated, this approach is much less computationally expensive than deriving multiple state-space representations sequentially.

3.2. Frequency Sampling of the Tone Stack Filter

Several related works have already described frequency sampling of infinite impulse response (IIR) filters for deep learning purposes [37, 38, 39, 40]. Although IIR filters can be applied recursively in the time domain [41], the finite impulse response (FIR) approximation by frequency sampling allows for much faster training times [38]. Previous works investigated mainly frequency sampling of second-order sections (biquads). Nevertheless, we can do the same with state-space filters. Let us consider a single-input single-output (SISO) discrete system with scalar input $u[n]$ and output $y[n]$, that is defined by

$$\mathbf{x}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{B}u[n], \tag{16}$$

$$y[n] = \mathbf{D}\mathbf{x}[n] + \mathbf{E}u[n], \tag{17}$$

where $\mathbf{A}$ is the system matrix, $\mathbf{B}$ is the input matrix, $\mathbf{D}$ is the output matrix, $\mathbf{E}$ is the feedthrough matrix, and $\mathbf{x}$ is the state vector. To obtain the transfer function of a linear state-space filter, first, one has to take the $\mathcal{Z}$-transform of (16), which yields

$$z\mathbf{X}(z) - z\mathbf{x}[0] = \mathbf{A}\mathbf{X}(z) + \mathbf{B}U(z), \tag{18}$$

where $\mathbf{x}[0]$ represents the initial conditions. Then, the transformed state vector is given by

$$\mathbf{X}(z) = (z\mathbf{I} - \mathbf{A})^{-1}z\mathbf{x}[0] + (z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}U(z). \tag{19}$$

The transformed filter output (17) is equal to

$$\begin{aligned} Y(z) &= \mathbf{D}\mathbf{X}(z) + \mathbf{E}U(z) \\ &= \mathbf{D}(z\mathbf{I} - \mathbf{A})^{-1}z\mathbf{x}[0] + [\mathbf{D}(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{E}]U(z). \end{aligned} \tag{20}$$

If we assume zero initial conditions ($\mathbf{x}[0] = 0$), (20) can be rearranged to obtain equation for the transfer function

$$H(z) = \frac{Y(z)}{U(z)} = \mathbf{D}(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{E}, \tag{21}$$

which can be reformulated to

$$H(z) = \frac{\det(z\mathbf{I} - \mathbf{A} + \mathbf{B}\mathbf{D}) + \det(z\mathbf{I} - \mathbf{A})\mathbf{E}}{\det(z\mathbf{I} - \mathbf{A})}, \tag{22}$$

where $\det$ denotes the matrix determinant. Then, we derive a polynomial-form transfer function from (22) defined by

$$H(z) = \frac{b_0 z^m + b_1 z^{m-1} + \cdots + b_{m-1}z + b_m}{a_0 z^m + a_1 z^{m-1} + \cdots + a_{m-1}z + a_m}, \tag{23}$$

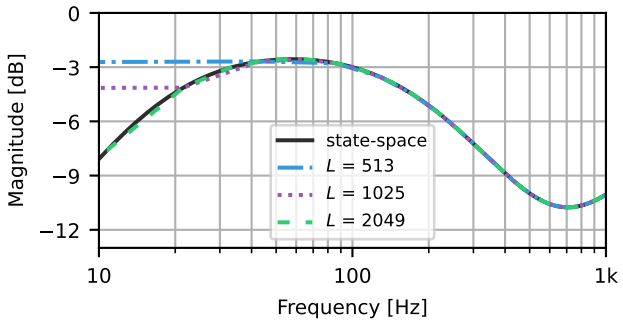


Figure 4: *Comparison of the ideal tone stack response with frequency sampled responses using a sampling vector of different lengths L, showing differences at low frequencies.*

where m is equal to the order of the system, and $b_{0\text{–}m}$ and $a_{0\text{–}m}$ are the numerator and denominator polynomial coefficients.

To frequency sample a linear state-space filter, first, we consider its frequency response denoted as $H(e^{j\omega})$, which can be frequency sampled at angular frequencies $\omega_k = 2\pi k/N$ where $k = 0, \ldots, \lfloor N/2 \rfloor$, and N is the length of the discrete Fourier transform (DFT). If we set z to $e^{j\omega_k}$, then we can easily derive $H(e^{j\omega_k})$ from the transfer function $H(z)$. Similarly as in [37], we can frequency sample $H(z)$ by combining frequency sampled m-sample delays $(z^{-m})_N \in \mathbb{C}^{\lfloor N/2 \rfloor + 1}$ and computing

$$H(e^{j\omega_k}) = H_N[k] = \frac{\sum_{m=0}^{M} b_m (z^{-m})_N[k]}{\sum_{m=0}^{M} a_m (z^{-m})_N[k]}. \tag{24}$$

A transfer function and frequency response of a linear state-space model can be computed using MATLAB or SciPy Python library functions called `ss2tf` and `freqz`, respectively. However, we need to compute several responses of state-space representations on a graphics processing unit (GPU) in parallel. To overcome this issue, we implemented differentiable methods for the PyTorch tone stack model to derive transfer functions and frequency responses for any number of SISO state-space filters simultaneously.

A few precautions must be taken when frequency sampling IIR filters. First of all, a sufficient sampling vector length L must be used to maintain the fidelity of the frequency response, especially at low frequencies. We found that $L = 2049$ is satisfactory for the tone stack model despite slight inaccuracies below 30 Hz as shown in Fig. 4. Increasing L above 2049 makes it possible to get even closer to the ideal response, but it is not necessary for model training purposes.

A less obvious problem arises from the numerical precision chosen for deriving the frequency responses. Deep learning libraries often use single-precision floating-point numbers for all computations, which can result in incorrect frequency responses due to rounding errors. This problem is not straightforward to detect as, in the case of the tone stack model, the responses are miscalculated only with particular settings of the virtual potentiometers. A simple solution is to use double-precision numbers when deriving the polynomial coefficients of the state-space transfer function $H(z)$. The purple dash-dotted line in Fig. 5 represents an incorrect response of the tone stack model with the controls set to $c_\mathrm{b} = 1$, $c_\mathrm{m} = 0$, and $c_\mathrm{t} = 1$ when using single-precision.

3.3. Frequency Domain Tone Stack Filtering

To significantly speed up the training process, we use a similar procedure to [23], where a one-pole IIR filter was applied in the fre-

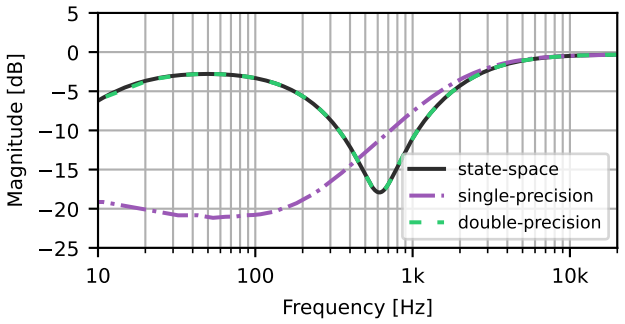


Figure 5: *Comparison of the ideal tone stack response with frequency sampled responses using single- and double-precision.*

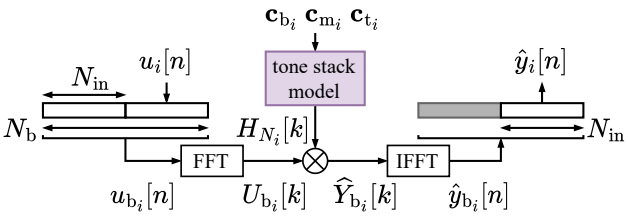


Figure 6: *Frequency domain filtering of an audio sub-segment with a tone stack frequency response.*

quency domain. We use truncated backpropagation through time (TBPTT) [42] for updating trainable parameters multiple times when processing longer audio segments. The TBPTT length determines N_{in}, which is the number of samples of the audio sub-segments used for training. Each TBPTT batch consists of s audio sub-segments $u_i[n]$ that get processed simultaneously during a single training step. The filtering process for a single audio sub-segment is depicted in Fig. 6.

First, an audio sub-segment $u_i[n]$ of length N_{in} is inserted into a buffer of length $N_{\text{b}} = 2N_{\text{in}}$. Then, we perform a fast Fourier transform (FFT) on the audio samples $u_{\text{b}_i}[n]$ stored in the buffer, and discard the negative frequency components, resulting in $\lfloor N_{\text{b}}/2 \rfloor + 1$ complex frequency coefficients $U_{\text{b}_i}[k]$. Next, we supply the conditioning values $\mathbf{c}_{\text{b}_i}$, $\mathbf{c}_{\text{m}_i}$, and $\mathbf{c}_{\text{t}_i}$ which affect the variable resistances of the tone stack model. After all virtual components of the tone stack model are updated, we can derive the state-space representation using the DK-method.

The tone stack transfer function $H_i(z)$ and frequency sampled response $H_{N_i}[k]$ is then computed. This is used to perform an element-wise multiplication of $H_{N_i}[k]$ with $U_{\text{b}_i}[k]$, which results in filtered complex frequency coefficients $\hat{Y}_{\text{b}_i}[k]$. Finally, we take the inverse FFT of $\hat{Y}_{\text{b}_i}[k]$. Since the beginning of the filtered buffer $\hat{y}_{\text{b}_i}[n]$ contains the starting transient, we only keep the last N_{in} samples, which allows for processing longer audio sequences without producing discontinuities in the filtered signal. As a result, there is no need to apply windowing functions during the filtering process. Note that we set the buffer size $N_{\text{b}} = 4096$ in order to obtain 2049 complex coefficients after performing the FFT so it matches the length of the frequency sampled response $H_{N_i}[k]$.

4. MODEL STRUCTURE AND TRAINING

The proposed grey-box model is composed of three blocks, as shown in Figure 7. The first block is a long short-term memory

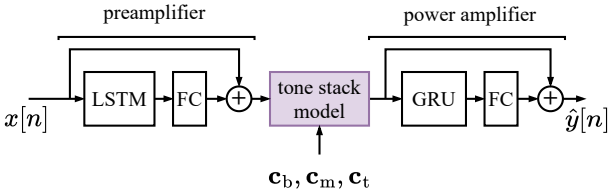


Figure 7: *Block diagram of the proposed grey-box model, which uses neural network models for the pre- and power- amplifier sections and a linear white-box model for the tone stack circuit.*

(LSTM) RNN [43] followed by the differentiable state-space tone stack model described in Sec. 3, and a gated recurrent unit (GRU) RNN [44]. The RNNs are used for modelling the preamplifier and power amplifier, respectively. Fully connected (FC) layers are added after each RNN to transform the hidden state vectors into single audio samples. We provide a reference PyTorch implementation, the dataset, and listening examples at[2].

The architecture of the RNNs is identical to a previously proposed black-box guitar amplifier model [28]. The hidden state size determines the accuracy of these models. An extensive hyperparameter search was conducted in previous works [28, 29] to assess the ideal hidden state sizes. We set the hidden size to 40 for the LSTM and to 8 in the case of the GRU. These hyperparameters were set empirically after initial training experiments on the dataset presented in Sec. 2.1.

Note that our grey-box model assumes that the preamplifier, tone stack, and power amplifier are decoupled in terms of their interaction. The Marshall JVM 410H has a feedback connection from the output transformer to the phase splitter, controlled by the "resonance" and "presence". We did not investigate how this feedback connection affects the interaction between the amplifier sections.

4.1. Objective Metrics

All models in this work were trained to minimise the Error-To-Signal (ESR) loss, which has been used extensively for modelling nonlinear audio circuits [26, 27, 28, 29]. The ESR is given by

$$\mathcal{E}_{\text{ESR}} = \frac{\sum_{n=0}^{N-1} |y[n] - \hat{y}[n]|^2}{\sum_{n=0}^{N-1} |y[n]|^2}, \tag{25}$$

where $y[n]$ is the target signal, $\hat{y}[n]$ is the predicted signal, and N is the length of the training segment.

Furthermore, we use a frequency domain error metric based on short-time Fourier transform (STFT) from [45] denoted as $\mathcal{E}_{\text{STFT}}$ solely for validation purposes. It is a linear combination of spectral convergence and log-scale STFT-magnitude error. Contrary to the ESR, it discards the phase information and provides insight into how well the models perform regarding spectral similarity. We utilise this metric because phase differences between the model output and the target signal can result in high ESR with the model still performing well perceptually [21].

4.2. Training Hyperparameters

To train the models, we use similar hyperparameters to those proposed in [29]. The training dataset was split into 0.5 s audio segments. The first 1000 samples of each segment are processed without updating the parameters of the network, which allows for the

[2]https://stepanmk.github.io/grey-box-amp

Table 3: *Number of trainable parameters, duration of a training step on a GPU, and a real-time factor (RTF) at a sampling rate of 44.1 kHz for the baseline and proposed models.*

Model Type	Params.	Train. Step (s)	RTF
RNN (baseline [28])	10417	0.16	21.28
TS (tone stack from [16])	7208	26.59	11.49
TS (proposed recursive)	7208	14.17	19.61
TS (proposed freq. domain)	7208	0.45	19.61

initialisation of the RNN states and the tone stack buffer. The rest of the samples were processed with TBPTT being applied every 2048 samples. The training batch size was set to $s = 80$. We calculated the validation loss every 2 epochs, and the maximum number of training epochs was set to 350. We applied early stopping with a patience of 15 epochs whilst monitoring the validation loss. The models were trained with the Adam optimiser with an initial learning rate of 2×10^{-3}. The learning rate was decreased by a factor of 0.5 if the validation loss did not improve for 10 consecutive epochs. The training time of the models before early stopping varied but generally took approximately 10 to 40 min on a GPU, depending on the size of the training dataset.

5. EXPERIMENTS

We hypothesise that the white-box tone stack will greatly improve the ability of the model to generalise to the unseen values described in Sec. 2.1, especially when only a small number of parameter values are seen during training. To test this we train models using different subsets of the training dataset.

As a baseline, we use a fully black-box RNN model from [28] consisting of an LSTM of hidden size 48 followed by a FC layer. We use a larger hidden size for the baseline model to compensate for the fact that our proposed model includes an additional RNN stage after the tone stack model. The black-box baseline RNN model receives the conditioning values as additional input channels, and as such has an input size of 4.

Four different datasets were used for training, with the number of unique permutations of conditioning values varying from 1 to 21. The smallest dataset contains a single permutation, with all the tone stack controls set to the midpoint (5). The second dataset has 2 additional targets, where all the tone stack controls are set to either 0 or 10. The third dataset includes all the targets from the second dataset, in addition to cases where each tone stack control is set to either 0, 2, 8, or 10, whilst the remaining controls are set to 5. This results in a total of 15 tone stack parameter permutations in the third dataset. Finally, the last dataset includes the second dataset, as well cases where each tone stack control is varied in turn to either 0, 2, 4, 6, 8 or 10, whilst leaving the rest of the controls set to 5. This results in a total of 21 unique permutations for the fourth dataset. Note that we left the nonlinear potentiometer tapers non-trainable in the case of our TS1 model, as only a single conditioning value was presented during training.

6. RESULTS

The proposed model has less trainable parameters than the baseline, however, as shown in Table 3, it is slightly slower to train. Frequency sampling of the tone stack filter significantly improves the training times on a GPU. The difference in an average training step duration is even more pronounced when compared to a previously proposed differentiable tone stack model from [16], where the filtering is applied in the time domain. In addition, the previous approach uses a different discretisation scheme and does not allow direct derivation of the transfer function needed for frequency sampling.

Table 4: *Objective results for the baseline and proposed models. Bold indicates best-performing model. The numbering of the model names corresponds to the number of unique permutations of conditioning seen during training.*

	Test		Test Unseen		Train. Dataset
Model	$\mathcal{E}_{\text{ESR}}$	$\mathcal{E}_{\text{STFT}}$	$\mathcal{E}_{\text{ESR}}$	$\mathcal{E}_{\text{STFT}}$	Duration (min)
RNN1	0.020	0.793	1.748	2.573	4
RNN3	0.025	0.933	1.446	2.511	12
RNN15	0.017	0.743	0.040	0.953	60
RNN21	0.015	0.693	**0.039**	**0.893**	84
TS1	0.011	0.760	0.048	0.854	4
TS3	0.020	0.764	**0.023**	**0.715**	12
TS15	0.029	0.780	0.038	0.805	60
TS21	0.028	0.924	0.039	0.955	84

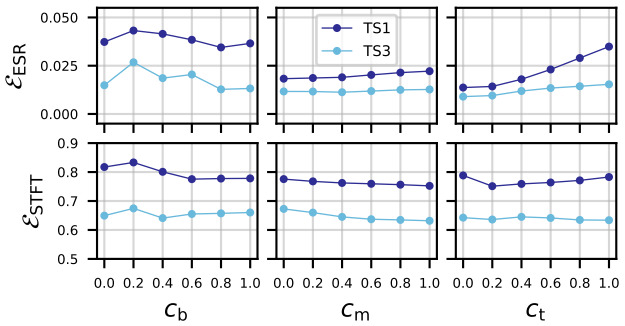


Figure 8: *Further evaluation of the proposed TS1 and TS3 models on unseen tone stack permutations: (top) time domain and (bottom) spectral errors for (left) bass, (center) middle, and (right) treble controls.*

Furthermore, we measured how long it takes to process 1 s of audio with a custom C++ implementation of our model expressed as a real-time factor (RTF) computed according to [18]. An RTF larger than 1 means the model can process the signal faster than in real time. The baseline model was found to be negligibly faster than the proposed model, as shown in Table 3.

Objective metrics for all trained models are shown in Table 4. In the case of the baseline RNN models, it is clear that the more tone stack permutations the models see during training, the better they perform on unseen permutations. This is evident from both the time and frequency domain metrics. The RNN21 that was trained on 84 min of data performed the best in comparison to other baseline models. In contrast to this, the proposed models performed well even when trained on very small datasets. The TS1 model, trained on just a single tone stack setting (4-min dataset), outperformed the best-performing baseline model in terms of STFT error computed on unseen parameter values. The TS3 performed the best of all the models, producing the smallest error on unseen data, for both metrics.

As the TS1 and TS3 models were only trained on 1 and 3 tone stack settings, we may also consider the rest of the training dataset as unseen, and use this to further evaluate the performance of these models, as shown in Fig. 8. It was observed during training that for the models TS15 and TS21, the additional data seemed to impact the model training negatively. An abrupt increase in the validation loss occurs after a few epochs, and this behaviour was observed

consistently across different training runs. One possible explanation could be that the trainable potentiometer tapers used in the tone stack model might be sensitive to errors caused by manually setting the tone stack controls on the real amplifier when recording the dataset.

6.1. Listening Tests

A MUSHRA [46] listening test was conducted to evaluate the perceptual quality of the models. In each trial, participants were presented with a reference clip that was processed by the guitar amplifier being modelled. Participants were asked to rate seven test conditions on a scale of 0 to 100, based on perceived similarity to the reference. The test conditions included five neural network models: 3 black-box RNN models, which were trained using either 1, 3, or 21 unique permutations of tone stack parameters, and 2 versions of our proposed grey-box model, trained with either 1 or 3 permutations of tone stack parameters. These were selected based on the objective results of the previous section. Additionally an anchor, created by processing the input with a $\tanh$ nonlinearity, and a hidden reference, were included in the test.

Fourteen participants completed the listening tests. Two participants identified as female and twelve as male. All participants had experience completing listening tests, and their mean age was 28.5 years. One participant was excluded from the results as they rated the hidden reference below 90 in more than 15% of the trials.

The listening test was conducted for two different tone stack settings that the models had not seen during training. Results of both test scenarios are shown in Fig. 9. First, the tone stack parameters were set to $c_\mathrm{b} = 0.1$, $c_\mathrm{m} = 0.3$, and $c_\mathrm{t} = 0.7$. In this case, the conditioning values were in-between the steps used to capture the training dataset. This represents a case that should easier for the model to predict accurately. The RNN1 and RNN3 baseline models were rated as Poor and Fair, respectively. Interestingly the RNN3 model, which was trained on more data than RNN1, performed worse. The RNN21 model, on the other hand, was rated as excellent as it was trained on a substantially larger dataset. Our proposed TS1 and TS3 models were both rated as Excellent, showing that adding the tone stack model greatly improves generalisation, even though the models were trained only on 4 and 12 min data, respectively.

In the second scenario, the tone stack parameters were set to $c_\mathrm{b} = 1$, $c_\mathrm{m} = 0$, and $c_\mathrm{t} = 0$. In this case, the RNN models trained on the small datasets were rated the worst, as in the first scenario. The RNN21 model trained on the largest dataset was only rated as Fair, contrary to how it was rated in the first scenario. This shows that the RNN models must be supplied with additional permutations of the tone stack parameters to generalise better. Conversely, the TS1 and TS3 models with the tone stack included were again rated as Excellent as shown in the bottom half of Fig. 9. The TS3 model trained on 12 min of data was rated slightly worse than the TS1 model, in contrast to the first test scenario. This also goes against the objective results, however informal listening tests confirm only slight differences in the high frequencies. We encourage readers to evaluate the models for themselves by listening to the sound examples provided on the demo page.

7. CONCLUSIONS

In this work we present a grey-box approach for guitar amplifier modelling, which allows for the inclusion of user parameters whilst requiring minimal training data. The proposed approach can be applied to many popular guitar amplifiers, as the modelled tone stack circuit is ubiquitous in the industry. We also demonstrated how a state-space model of a linear parametric circuit can be implemented using the frequency sampling method, allowing for efficient training within a deep learning framework. A subjective and objective evaluation of our proposed method demonstrates excellent generalisation to unseen data and excellent perceptual quality. Future work should validate our approach on other devices as our study was limited to a single amplifier. Using the DK-method combined with frequency sampling should also allow for incorporating other controllable linear circuits into neural network models of guitar pedals and various analogue effects.

We acknowledge that modelling different guitar amplifiers, which produce very high amounts of distortion, could result in the need to use RNNs with larger hidden sizes, thus making the model more expensive to run in real time. However, recent work [47] has shown that it is possible to prune the trainable weights of black-box RNN guitar amplifier models, resulting in a smaller effective hidden size whilst slightly improving the perceptual modelling quality. It is likely that this method could also be applied to our grey-box model.

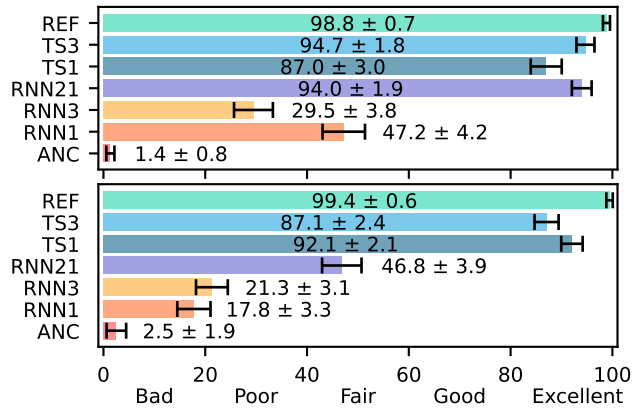


Figure 9: *MUSHRA scores with 95% confidence intervals for the tone stack settings of* $c_\mathrm{b} = 0.1$, $c_\mathrm{m} = 0.3$, *and* $c_\mathrm{t} = 0.7$ *(top) and* $c_\mathrm{b} = 1$, $c_\mathrm{m} = 0$, *and* $c_\mathrm{t} = 0$ *(bottom).*

8. REFERENCES

[1] V. Välimäki, S. Bilbao, J. O. Smith, J. Pakarinen, and D. Berners, *DAFX: Digital Audio Effects*, chapter "Virtual analog effects", pp. 473–522, Wiley, Chichester, UK, second edition, 2011.

[2] K. J. Werner, *Virtual Analog Modeling of Audio Circuitry Using Wave Digital Filters*, Ph.D. thesis, Stanford University, 2016.

[3] S. D'Angelo, "Lightweight virtual analog modeling," in *Proc. 22nd Colloquium on Music Informatics*, Udine, Italy, 2018, pp. 20–23.

[4] F. Eichas and U. Zölzer, "Black-box modeling of distortion circuits with block-oriented models," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sep. 2016.

[5] D. T. Yeh, J. S. Abel, and J. O. Smith III, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part I: Theoretical development," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 18, no. 4, pp. 728–737, Oct. 2010.

[6] M. Holters and U. Zölzer, "A generalized method for the derivation of non-linear state-space models from circuit schematics," in *Proc. 23rd European Signal Processing Conf.*, Nice, France, Sep. 2015.

[7] K. Dempwolf, M. Holters, and U. Zölzer, "Discretization of parametric analog circuits for real-time simulations," in *Proc. Int. Conf. Digital Audio Effects (DAFx-10)*, Graz, Austria, Sep. 2010.

[8] G. De Sanctis and A. Sarti, "Virtual analog modeling in the wave-digital domain," *IEEE Trans. Audio Speech Lang. Process.*, vol. 18, no. 4, pp. 715–727, May 2010.

[9] K. J. Werner, A. Bernardini, J. O. Smith, and A. Sarti, "Modeling circuits with arbitrary topologies and active linear multiports using wave digital filters," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 65, no. 12, pp. 4233–4246, Dec. 2018.

[10] J. Macak, J. Schimmel, and M. Holters, "Simulation of Fender type guitar preamp using approximation and state-space model," in *Proc. Int. Conf. Digital Audio Effects (DAFx-12)*, York, UK, Sep. 2012.

[11] F. Eichas, M. Fink, M. Holters, and U. Zölzer, "Physical modeling of the MXR Phase 90 guitar effect pedal," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, Erlangen, Germany, Sep. 2014.

[12] M. Holters and U. Zölzer, "Physical modelling of a wah-wah effect pedal as a case study for application of the nodal DK method to circuits with variable parts," in *Proc. Int. Conf. Digital Audio Effects (DAFx-11)*, Paris, France, Sep. 2011.

[13] K. Dempwolf and U. Zölzer, "A physically-motivated triode model for circuit simulations," in *Proc. Int. Conf. Digital Audio Effects (DAFx-11)*, Paris, France, Sep. 2011.

[14] B. Holmes and M. Van Walstijn, "Physical model parameter optimisation for calibrated emulation of the dallas rangemaster treble booster guitar pedal," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sep. 2016.

[15] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," *arXiv preprint*, 2020, arXiv:2001.04643.

[16] F. Esqueda, B. Kuznetsov, and J. D. Parker, "Differentiable white-box virtual analog modeling," in *Proc. Int. Conf. Digital Audio Effects (DAFx20in21)*, Vienna, Austria, Sep. 2021.

[17] J. Chowdhury and C. J. Clarke, "Emulating diode circuits with differentiable wave digital filters," in *Proc. Int. Sound and Music Computing Conf. (SMC-22)*, Saint-Étienne, France, Jun 2022, pp. 332–339.

[18] C. C. Darabundit, D. Roosenburg, and J. O. Smith III, "Neural net tube models for wave digital filters," in *Proc. Int. Conf. Digital Audio Effects (DAFx20in22)*, Vienna, Austria, Sep. 2022.

[19] J. Pakarinen and D. T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music J.*, vol. 33, no. 2, pp. 85–100, 2009.

[20] J. D. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. Int. Conf. Digital Audio Effects (DAFx-19)*, Birmingham, UK, Sep. 2019.

[21] F. Eichas and U. Zölzer, "Gray-box modeling of guitar amplifiers," *J. Audio Eng. Soc.*, vol. 66, no. 12, pp. 1006–1015, Dec. 2018.

[22] R. Kiiski, F. Esqueda, and V. Välimäki, "Time-variant gray-box modeling of a phaser pedal," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sep. 2016.

[23] A. Wright and V. Välimäki, "Grey-box modelling of dynamic range compression," in *Proc. Int. Conf. Digital Audio Effects (DAFx20in22)*, Vienna, Austria, Sep. 2022.

[24] Z. Zhang, E. Olbrych, J. Bruchalski, T. J. McCormick, and D. L. Livingston, "A vacuum-tube guitar amplifier model using long/short-term memory networks," in *Proc. IEEE SoutheastCon*, Saint Petersburg, FL, April 2018.

[25] T. Schmitz and J.-J. Embrechts, "Nonlinear real-time emulation of a tube amplifier with a long short time memory neural-network," in *Proc. Audio Eng. Soc. 144th Conv.*, Milan, Italy, May 2018.

[26] E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Int. Sound and Music Computing Conf. (SMC-19)*, Malaga, Spain, May 2019, pp. 332–339.

[27] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP19)*, Brighton, UK, May 2019.

[28] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Appl. Sci.*, vol. 10, no. 3, Jan. 2020.

[29] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. Int. Conf. Digital Audio Effects (DAFx-19)*, Birmingham, UK, Sep. 2019.

[30] L. Juvela et al., "End-to-end amp modeling: from data to controllable guitar amplifier models," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP23)*, Rhodes Island, Greece, Jun 2023.

[31] D. T. Yeh and J. O. Smith, "Discretization of the '59 Fender Bassman tone stack," in *Proc. Int. Conf. Digital Audio Effects (DAFx-06)*, Montreal, Canada, Sep. 2006.

[32] J. Abeßer, P. Kramer, C. Dittmar, and G. Schuller, "Parametric audio coding of bass guitar recordings using a tuned physical modeling algorithm," in *Proc. Int. Conf. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sep. 2013.

[33] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters," in *Proc. Int. Conf. Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sep. 2014.

[34] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 8024–8035, 2019.

[35] D. T. Yeh, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part II: BJT and vacuum tube examples," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 20, no. 4, pp. 1207–1216, May 2012.

[36] B. Holmes and M. van Walstijn, "Potentiometer law modelling and identification for application in physics-based virtual analogue circuits," in *Proc. Int. Conf. Digital Audio Effects (DAFx-19)*, Birmingham, UK, Sep. 2019.

[37] S. Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *Proc. Int. Conf. Digital Audio Effects (DAFx-20)*, Vienna, Austria, Sep. 2020.

[38] S. Nercessian, A. Sarroff, and K. J. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *Proc. ICASSP21*, Toronto, Canada, June 2021, pp. 890–894.

[39] S. Lee, H. Choi, and K. Lee, "Differentiable artificial reverberation," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 30, pp. 2541–2556, July 2022.

[40] C. J. Steinmetz, N. J. Bryan, and J. D. Reiss, "Style transfer of audio effects with differentiable signal processing," *J. Audio Eng. Soc.*, vol. 70, no. 9, pp. 708–721, Sept. 2022.

[41] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. Int. Conf. Digital Audio Effects (DAFx-20in21)*, Vienna, Austria, Sep. 2020.

[42] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Dec. 1997.

[44] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint*, 2014, arXiv:1412.3555 [cs.NE].

[45] R. Yamamoto, E. Song, and J. Kim, "Probability density distillation with generative adversarial networks for high-quality parallel waveform generation," *arXiv preprint*, 2019, arXiv:1904.04472v2.

[46] ITU, "BS.1534: Method for the subjective assessment of intermediate quality levels of coding systems," Geneva, Switzerland, 2015.

[47] D. Südholt, A. Wright, C. Erkut, and V. Välimäki, "Pruning deep neural network models of guitar distortion effects," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 31, pp. 256–264, 2023.

A GENERAL USE CIRCUIT FOR AUDIO SIGNAL DISTORTION EXPLOTING ANY NON-LINEAR ELECTRON DEVICE

Christoforos Theodorou
University Grenoble Alpes, University Savoie Mont-Blanc,
CNRS, Grenoble INP, IMEP-LAHC
Grenoble, France
christoforos.theodorou@grenoble-inp.fr

Michail Ziogas
MadNote Electronics
Thessaloniki, Greece
madnote.electronics@gmail.com

ABSTRACT

In this paper, we propose the use of the transimpedance amplifier configuration as a simple generic circuit for electron device-based audio distortion. The goal is to take advantage of the non-linearities in the transfer curves of any device, such as diode, JFET, MOSFET, and control the level and type of harmonic distortion only through bias voltages and signal amplitude. The case of a nMOSFET is taken as a case study, revealing a rich dependence of generated harmonics on the region of operation (linear to saturation), and from weak to strong inversion. A continuous and analytical Lambert-W based model was used for simulations of harmonic distortion, which were verified through measurements.

1. INTRODUCTION

Since the beginning of electrical musical instruments, musicians are trying to achieve a unique sound through their gear with all kinds of effect units. Modulation, reverb, overdrive and distortion effects are widely used for electric guitars but nowadays also by any electric string instrument, synthesizers, keyboards or even vocals. For example, a signal can be distorted through soft or hard clipping, or through any non-linear transfer function, symmetrical or asymmetrical. To achieve all these cases of signal distortion, the typical circuits used are single or multiple gain stages with or without diode clipping. For the gain stages there are hundreds of different combinations that can be used: from vacuum tubes [1], [2] to opamps [3], from silicon to germanium diodes [4], from BJTs to JFETs and MOSFETs [5], [6], each circuit has a different transfer curve and as a result a unique tone character, usually thanks to the devices' non-linear response. Diodes are used for both soft and hard or shunt clipping, in different configurations: in amplifier feedback loop or shunt to ground, respectively. The clipping of the signal can also be symmetrical or asymmetrical, leading to purely odd or odd/even harmonics.

To our knowledge, in all above cases, the non-linear devices are either used as non-linear resistors in an OPAMP's feedback loop or as shunting elements. As a result, only specific amplitude-dependent regions of non-linear response are being exploited. In this work, we propose a method with which one can make use of any non-linear I-V characteristic exploiting various voltage bias regions, while having complete control of the generated harmonics. Despite the fact that there has been an in-depth analysis of harmonic distortion in MOSFETs [7] and even more advanced FET structures [8], [9], it has not yet been used in the configuration that we propose, therefore we chose the MOSFET device as a case study to demonstrate the applicability and advantages of our method.

2. DEVELOPMENT OF THE PROPOSED METHOD

In this section, we present the step-by-step development of the proposed method, presenting the case of a n-channel MOSFET as an example for electron device-based harmonic distortion.

2.1. The transimpedance amplifier as an electron device characterization instrument

The Transimpedance amplifiers (TIA) are widely used to translate the current output of sensors like photodiode-to-voltage signals [10], since many circuits and instruments can only accept voltage input. Moreover, they are used as current preamplifiers in precision measurement instruments such as in current DC and noise characterization [11].

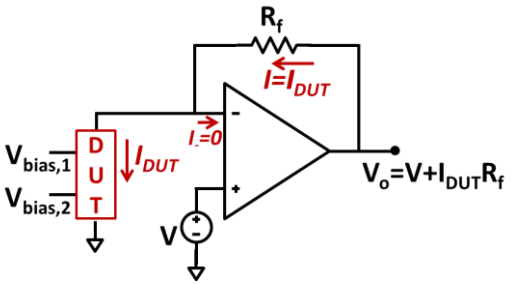


Figure 1: *Typical current-to-voltage converter configuration with transimpedance amplifier for electron device characterization (DC and noise)*

As shown in Fig. 1, the (TIAs) configuration of an operational amplifier (OPAMP) can be used as a current-to-voltage converter

for any device under test (DUT) connected at its input and controlled by various bias voltages (V, V_{bias}). Since the current flowing in the input of the OPAMP can be considered zero, and the negative feedback resistor guarantees the OPAMP's linear regime, the DUT current has to be equal to the current flowing in the feedback loop. Therefore $I_{DUT} = (V_o\text{-}V)/R_f$, because the OPAMP functions in linear mode thus $V_+ = V_-$. So, the output voltage of a TIA would be:

$$V_o = V + I_{DUT}R_f \tag{1}$$

It becomes obvious that the DC voltage output is actually independent of any other device bias voltages other than the one connected to the OPAMP. Of course, it has to be noted that we consider an ideal OPAMP with linear response for a large range of voltage swings and with no input voltage offsets. In a real scenario, one needs to correct the input offset and make sure the output voltage does not exceed the linear response region of the OPAMP, which is directly limited by the supply voltage bias. Fortunately, this can be easily regulated with the value of Rf, so that the output signal is never higher than the maximum allowed limit for linear OPAMP operation. Moreover, a real OPAMP has also a maximum output current above which it also saturates. The latter issue cannot be avoided by adjusting Rf, but only by choosing carefully the OPAMP model so that it can provide the maximum current of the connected DUT.

Now if we consider an AC output after a high-pass filter (HPF) that cuts frequency content below 10 Hz for example, we could approximate the output voltage signal as:

$$v_o = i_{DUT}R_f \tag{2}$$

where iDUT are the current variations of the DUT due to variations in one (or more) of the bias voltages. Therefore the voltage signal of the output is a direct linear function of solely the device AC current, with a gain equal to the feedback resistor value, Rf. An alternative way to obtain (2) without the use of a filter, which could impact the signal's phase at low frequencies, is to connect a voltage subtractor circuit in series with the output, to remove the same DC voltage V that we apply at the OPAMP's (+) input.

Finally, as happens with all analog audio circuits, since both the DUT and the OPAMP have DC/AC characteristics and responses that vary for each copy of the device, some manual tweaking of supply bias and DUT bias may be needed to obtain exactly the same behaviour from one circuit to another identical one.

2.2. Controlling the harmonic distortion through voltage bias and signal amplitude

In the example of Fig. 1, the current flowing through the device under test can be expressed as $I_{DUT} = I(V, V_{bias1}, V_{bias2})$. Therefore, an AC component, v_{in}, is added to one of the bias voltages, and the rest are constant with time, the small-signal AC output of the OPAMP would be:

$$v_0 = \frac{\partial I_{DUT}}{\partial V_{in}} v_{in} R_f = f(V_{in}, v_{in}, V_{bias,1-2}, \dots) \tag{3}$$

where $\frac{\partial i_{DUT}}{\partial V_{in}}$ is a function of the DC bias voltages and corresponds to the 1st order sensitivity of I_{DUT} to V_{in}. This means that the dynamic output response is unique and fully controlled by the choice of signal input and the bias voltages. Now, if we also account for higher order components [7] introduced by a non-linear relation of I_{DUT} with V_{bias}, we obtain:

$$v_0 = R_f \left(\frac{\partial I_{DUT}}{\partial V_{in}} v_{in} + \frac{1}{2} \frac{\partial^2 I_{DUT}}{\partial {V_{in}}^2} {v_{in}}^2 + \dots + \frac{1}{n!} \frac{\partial^n I_{DUT}}{\partial {V_{in}}^n} {v_{in}}^n \right) \tag{4}$$

We can therefore see that with the configuration of Fig. 1, even the generated harmonics that define the distortion levels only depend on the relation between I_{DUT} and the voltage bias where the signal is applied, for the given DC biases.

3. CASE STUDY: THE MOSFET

3.1. TIA-based MOSFET current converter

A simple example of a TIA-based circuit is the current measurement schematic for a n-channel MOSFET (Fig. 2). The output signal's (V_o) linearity is directly dependent on the region of operation (linear/saturation, weak/strong inversion) and the varying voltage. In case the signal is applied on the gate, G, only the input characteristics (I_d-V_g) should be accounted for, whereas only the output ones (I_d-V_d) if the drain, D, carries the signal.

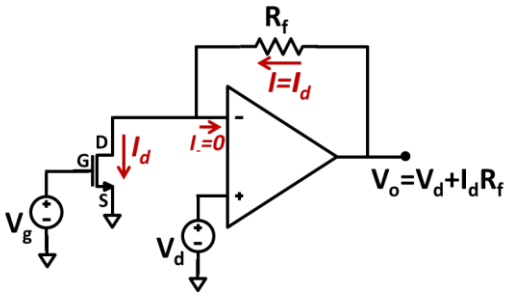


Figure 2: *Current-to-voltage converter configuration for MOSFET characterization (DC and noise)*

To understand the potential advantages for distortion, compared to a MOSFET amplifier, we underline that for the latter, the signal gain is also a function of the region of operation (linear/saturation) which affects the DC operating point and the load line. In fact, in order for the gain to be independent from the drain bias, the MOSFET in linear amplifiers is always biased in saturation, where -supposing a perfectly saturated current- the current is no more dependent on the drain voltage. In conclusion, for each DC bias and signal amplitude, one would obtain a specific set of harmonics, which only depends on the I-V transfer curves of the DUT and not in any other circuit elements, such as bias resistances as is the case in MOSFET amplifiers. This way, a high level of control and predictability over the generated distortion can be achieved, only by knowing the static (DC) behaviour of the electron device.

3.2. MOSFET modelling for simulations

In order to simulate the precise response of such a circuit, we need MOSFET models that are continuous and analytical from weak to strong inversion and from linear to saturation regimes. Thankfully, such a modelling approach exists and utilizes the Lambert-W (LW) function [12]-[13], which provides a very good description of the inversion charge behavior in all bias regions. This modelling approach is described by equations (5):

$$I_d = \frac{W}{L} \mu_{eff} \left[(q_s - q_d) + \frac{1}{2\eta k T C_{ox}} \left(q_s^2 - q_d^2 \right) \right] \tag{5a}$$

$$q_s = \eta k T C_{ox} L W \left(e^{(V_g - V_t)/\eta k T} \right) \tag{5b}$$

$$q_d = \eta kTC_{ox}LW\left(e^{(V_g - V_t - V_d)/\eta kT}\right) \quad (5c)$$

where W, L the width and length of the channel, μ_{eff} the effective mobility, V_t the threshold voltage, η the sub-threshold ideality factor (=1 for 60mV/dec), kT the thermal voltage (≅26mV at room temperature), and C_{ox} the oxide capacitance per unit area. The quantities q_s and q_d represent the carrier charge densities near the source, S, and drain, D, of the transistor, respectively. Fig. 3 shows an example of input and output MOSFET transfer curves as calculated by (5), with W = 1 μm, L = 0.2 μm, V_t = 0.5 V, μ_{eff} = 100cm^2/Vs, C_{ox} = 1.2μF/cm^2, and η = 1.

The typical behavior of a MOSFET's drain current, I_d, with gate and drain bias voltages can be seen:

1) exponential increase with V_g below V_t (weak inversion), whereas linear above V_t (strong inversion) for low V_d values and quadratic for $V_d > V_g - V_t$, whereas
2) linear increase with V_d for $V_d << V_g - V_t$ (linear regime), constant current (saturation) for $V_d > V_g - V_t$, and logarithmic behavior between the two (triode region).

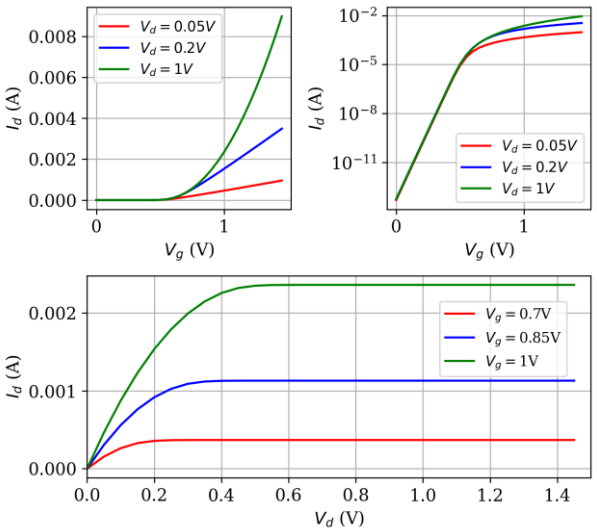


Figure 3: *Typical DC transfer characteristics for varying gate voltage (top/lin-log Y-scale) and drain voltage (bottom)*

Thanks to the simplicity of the proposed TIA-based circuit (Fig. 2), these curves are actually translated into voltage transfer functions (Fig. 4), because $V_0 = V_d + I_d R_f$. The R_f value is adapted so that the maximum voltage does not surpass the power supply of the OPAMP (+15V). The offset in high V_d values can be corrected by subtracting the DC bias of the drain from the output signal.

Consequently, depending on the DC bias around which we apply our signal, as well as the signal amplitude itself, we can expect very different results in terms of distortion. This is graphically demonstrated in Fig. 5, where an AC signal is applied at the MOSFET's gate, around 2 different DC bias (moderate and strong inversion) and with two different amplitudes. The first signal, oscillating around moderate inversion, is objected to the exponential dependence of $I_d(V_g)$, resulting in a signal with almost clipped bottom half and an exponentially distorted top half.

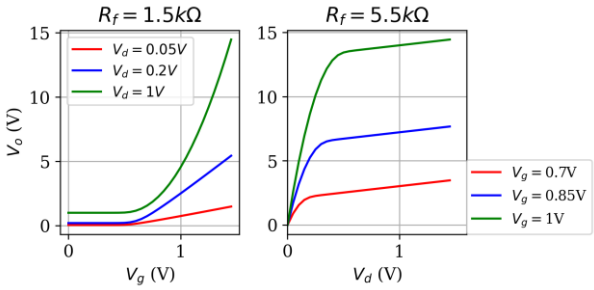


Figure 4: *Transfer characteristics of Fig.2's circuit with the I-V curves of Fig. 3.*

On the other hand, if we apply a signal around a V_g well above V_t, the signal has almost no distortion in linear region, while following a quadratic distortion in saturation (high V_d). It should be noted here that the signal amplitude was deliberately chosen to be significantly high, in order to capture the passage from weak to strong inversion and the quadratic dependence. If the signal amplitude is very small (mV range) and the DC bias is well above V_t, the distortion will be negligible.

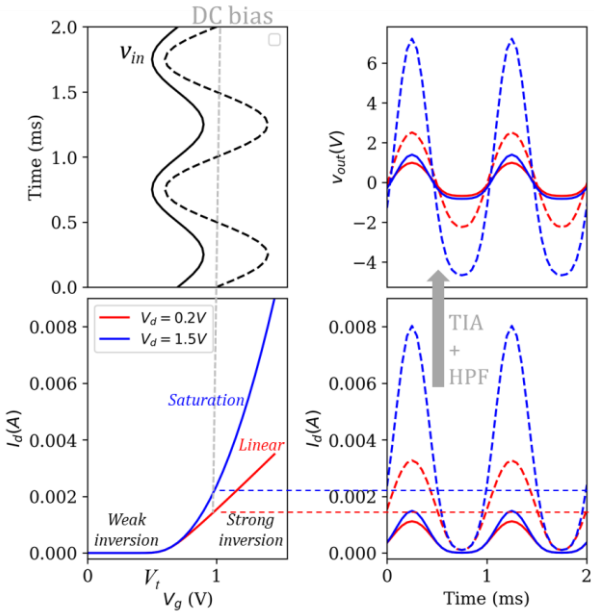


Figure 5: *Graphic demonstration of the distortion effect of the proposed circuit (Fig. 2), depending on the DC bias and AC amplitude of a signal applied at the transistor gate (TIA: Transimpedance amplifier, HPF: high-pass filter).*

3.3. Harmonic distortion analysis

Following the same simulation method, we examined many different scenarios of gate and drain DC bias and AC signal amplitudes, to reveal the variety of different harmonic content. Fig. 6 and Fig. 7 show two examples where both DC biases are kept constant at $V_g = V_d = 1$ V, but the signal (f = 1 kHz) is applied at the gate (Fig. 6) or at the drain (Fig. 7). The power spectra shown are obtained using the Welch periodogram method in SciPy (Python), whereas the signals have been normalized in amplitude so that they can be easily plotted and visualized together in the same

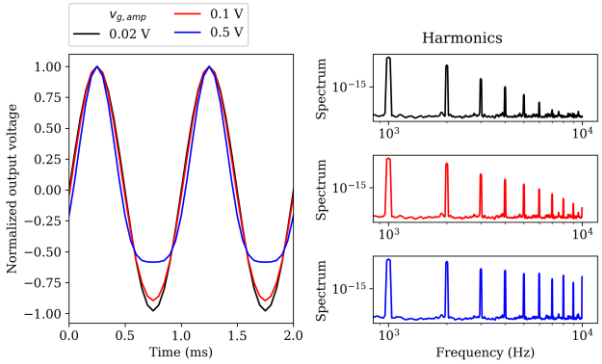


Figure 6: *Power spectrum calculations for the case where a signal is applied at the MOSFET gate for 3 different AC amplitudes. DC bias: V_g = V_d = 1 V.*

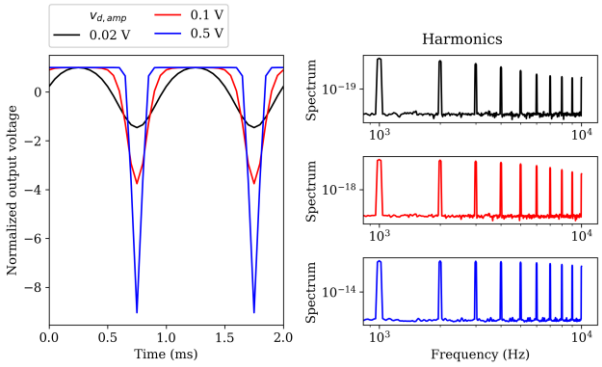


Figure 7: *Power spectrum calculations for the case where a signal is applied at the OPAMP's (+) input for 3 different AC amplitudes. DC bias: V_g = V_d = 1 V.*

graph; in any case, large amplitude differences can be compensated at the output thanks to R_f.

It becomes evident from Fig. 6 that when the signal amplitude is large enough to cover two different transistor operation regimes, it can dramatically affect the harmonic distortion. That's because in the case of varying V_g in saturation (Fig. 6), the signal is prone to the ~$(V_g$-$V_t)^2$ behavior of I_d, whereas varying V_d in strong inversion around 1 V (Fig. 7) can actually cause asymmetrical hard clipping of the signal due to the alternating between linear and triode regimes. As a result, the harmonic distortion in the latter case is much more severe, which is directly visible in the high amplitude of high-order harmonics.

We repeated this type of simulations for many combinations of DC biases V_g and V_d, from 0 up to 2 V, and various signal amplitudes, v_{amp}, from 50mV to 0.5V, and we extracted the power values of each harmonic in order to visualize their dependence with the applied voltages. Fig. 8 shows the relative (with regard to the fundamental's power) power of four harmonics (2nd to 5th) for v_{amp} = 0.5 V applied at the gate and Fig. 9 at the drain. For the case where the signal is around V_g (Fig. 8), a clear reduction of harmonic distortion with V_g is visible, whereas this reduction for the 2nd and 3rd harmonic can be cancelled out by high V_d bias. The 4th and 5th harmonic seem to have a negligible amplitude, except for the case of very low V_g values (weak inversion).

Concerning the case where the signal is applied at the drain (V_+ input of OPAMP), high V_d values can make the harmonic distortion immune to variations in V_g bias, and it is worth noting

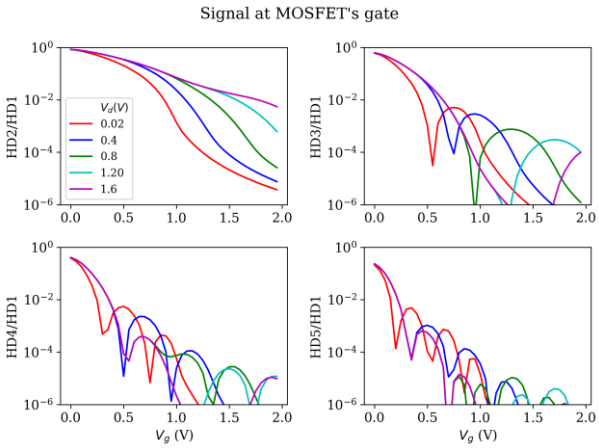


Figure 8: *Relative power levels of the 4 first harmonics for a signal of v_{amp} = 0.5 V amplitude applied at the gate, and various combinations of V_g and V_d DC bias.*

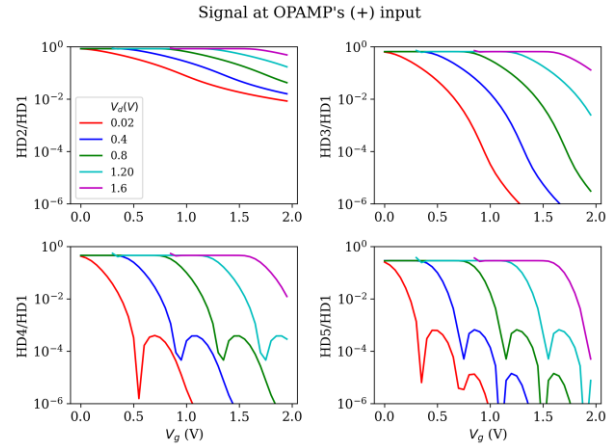


Figure 9: *Relative power levels of the 4 first harmonics for a signal of $v_{d,amp}$ = 0.5 V amplitude applied at (+), and various combinations of V_g and V_d DC bias.*

that here, even the 4th and 5th harmonics have very significant contributions.

By using formula (6), we also calculated the total harmonic distortion (THD) for various combinations of V_g and V_d, accounting up to the 5th harmonic.

$$\mathrm{THD}(\%) = 100\sqrt{\frac{\mathrm{HD2+HD3+HD4+HD5}}{\mathrm{HD1}}} \qquad (6)$$

where HDx the power of the xth harmonic. The results are plotted in Fig. 10, for v_{amp} = 0.2 V applied at the MOSFET's gate, and in Fig. 11 for the case where the signal is applied at the (+) input.

It is worth noting that in both cases (AC around V_g and around V_d), the THD reaches 100% for certain DC bias combinations, while it can also be decreased down to 0.1% for others. This is a direct confirmation of our hypothesis that with the circuit architecture of Fig. 2, any amount of THD can be achieved, provided that one examines all the possible bias conditions.

Moreover, the THD has a characteristic value for each combination of V_g, V_d and v_{amp}, making it easy to use this concept as the basis for a variety of audio distortion (or even tone control) applications. For example, as far as v_{amp} is concerned, Fig. 12

shows how it can impact THD when the signal is at the gate, for both linear (V_d = 0.1 V) and saturation (V_d = 2 V) regions. Similarly, in Fig. 13 is plotted the THD versus v_{amp} when the signal is at the (+) OPAMP's input, for both weak (V_g = 0.5 V) and strong (V_g = 1.5 V) inversion regions. As it can be seen, the THD is always increasing with v_{amp}, except when the DC bias is at weak inversion (V_g = 0.5 V), where THD reaches a plateau around v_{amp} = 0.2 V. Note also how there is no signal for V_d = 1.5 V in weak inversion, because the signal amplitude needed to reach triode region is higher than the maximum value of $v_{d,amp}$. Therefore the drain current is constant whatever the fluctuation of V_d.

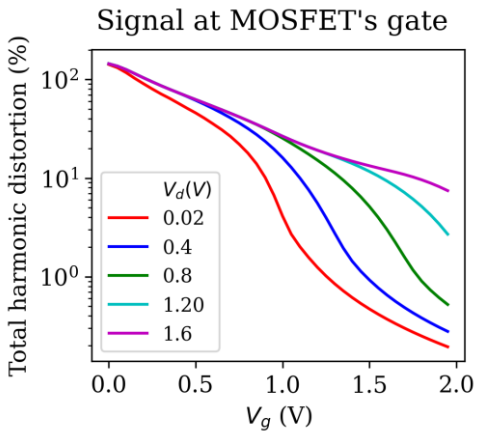


Figure 10: *THD for a signal of v_{amp} = 0.2 V amplitude applied at the gate, and various combinations of V_g and V_d DC bias.*

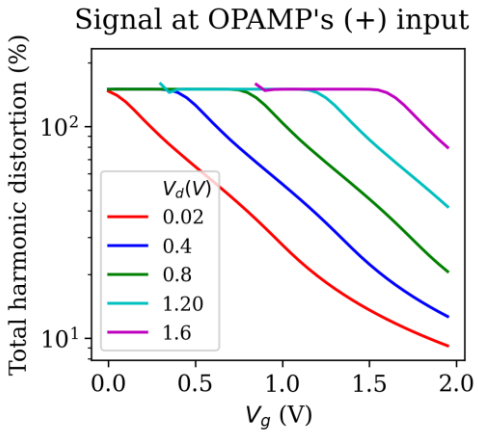


Figure 11: *THD for a signal of v_{amp} = 0.2 V amplitude applied at (+), and various combinations of V_g and V_d DC bias.*

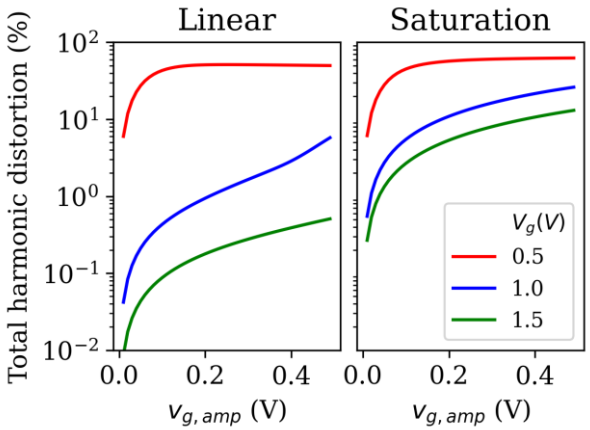


Figure 12: *THD versus varying signal amplitude v_{amp} applied at the MOSFET's gate, for linear (left) and saturation (right) regimes.*

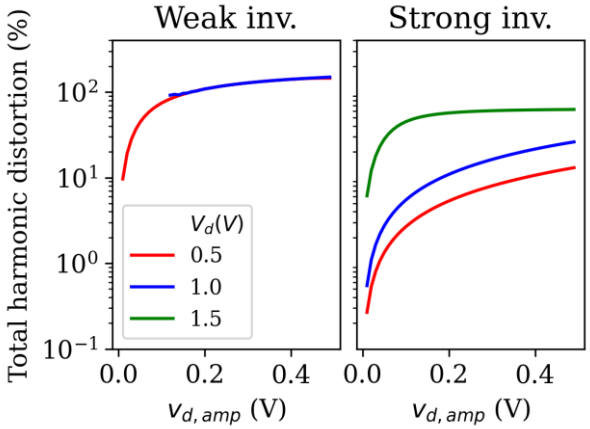


Figure 13: *THD versus varying signal amplitude v_{amp} applied at the OPAMP's (+) input, for weak (left) and strong (right) inversion regions.*

4. MEASUREMENT RESULTS

We constructed the circuit shown in Fig.2, using the Onsemi BS170 enhancement mode n-channel MOSFET as a device under test, which has a V_t around 2 V. Following the same methodology as in the simulations, we tested the output response for various V_g and V_d DC bias and signal amplitudes, with the signal applied at the MOSFET's gate. Some examples are shown in Fig. 14 (linear regime) and Fig. 15 (saturation regime). The value of Rf was modified for every case, in order to avoid OPAMP saturation or negligible output amplitude.

In Fig. 14 we note a verification of the bottom half soft clipping due to alternating from strong to weak inversion in linear region (as in Fig. 5), while in Fig. 15 the quadratic dependence of $I_d(V_g)$ in saturation regime gives a smooth distortion to the signal. For higher signal amplitudes, a more severe low-half clipping is visible, due to the passage to weak inversion (as in Fig. 6).

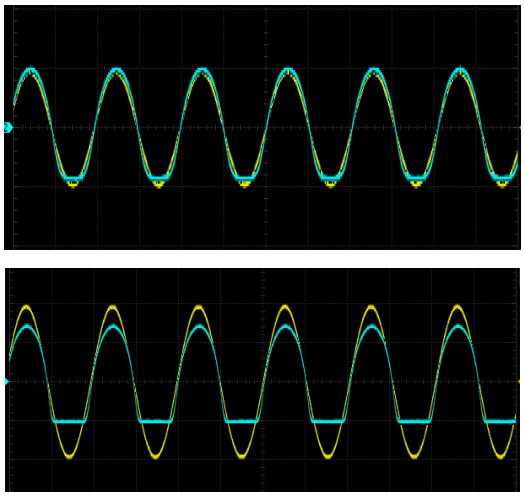


Figure 14: *Measured input (yellow) and output (blue) signals for V_d = 50 mV, V_g = 3 V, v_{amp} = 1 V (top) and 2 V (bottom) around gate voltage.*

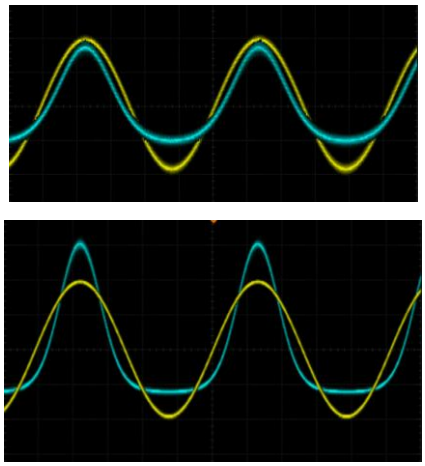


Figure 15: *Measured input (yellow) and output (blue) signals for V_d = 0.5 V, V_g = 2 V, v_{amp} = 0.1 V (top) and 0.2 V (bottom) around gate voltage.*

5. CONCLUSIONS

We have presented an analog circuit that can be used to exploit the non-linearities in any electron device, aiming the harmonic audio distortion. The n-channel MOSFET was shown and studied as an example, revealing a high level of control of the output signal harmonics through the DC gate and drain bias voltages, the signal's amplitude, as well as its input position (gate or drain). The effect was studied both through simulations and measurements. The proposed circuit configuration could be potentially used in guitar effect pedals or analog synthesizers.

6. REFERENCES

[1] T. Hamasaki, "Learn about harmonics by means of vacuum tube guitar amplifier," in *Proceedings of the AES International Conference*, 2013, pp. 98–102.

[2] K. Takemoto, S. Oshimo, and T. Hamasaki, "Supply voltage scaling technique of triode tube based on harmonic distortion characteristics," in *142nd Audio Engineering Society International Convention 2017, AES 2017*, 2017.

[3] R. C. D. Paiva, S. D'Angelo, J. Pakarinen, and V. Välimäki, "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 59, no. 10, 2012,DOI:10.1109/TCSII.2012.2213358.

[4] J. D. Reiss and A. McPherson, "Overdrive, Distortion, and Fuzz," in *Audio Effects*, CRC Press, 2014, pp. 182–203.

[5] J.-M. Réveillac, *Musical Sound Effects*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2018.

[6] D. J. Dailey, *Electronics for Guitarists*. Springer Cham, 2013.

[7] W. Sansen, "Distortion in elementary transistor circuits," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 46, no. 3, 1999,DOI:10.1109/82.754864.

[8] B. C. Paz *et al.*, "Non-linearity analysis of triple gate SOI nanowires MOSFETS," *SBMicro 2016 - 31st Symp. Microelectron. Technol. Devices Chip Mt. co-located 29th SBCCI - Circuits Syst. Des. 6th WCAS - IC Des. Cases, 1st INSCIT - Electron. Instrum. 16th SForum - Undergraduate-Stude*, pp. 3–6, 2016,DOI:10.1109/SBMicro.2016.7731355.

[9] A. Cerdeira, M. Estrada, R. Quintero, D. Flandre, A. Ortiz-Conde, and F. J. García Sánchez, "New method for determination of harmonic distortion in SOI FD transistors," *Solid. State. Electron.*, vol. 46, no. 1, pp. 103–108, 2002,DOI:10.1016/S0038-1101(01)00258-1.

[10] C. Beguni, A. M. Cailean, S. A. Avatamanitei, and M. DImian, "Photodiode Amplifier with Transimpedance and Differential Stages for Automotive Visible Light Applications," in *2020 15th International Conference on Development and Application Systems, DAS 2020 - Proceedings*, 2020.

[11] J. A. Chroboczek, A. Szewczyk, And G. Piantino, "Low Frequency Noise Point Probe Measurements On A Wafer Level Using A Novel Programmable Current Amplifier," in *Noise in Physical Systems and 1/f Fluctuations*, 2001, pp. 701–704.

[12] K. Papathanasiou *et al.*, "Symmetrical unified compact model of short-channel double-gate MOSFETs," *Solid. State. Electron.*, vol. 69, pp. 55–61, 2012.

[13] T. A. Karatsori *et al.*, "Full gate voltage range Lambert-function based methodology for FDSOI MOSFET parameter extraction," *Solid. State. Electron.*, vol. 111, 2015,DOI:10.1016/j.sse.2015.06.002.

ANTIALIASED STATE TRAJECTORY NEURAL NETWORKS FOR VIRTUAL ANALOG MODELING

Lasse Köper and Martin Holters

Department of Signal Processing and Communication
Helmut Schmidt University – University of the Federal Armed Forces
Hamburg Germany
lasse.koeper@hsu-hh.de | martin.holters@hsu-hh.de

ABSTRACT

In recent years, virtual analog modeling with neural networks experienced an increase in interest and popularity. Many different modeling approaches have been developed and successfully applied. In this paper we do not propose a novel model architecture, but rather address the problem of aliasing distortion introduced from nonlinearities of the modeled analog circuit. In particular, we propose to apply the general idea of antiderivative antialiasing to a state-trajectory network (STN). Applying antiderivative antialiasing to a stateful system in general leads to an integral of a multivariate function that can only be solved numerically, which is too costly for real-time application. However, an adapted STN can be trained to approximate the solution while being computationally efficient. It is shown that this approach can decrease aliasing distortion in the audioband significantly while only moderately oversampling the network in training and inference.

1. INTRODUCTION

In the realm of audio signal processing, nonlinear systems are widely used to create certain musical effects. The main class of these effects is comprised of clipping or overdrive effects, which add an amount of harmonic frequency content to the output, in order to achieve a distorted sound. Historically these kinds of effects were designed in the analog domain. The sound and behaviour of these analog devices are sought after until today. Consequently, there is a natural interest in recreating them in digital models, a process referred to as virtual analog modeling. Over the last decades many different approaches have been developed for converting the analog circuit into a virtual model. In [1, 2, 3], the authors show that Wave-Digital filters can be used to create a digital model of nonlinear stateful systems. [4] and [5] construct a discrete-time state-space model from circuit schematics, while [6] uses a Port-Hamiltonian formalism to create gueranteed-passive systems. Another approach, which in the recent years experienced an increase in popularity, is to use artificial neural networks. In this work we are using so called state-trajectory networks, which approximate the trajectory in the state-space using a feedforward neural network.

Since nonlinear systems introduce harmonic frequency content that can exceed the Nyquist frequency, all before-mentioned approaches can suffer from aliasing distortion. The most commonly used technique to reduce this aliasing distortion is to oversample the signal with a very high sampling rate. Obviously this increases computational complexity and memory consumption. In order to keep real-time capabilities as good as possible, we propose in this work an antialiased neural network model, which uses only modest oversampling.

This paper is organized as follows. In section 1.1 we give a detailed problem statement, followed by a derivation of the proposed method in section 2. Sections 3 and 4 discuss the neural network structure and training data generation. In section 5 the proposed method is applied to different example circuits, providing the results for this work, followed by some concluding remarks in section 6.

1.1. Problem statement

We consider an analog circuit which is described by an implicit ordinary differential equation

$$g\big(\dot{x}(t), y(t), x(t), u(t)\big) = 0 \tag{1}$$

where the state $x(t)$, the input $u(t)$, and the output $y(t)$ may be vector-valued. We assume an explicit solution

$$\dot{x}(t) = f_{\mathrm{x}}\big(x(t), u(t)\big) \tag{2a}$$

$$y(t) = f_{\mathrm{y}}\big(x(t), u(t)\big) \tag{2b}$$

exists, but does not have a closed form, i.e. may only be determined numerically from (1) using an iterative approach. The system can be transformed to discrete-time by various well-known methods such as the trapezoidal rule, resulting in an implicit update rule of the form

$$\bar{g}\big(\bar{x}(n), \bar{y}(n), \bar{x}(n-1), \bar{u}(n); T\big) = 0 \tag{3}$$

where T denotes the sampling interval, $\bar{u}(n) = u(nT)$ is the sampled input signal and $\bar{y}(n) \approx y(nT)$ approximates samples of the output signal subject to the error introduced by the discretization scheme. The states $\bar{x}(n)$ maintain a relationship to the continuous-time states $x(t)$, but do not necessarily correspond to samples thereof.

There are three problems associated with this approach:

1. Numerical solution of (3) is computationally expensive and may prevent real-time operation. (Note that using an explicit discretization scheme such as forward Euler will not help here due to the implicit nature of (1)).
2. The discretization scheme introduces an error which may become significant for high-frequency signals.
3. Even if the approximation error of the discretization scheme is tolerable, samples of $y(t)$ may not be the desired output: The continuous-time output may contain high-frequency

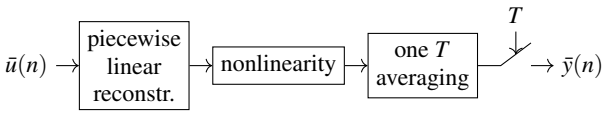


Figure 1: *Underlying idea of antiderivative antialiasing.*

content due to harmonics introduced by the nonlinearity of the system which results in aliasing distortion when sampled. Ideally, one would wish $\bar{y}(n)$ to be samples of a band-limited version of $y(t)$.

Problems 2 and 3 can be mitigated by oversampling the system, but this obviously aggravates problem 1. Problem 3 can also be approached by antiderivative antialiasing [7, 8, 9], but only for stateless and a very limited class of stateful systems [10, 11]. A recent trend to tackle problem 1 is to approximate the solutions

$$\bar{x}(n) = \bar{f}_{\mathrm{x}}\big(\bar{x}(n-1), \bar{u}(n)\big) \tag{4a}$$

$$\bar{y}(n) = \bar{f}_{\mathrm{y}}\big(\bar{x}(n-1), \bar{u}(n)\big) \tag{4b}$$

of (3)[1] (which lack a closed form) with neural networks [12] which are more efficient to evaluate than iterative numerical solution.

In this work, we explore the combination of the approximation with neural networks with the idea underlying the development of antiderivative antialiasing to attack all three problems simultaneously without imposing new restrictions on the considered systems.

2. ANTIALIASED NEURAL NETWORK APPROACH

Perfect aliasing suppression could be obtained by operating in the continuous-time domain. That is, for the samples $\bar{u}(n)$, the continuous input signal $u(t)$ could be formed by using an ideal reconstruction lowpass filter. Then, the continuous-time non-linear system of (2) could be applied to obtain $y(t)$. Bandlimiting to half the sampling-rate with an ideal lowpass filter before sampling would then allow to obtain an aliasing-free output $\bar{y}(n)$. While theoretically perfect, this is clearly impractical.

However, by using non-ideal lowpass filters, one may actually arrive at a practical system. In particular, consider linear interpolation for the reconstruction filter and averaging over one sampling interval for bandlimiting as depicted in figure 1. When implementing a digital system, the continuous signals obviously still pose a problem. The key insight of antiderivative antialiasing is that for a stateless nonlinearity, an equivalent system can be derived that operates solely on the sampled signals, but requires the antiderivative of the nonlinear mapping function—hence the method's name (see [7] for details.) Unfortunately, this only works for stateless nonlinear systems and for a limited class of stateful systems after some modification [10, 11].

But now consider a general stateful nonlinear system of the form (1). Focusing on the time interval from $(n-1)T$ to nT, we first observe that given both the state $x((n-1)T)$ at the beginning of that interval and the input $u(t)$ for the whole interval, the state trajectory $x(t)$ and the output $y(t)$ are fully determined during that interval. Now if $u(t)$ is assumed piecewise linear, i.e. linear in that interval, it in turn is fully determined by its values at the beginning and the end of the interval, namely its samples $\bar{u}(n-1)$ and $\bar{u}(n)$. So from these input samples and the initial state $\bar{x}(n-1) = x((n-1)T)$, the state trajectory $x(t)$ and the output $y(t)$ can be determined up to the interval's end nT. In particular, this allows to determine the state $\bar{x}(n) = x(nT)$ at the end of the interval to be used as the initial state for the subsequent time interval. Furthermore, we can apply the averaging operation to the output to obtain the antialiased output samples

$$\bar{y}(n) = \frac{1}{T}\int_{(n-1)T}^{nT} y(t). \tag{5}$$

So to summarize, we may conclude that input samples $\bar{u}(n-1)$ and $\bar{u}(n)$ and initial state $\bar{x}(n-1)$ are sufficient to determine the next state $\bar{x}(n)$ and the antialiased output $\bar{y}(n)$, or in other words, that there exist functions

$$\bar{x}(n) = \bar{f}_{\mathrm{x}}\big(\bar{x}(n-1), \bar{u}(n-1), \bar{u}(n)\big) \tag{6a}$$

$$\bar{y}(n) = \bar{f}_{\mathrm{y}}\big(\bar{x}(n-1), \bar{u}(n-1), \bar{u}(n)\big) \tag{6b}$$

that correspond to the approach from figure 1 applied to an arbitrary stateful system. However, these functions lack a closed form and in fact, evaluating them numerically requires a scheme like (3) operating at a sampling rate high enough that discretization error and aliasing distortion are sufficiently low. Thus, we are back to oversampling, but with deliberately simple interpolation and decimation filters.

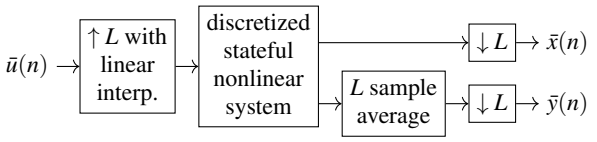


Figure 2: *Oversampled system to obtain training data for approximation with neural network.*

At this point, the neural network comes into play. Given that (6) describes a system with the desired properties except for the functions being computationally rather costly, it suggests itself to approximate them using a neural network. We may therefore boil down our approach to the following: Feed a suitable training stimulus $\bar{u}(n)$ to an oversampled, classical simulation approach to obtain corresponding sequences of $\bar{x}(n)$ and $\bar{y}(n)$ and use these as training data for a neural network to approximate (6). But note that the oversampling has to be of a particular form: Upsampling of the input uses linear interpolation, downsampling of the output uses averaging over one sampling interval. In contrast, downsampling of the states uses no decimation filter at all, as these act like snapshots of the oversampled system from which it could be restarted; i.e. it must be possible to reconstruct the original state trajectory, which would be impossible if the states were filtered. Thus, the system for generating training data finally looks as depicted in figure 2.

The simple interpolation and decimation filters are required to avoid additional states from appearing. Comparing (4) and (6), only $\bar{u}(n-1)$ needs to be newly introduced. It would be possible, however, to utilize more sophisticated filters as long as more samples of $\bar{u}$ are provided to fully cover the filters' combined support.

[1] Some modeling approaches yield an output equation dependent on $\bar{x}(n)$ instead of $\bar{x}(n-1)$, i.e. of the form $\bar{y}(n) = \bar{f}_{\mathrm{y}}^{*}\big(\bar{x}(n), \bar{u}(n)\big)$. It may not always be possible to express the solution of (3) like that, however. On the other hand, it is always possible to rewrite from $\bar{f}_{\mathrm{y}}^{*}$ to $\bar{f}_{\mathrm{y}}$ via $\bar{f}_{\mathrm{y}}\big(\bar{x}(n-1), \bar{u}(n)\big) = \bar{f}_{\mathrm{y}}^{*}\big(\bar{f}_{\mathrm{x}}\big(\bar{x}(n-1), \bar{u}(n)\big), \bar{u}(n)\big)$, so the case considered here is the more general one.

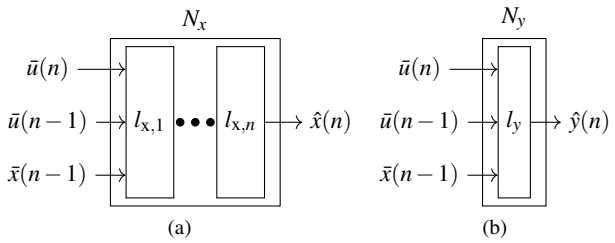


Figure 3: *Network structure during training, (a) trained on downsampled states, (b) trained on downsampled and averaged output.*

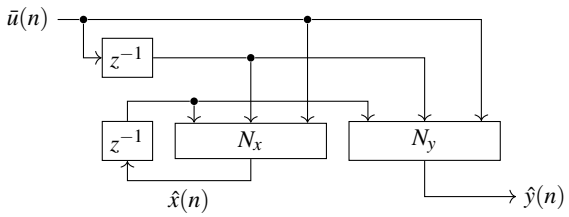


Figure 4: *Network structure during inference time.*

3. NEURAL NETWORK ARCHITECTURE

For approximating a stateful nonlinear system like (6), many suitable neural network modeling approaches can be found. To name a few, Wright et al. [13] employ recurrent neural networks in a black-box model approach, while in [14] time-varying effects are modeled with neural grey box models. Wilczek et al. [15] approximate the underlying nonlinear differential equation by a neural network and combine it with a numerical solver.

However, given that the system is in state-space form, it is preferable to apply the neural network directly in the state-space. Therefore using state-trajectory networks [12] seems to be the natural choice. Furthermore, this has the advantage that modeling of states and output can be performed in two different networks. This can be helpful to reduce stability issues during inference, since one network can be trained to solely model the dynamics of the system, whereas the other is only responsible for the nonlinear mapping from states to output.

Figure 3 shows the two models during training. The amount of hidden layers in the state predicting model can be adjusted to the complexity of the modeled analog circuit. In this work it was sufficient to use small networks with up to three hidden layers, which is also beneficial in terms of realtime capability. For the network predicting the output, it is even possible to use only one layer. Regarding the hidden layer type, we opted for fully connected layers followed by a hyperbolic tangent activation function. The last layer is a fully connected layer with no bias and linear activation.

All trainings were performed using a mean-squared error loss function and an NAdam optimizer following the hyperparameter settings of [16]. Figure 4 shows the model during inference. The networks N_x and N_y correspond to the two networks trained in figure 3 respectively. The predicted state vector $\hat{x}(n)$ is then used in the next time step as an input of N_x following the typical structure of a state-trajectory network. Similarly, by using the current and previous input sample, as well as the previously predicted state, the network N_y can approximate the averaging and downsampling operation from figure 2 employing a simple static mapping.

4. TRAINING DATA GENERATION

Generating suitable training data for the models is a crucial point in this work, since the antialiasing operation is in great part performed during training data generation. One of the first questions arising is whether to use measurement or simulation data. In this work it is obvious that we have to use simulation data, since the antialiasing lowpass filters in measurement hardware are in general not in the particular form required for the presented approach. Therefore the proposed antialiasing scheme could not be applied. The training data is generated by simulation using the ACME framework[2] in Julia[3]. This simulation is based on a state-space modeling approach, discretizing a continuous-time state-space system obtained from circuit dynamics using the trapezoidal rule [5].

We now use a low sampling rate f_{s1} to create our input signal $\bar{u}(n)$. This sampling rate should already be a reasonably high audio sampling rate, because this scheme is most effective if combined with modest oversampling. Therefore, we choose $f_{s1} = 96\,\text{kHz}$ for all examples in the following section. Afterwards the input signal is upsampled to a sampling rate f_{s2} like we saw in figure 2. Unless otherwise noted, we use $f_{s2} = L \cdot f_{s1}$ with $L = 4$, i.e. $f_{s2} = 384\,\text{kHz}$. The simulation is now run using the high sampling rate. After averaging the output, as well as downsampling states and output, the data is ready to be used for training. In order to evaluate the proposed method's antialiasing capabilities, we also create a reference training signal. This signal is obtained by just running the simulation at the low sampling rate f_{s1} with the signal $\bar{u}(n)$ as its input. The reference signal and the antialiased training data are the basis for all comparisons in the following examples. That means we compare the network structure from 3, trained with the antialiased data, against a standard STN using only the current input sample and trained with the reference signal.

As the training input stimulus, we use the exponential sine sweep

$$\bar{u}(n) = A \sin\left(\frac{\Omega_l\,(N-1)}{\log\frac{\Omega_h}{\Omega_l}} \exp\left(\frac{n}{N-1}\log\frac{\Omega_h}{\Omega_l}\right) - 1\right), \quad (7)$$

where N is the signal length in samples, Ω_l and Ω_h are the lower and upper normalized angular frequency $\Omega_{\cdot} = 2\pi\frac{f_{\cdot}}{f_s}$ respectively and A is the amplitude.

The signal length is chosen as $N = 960000$ and the lower and upper frequency as $f_l = 20\,\text{Hz}$ and $f_u = 2000\,\text{Hz}$, respectively. This stimulus is then repeated for different amplitudes A from a range differing between the examples as given in the next section.

5. APPLICATION

5.1. Example 1: 2nd Order Diode Clipper

As a first example we study the aliasing behavior of a second order diode clipper. The schematic can be seen in figure 5 and the corresponding component values are listed in table 1. For the training, the model structure from figure 3 was used with two

[2]Analog Circuit Modeling and Emulation for Julia (v0.10.0): https://github.com/HSU-ANT/ACME.jl

[3]The Julia Programming Language (v1.8.5): https://julialang.org/

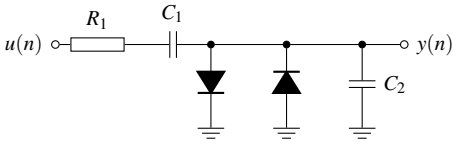


Figure 5: *Second-Order Diode Clipper.*

Table 1: *Second-Order Diode Clipper - Component List.*

Element	Value
R_1	2.2 kΩ
C_1	470 nF
C_2	10 nF
Diodes	$I_s = 1\,\text{pA}$, $v_t = 25\,\text{mV}$, $\eta = 1$

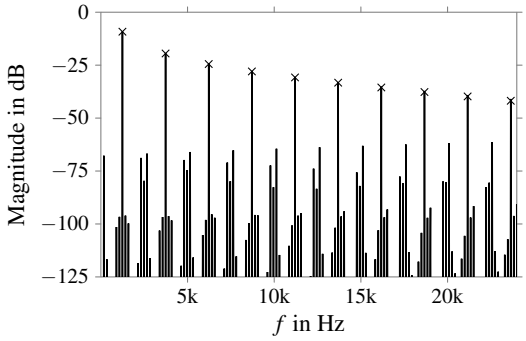


Figure 6: *Second-order diode clipper - reference model without antialiasing, crosses mark the desired harmonics.*

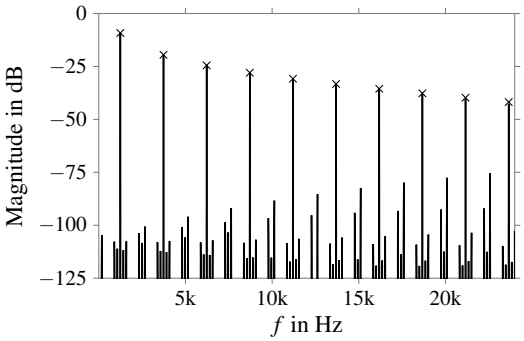


Figure 7: *Second-order diode clipper - model with antialiasing L=4, crosses mark the desired harmonics.*

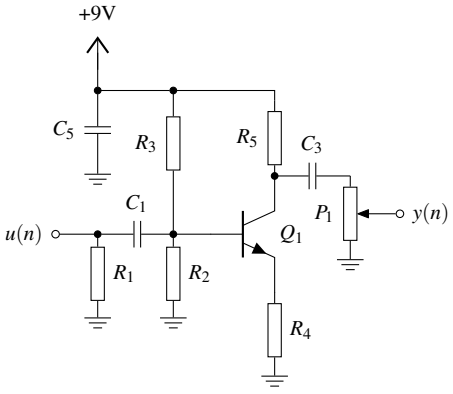


Figure 8: *Birdie Treble Booster.*

hidden layers, each comprising 12 neurons and a hyperbolic tangent activation function. The output model from figure 3 uses one hidden layer with 8 neurons.

In order to train the model on a variety of signal amplitudes, the sinesweep from (7) was evaluated for a set of different amplitudes and combined afterwards to form the final training signal. With the goal of achieving a high amount of aliasing to show the capabilities of the proposed method, we opted for rather high amplitudes in the training signal, ranging from 0.2 V to 12 V. The model was trained over 40 epochs after which the loss did not decrease significantly further. The batch size was set to 1024 samples. To test the aliasing reduction of the proposed model, a single sinusoid with an amplitude of 10 V and a frequency of 1244.5 Hz was applied to the reference(no antialiasing) and the antialiased model. Figure 6 shows the frequency content of the test signal, after being applied to the reference model. As expected we can observe a high amount of aliasing distortion between the desired harmonics. In comparison, the output of the proposed model in figure 7 shows a high reduction of aliasing in the audio band. The aliasing components for higher frequencies experience a much smaller reduction than for the low frequencies. However, this is a well-known property of the antiderivative antialiasing approach from [7] and was to be expected, since the proposed method is based on it.

It should be noted that the diode clipper is simple enough so that it could also be treated with the antialiasing approach of [10, 11]. Indeed, applying that approach yields very similar results. Furthermore, when based on a lookup table, it is more straight-forward to design than training a neural network and computationally cheaper at runtime. So the diode clipper serves as a proof-of-concept, but does not exhibit the advantage of the present method. That will be different for the following cases, where the method of [10, 11] is no longer applicable.

5.2. Example 2: Birdie - Treble Booster

Our next case study is the guitar treble booster *The Birdie*. It is sold as a DIY soldering kit by *musikding* [4] and is based on the Electro-Harmonix *Screaming Bird*. Figure 8 shows the schematic and table 2 the corresponding component values of the circuit. The circuit itself is based on a common emitter amplifier, with a simple highpass filter comprising R_1, C_1 and R_2 at the input. Note that the capacitor C_5 is only used to stabilize the supply voltage. However, we assume an ideal voltage source for the supply and can therefore safely omit C_5 from the simulation, reducing the circuit's dynamics to second order. We will also model the circuit at a fixed level of the volume potentiometer P_1 of 0.5.

Like in the previous example, we go for a model with two fully connected hidden layers and 12 neurons each. The training signal from (7) is adapted to a different set of amplitudes ranging from 0.2 to 8 volts. The model was trained over 30 epochs with a batch size of 1024 samples. As a test signal for inference, we use again a single sinusoid with a frequency of 1244.5Hz but with a slightly smaller amplitude of 8 volts. The frequency domain

[4] The Birdie: https://www.musikding.de/docs/musikding/birdie/birdie_schalt.pdf

Table 2: *Birdie Treble Booster - Component List.*

Element	Value
R_1	1 MΩ
R_2	43 kΩ
R_3	430 kΩ
R_4	390 Ω
R_5	10 kΩ
C_1	2.2 nF
C_3	2.2 nF
C_5	100 µF
Q_1	2N5088
P_1	100 kΩ log

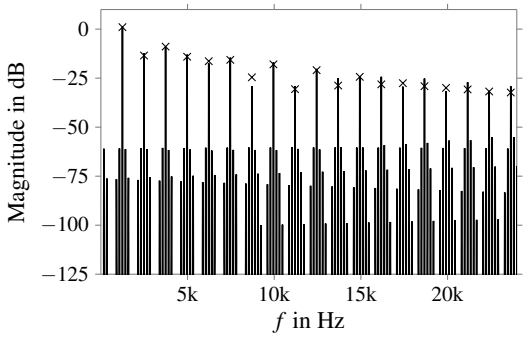


Figure 9: *Birdie - reference model without antialiasing, crosses mark the desired harmonics.*

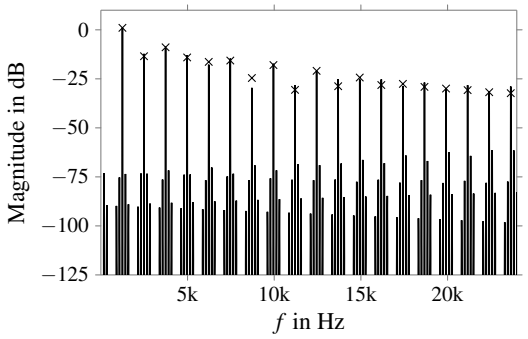


Figure 10: *Birdie - model with antialiasing L=4, crosses mark the desired harmonics.*

output to this test signal can be seen in figure 9 (reference) and in figure 10 (antialiased model). We can observe a similar behavior as for the diode clipper. The reference shows a decent amount of aliasing distortion between the desired harmonics of the output signal, whereas the antialiased model shows only a small amount of aliasing distortion. For low frequencies the aliasing components in figure 10 are barely visible, but they increase for higher frequencies.

5.3. Example 3: Fuzzface

To conclude this section we show the effectiveness of the proposed method on the *Fuzzface* guitar distortion effect. This circuit is a useful addition to the previous examples, because in comparison to the treble booster from the last section it provides a lot more

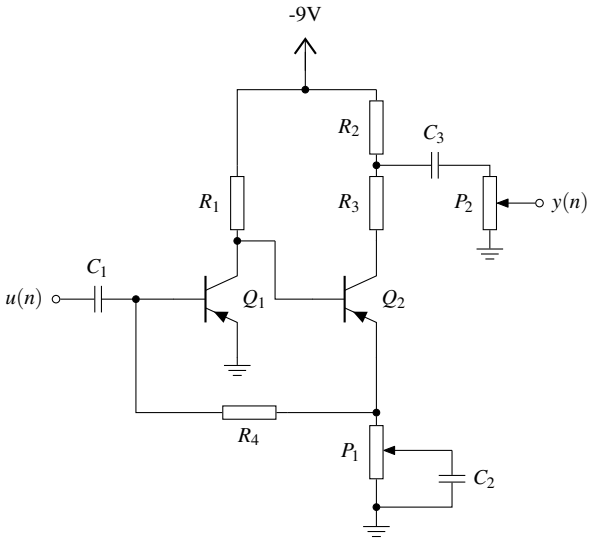


Figure 11: *Fuzzface.*

Table 3: *Fuzzface - Component List.*

Element	Value
R_1	33 kΩ
R_2	470 Ω
R_3	8.2 kΩ
R_4	100 kΩ
C_1	2.2 µF
C_2	20 µF
C_3	100 nF
Q_1, Q_2	AC128
P_1	1 kΩ linear
P_2	500 kΩ log

distortion to the output. Consequently it will produce more aliasing distortion, due to the high amount of harmonic frequency content.

The schematic of the *Fuzzface* can be seen in figure figure 11 and the component values can be found in table 3. The circuit basically consists of an input stage providing a high voltage gain, an output stage and a feedback loop to stabilize the circuit. The amount of distortion added to the input can be adjusted with the potentiometer P_1, which controls the amount of negative feedback via R_4. The output volume can be controlled with the remaining potentiometer P_2. For the sake of simplicity we train the models of the system with fixed values of 0.5 for both potentiometers.

Since this circuit is more complex with respect to the number of states and nonlinear behavior than the previous examples, it is necessary to adjust the simple STN from figure 3 in order to reduce exposure bias and ensure stability. For this reason the network is split into three separate sub-networks, where each network is trained to solely predict one of the three states. This allows a more precise prediction of each state. During inference the three networks are connected in parallel and each predicted state is fed to each individual network as the next input sample. The output network remains the same as in the previous examples. Note that the modification of the network does not effect the proposed antialiasing approach and is only necessary to obtain a stable model during inference.

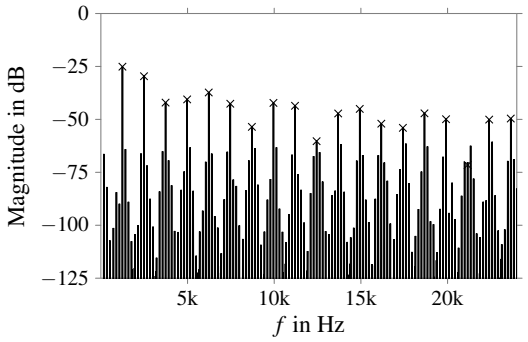


Figure 12: *Fuzzface - reference model without antialiasing, crosses mark the desired harmonics.*

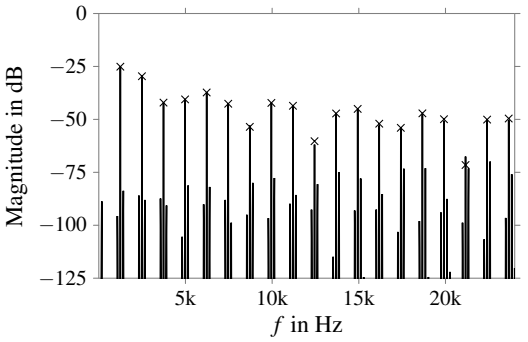


Figure 13: *Fuzzface - model with antialiasing L=4, crosses mark the desired harmonics.*

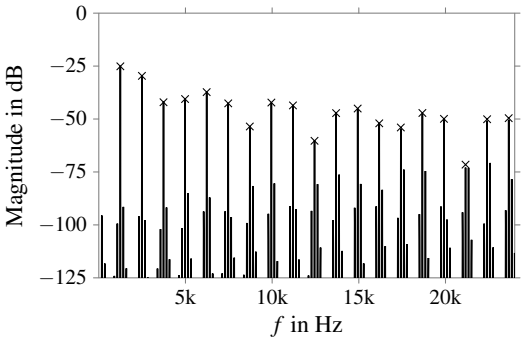


Figure 14: *Fuzzface - model with antialiasing L=8, crosses mark the desired harmonics.*

Regarding the model structure of each individual network, we use two hidden layers with 12 neurons each. For this example, we opted to normalize the states before training. This was necessary because when using the capacitor's voltages as states for the model, they become very difficult to train. The reason for this is that the bias voltage at the capacitors differ a lot from each other. Furthermore the amplitude around this bias values is fairly small. Training the network with these physically meaningful states resulted into a bad model performance. Consequently the states were normalized to

$$x_{i,norm} = \frac{x_i - \mu_{x_i}}{\max\left(x_i - \mu_{x_i}\right)}, \tag{8}$$

where x_i are the individual states and μ_{x_i} are the mean values for each state.

The models for this circuit were trained over 40 epochs using the same training sinesweep as before, but with adapted amplitudes in the more reasonable range of 0.2 to 3 volts. Figure 12 shows the frequency content of the reference model, when excited with the signal $u(n) = 2\sin\left(2\pi \cdot 1244.5\frac{n}{f_s}\right)$. Between the desired harmonics a lot of aliasing components are present. In comparison the antialiased model in figure 13 can reduce these aliasing distortion by a large margin using only an oversampling factor of $L = 4$. Using a higher oversampling factor than $L = 4$ for this model does not result into a significant increase in aliasing reduction. Figure 14 shows the frequency content for $L = 8$. It can be seen that the improvement in comparison to figure 13 is marginally small. However, since the model uses the downsampled data there is no computational overhead when running the model trained with $L = 8$ or higher.

6. CONCLUSION AND OUTLOOK

In this work, we presented an antialiasing approach for state-trajectory networks. This approach uses the general idea of antiderivative antialiasing [7, 8, 9, 10, 11] and applies an approximation of the method to the model's training data. In contrast to the original antiderivative antialiasing, the proposed method is not limited to systems having only one nonlinearity with scalar input. The necessary modification of the STN in order to incorporate the method is straight-forward and easily implemented. Only one additional input with the previous input sample has to be added.

The method was successfully applied to three different nonlinear circuits. In comparison to a reference model, which was sampled with a sampling rate of 96 kHz, the antialiased model was able to attenuate most of the aliasing distortion while using the same sampling rate. Consequently the proposed method can reduce aliasing with only the cost of one additional input. The aliasing reduction works especially well for lower frequencies. Although high frequency aliasing components receive a smaller attenuation it is still an improvement compared to the reference model.

Generally speaking the proposed method is not only applicable to STNs, but basically to all neural virtual analog modeling approaches. The implementation should be straight-forward, since most of the modifications happen during training data generation. For the neural networks only minor adjustments should be necessary.

Another possible extension is the use of more complex filters for the up- and downsampling. E.g. one could replace the rectangular filter kernel of the decimation lowpass with a triangular one, just as was also done in [7] and then add $\bar{u}(n-2)$ as an additional input to the neural networks. But it is also possible to use more complex interpolation filters, which is problematic in antiderivative antialiasing. The only constraint is that the input samples provided to the neural network have to cover the combined support of interpolation and decimation filter. In practice, it may even be possible to violate this constraint and provide fewer input samples than theoretically needed.

In fact, preliminary simulations have shown that for many applications good results could be achieved by using only the current input sample $\bar{u}(n)$. This makes sense, because it basically reduces the linear interpolation filter to a zero-order hold. This approximation is acceptable if the change in amplitude between two adjacent

input samples is sufficiently small. Exploring this design space is left to future work.

7. REFERENCES

[1] K.J. Werner, *Virtual Analog Modeling of Audio Circuits using Wave Digital Filters*, Ph.D. thesis, 2016.

[2] K.J. Werner, E.J. Tebout, S. Cluett, and E. Azelborn, "Modeling and extending the RCA Mark II sound effects filter," in *Proc. 25th Int. Conf. on Digital Audio Effects (DAFx-20in22)*, Vienna, Austria, 2022.

[3] D.T. Yeh and J.O. Smith, "Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations," in *Proc. 11th Int. Conf. on Digital Audio Effects (DAFx-08)*, Espoo, Finland, 2008.

[4] D.T. Yeh, J.S. Abel, and J.O. Smith, "Automated physical modeling of nonlinear audio circuits for realtime audio effects – part I: Theoretical development," *IEEE Trans. Audio, Speech and Language Process.*, vol. 18, no. 4, pp. 728–237, 2010.

[5] M. Holters and U. Zölzer, "A generalized method for the derivation of non-linear state-space models from circuit schematics," in *Proc. European Signal Processing Conference (EUSIPCO-15)*, 2015, pp. 1073–1077.

[6] M. Danish, S. Bilbao, and M. Ducceschi, "Applications of port hamiltonian methods to non-iterative stable simulations of the Korg35 and Moog 4-pole VCF," in *Proc. 24th Int. Conf. on Digital Audio Effects (DAFx20in21)*, Vienna, Austria, 2021.

[7] J.D. Parker, V. Zavalishin, and E. Le Bivic, "Reducing the aliasing of nonlinear waveshaping using continuous-time convolution," in *Proc. 19th Int. Conf. on Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, 2016, pp. 137–144.

[8] S. Bilbao, F. Esqueda, J.D. Parker, and V. Välimäki, "Antiderivative antialiasing for memoryless nonlinearities," *IEEE Signal Process. Lett.*, vol. 24, no. 7, pp. 1049–1053, 2017.

[9] S. Bilbao, F. Esqueda, and V. Välimäki, "Antiderivative antialiasing, lagrange interpolation and spectral flatness," in *2017 IEEE Workshop on Appl. of Signal Process. to Audio and Acoust. (WASPAA)*, New Paltz, NY, USA, 2017, pp. 141–145.

[10] M. Holters, "Antiderivative antialiasing for stateful systems," in *Proc. 22nd Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019.

[11] M. Holters, "Antiderivative antialiasing for stateful systems," *Appl. Sciences*, vol. 10, no. 1, 2020.

[12] J.D. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. 22nd Int. Conf. on Digital Audio Effects (DAFx-19)*, Birmingham, UK, 2019.

[13] A Wright, E.P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. 22nd Int. Conf. on Digital Audio Effects (DAFx-19)*, Birmingham, UK, 2019.

[14] A. Wright and V. Välimäki, "Neural modelling of periodically modulated time-varying effects," in *Proc. 23rd Int. Conf. on Digital Audio Effects (DAFx-20)*, Vienna, Austria, 2020.

[15] J. Wilczek, A. Wright, V. Välimäki, and E.A.P. Habets, "Virtual analog modeling of distortion circuits using neural ordinary differential equations," in *Proc. 25th Int. Conf. on Digital Audio Effects (DAFx20in22)*, Vienna, Austria, 2022.

[16] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. 4th Int. Conf. Learn. Repres. (ICLR 2016)*, San Juan, Puerto Rico, 2016.

EXPLICIT VECTOR WAVE DIGITAL FILTER MODELING OF CIRCUITS WITH A SINGLE BIPOLAR JUNCTION TRANSISTOR

Oliviero Massi, Riccardo Giampiccolo, Alberto Bernardini, and Augusto Sarti

Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Piazza L. Da Vinci 32, 20133 Milano, Italy
oliviero.massi@polimi.it | riccardo.giampiccolo@polimi.it | alberto.bernardini@polimi.it
augusto.sarti@polimi.it

ABSTRACT

The recently developed extension of Wave Digital Filters based on vector wave variables has broadened the class of circuits with linear two-port elements that can be modeled in a modular and explicit fashion in the Wave Digital (WD) domain. In this paper, we apply the vector definition of wave variables to nonlinear two-port elements. In particular, we present two vector WD models of a Bipolar Junction Transistor (BJT) using characteristic equations derived from an extended Ebers-Moll model. One, implicit, is based on a modified Newton-Raphson method; the other, explicit, is based on a neural network trained in the WD domain and it is shown to allow fully explicit implementation of circuits with a single BJT, which can be executed in real time.

1. INTRODUCTION

Virtual Analog (VA) modeling [1] is an audio signal processing field of research which focuses on the digital emulation of analog audio equipment. Over recent years, a lot of research effort has been dedicated to the faithful and efficient digital implementation of circuit nonlinearities, which concur to the well appreciated timbral characteristics of analog audio gear. Forming the core of countless amplifier models used in a broad range of audio equipment, Bipolar Junction Transistors (BJTs) are arguably the most relevant components in this regard.

In the literature, VA modeling approaches can be generally divided into two categories: *black-box* approaches that infer a global model of a reference circuit relying on pairs of observed input/output data using, e.g., Volterra series [2] or neural networks [3], and *white-box* approaches that emulate the reference circuit by simulating the corresponding system of ordinary differential equations, e.g., using state-space methods [4], the port-Hamiltonian method [5], or Wave Digital Filters (WDFs) [6].

Among white-box techniques, WDFs have proved to be a very promising framework for creating digital models of reference analog circuits. Developed in the 70s by A. Fettweis to derive digital implementations of passive analog filters [7], WDFs rely on a port-wise linear mapping of Kirchhoff pairs of variables (voltage and current) into pairs of wave variables (incident and reflected) with the introduction of a scalar free parameter per port called *port resistance*. Circuit elements and topological connection networks are modeled separately, in a modular fashion. The reference circuit is represented in the Wave Digital (WD) domain as an interconnection of one-port and multi-port WD blocks, characterized by implicit scattering relations between wave variables, called *delay-free loops*. By proper choice of port resistances (through the so-called *adaptation* process) and making use of stable discretization methods [8] (e.g., trapezoidal rule), circuits containing linear one-port elements can be implemented in a fully explicit fashion, i.e., removing all delay-free loops [7]. Furthermore, a WD structure which relies on a scalar definition of wave variables is able to accommodate a single nonlinear one-port circuit element by placing it at the root of a tree-like structure and connecting it to an adapted (reflection-free) port of a WD junction [9, 10]. In order to design an explicit WDF, since nonlinear one-port elements cannot be generally adapted, the reflection-free port of the junction is necessary, otherwise the instantaneous wave reflection from the junction back to the nonlinearity would result in a delay-free loop [9, 10]. Such a WDF design methodology, originally conceived for static nonlinearities, has been further extended to reference circuits with a single nonlinear one-port element with memory [11, 12].

The previous considerations on traditional WDFs with a single nonlinear one-port element are not directly applicable to WDFs with a single nonlinear multi-port element. In fact, in that case, computability issues might arise due to unavoidable delay-free loops. More in general, when both ports of a generic two-port element are connected to the same multi-port junction, as in Fig. 1, a double delay-free loop passing through the element and the junction always arises, and it cannot be eliminated even in the case in which both ports of the junction are locally made reflection-free [13, 14, 15].

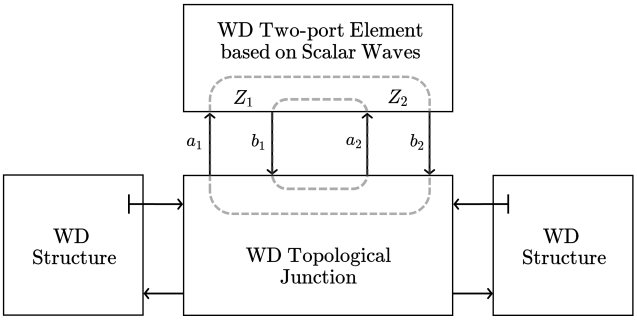


Figure 1: *Example of traditional WDF based on a scalar definition of waves that includes a WD two-port element. The dashed circles indicate two delay-free loops that arise in the WDF.*

Several works in the literature of WDFs are devoted to the modeling of multi-port nonlinearities and specifically to the modeling of BJTs. Werner *et al*. [16] proposed a hybrid Kirchhoff/Wave approach based on the K-method [4, 17, 18] to solve circuits with multiple/multi-port nonlinearities that are grouped at the root of the digital structure. Olsen *et al*. [19] used the Newton-Raphson (NR) method to speed-up the computation of the same digital structures described in [16]. In [20], *et al*. defined a general WD model for 3-terminal devices, whose number of ports can vary from 1 to 6, along with explicit WD realizations of MOSFET and JFET transistors and an implicit WD realization of BJTs based on the Ebers-Moll model and implemented using a robust modified Newton-Raphson solver. Kolonko *et al*. [21] introduced a modified Ebers-Moll model where the forward and reverse conducting diodes are considered individually and then connected through a suitable WD junction.

All the WD models of BJT transistors presented in the literature are based on scalar definitions of port variables and this often causes unavoidable delay-free loops when two-port models are employed. The recently introduced Vector WDFs [22] generalized the scalar port-wise definition of wave variables in linear two-port elements to a vector definition. Eventually, this allowed to overcome several computability issues, such as delay-free loops that are formed when connecting a two-port to a topological junction.

In this paper, we apply Vector WDFs to the implementation of circuits containing a single two-port WD model of a BJT. The nonlinear two-port element can be accomodated at the root of the WDF, which allows to implement explicit WD structures preserving the modularity of Fettweis' traditional WDFs [7]. We first propose an extension of the Ebers-Moll model [23] of a BJT and we present two vector wave-based WD realizations. The first one is implicit and it relies on a modified Newton-Raphson method [20] to solve the characteristic equation derived from the extended Ebers-Moll model. The second one is explicit and it is characterized by a Multi-Layer Perceptron network trained in the WD domain; this accompanies a growing trend in the WDF literature [24, 25, 26], in which data-driven neural models of nonlinear devices are connected to "traditional" WD blocks.

The remainder of this manuscript is organized as follows. In Section 2, the design of WDFs containing a single nonlinear two-port element is discussed. Section 3 introduces the two proposed BJT models based on vector waves. In Section 4, the developed methods are applied for the emulation of the input stage of a guitar distortion pedal. Conclusions are drawn in Section 5.

2. VECTOR WAVE DIGITAL FILTERS

2.1. Scalar Waves

The design of WDFs is based on a port-wise description of a reference analog circuit. Circuit elements and topological connection networks are modeled using one- or multi-port WD blocks characterized by scattering relations. In traditional WDFs based on voltage waves, each pair of Kirchhoff variables at a generic port j of a circuit element, i.e., the port voltage v_j and the port current i_j, is substituted with a pair of scalar WD variables defined as [7]

$$a_j = v_j + Z_{jj} i_j \quad , \quad b_j = v_j - Z_{jj} i_j \,, \tag{1}$$

where a_j is the incident wave, b_j is the reflected wave and $Z_{jj} \neq 0$ is a free real-valued parameter, usually called *reference port resistance* and here renamed as *reference one-port resistance*. This free parameter is set to *adapt* linear one-port circuit elements, thus obtaining explicit WD scattering relations in the discrete-time domain in which the reflected wave does not instantaneously depend on the incident wave [7, 8].

N-port topological connection networks, characterized by a vector of port voltages $\mathbf{v}_\mathrm{J} = [v_{\mathrm{J}1}, \ldots, v_{\mathrm{J}N}]^T$ and a vector of port currents $\mathbf{i}_\mathrm{J} = [i_{\mathrm{J}1}, \ldots, i_{\mathrm{J}N}]^T$, are modeled in the WD domain using N-port junctions characterized by the wave variables

$$\mathbf{a}_\mathrm{J} = \mathbf{v}_\mathrm{J} + \mathbf{Z}_\mathrm{J} \mathbf{i}_\mathrm{J} \quad , \quad \mathbf{b}_\mathrm{J} = \mathbf{v}_\mathrm{J} - \mathbf{Z}_\mathrm{J} \mathbf{i}_\mathrm{J} \,, \tag{2}$$

where $\mathbf{a}_\mathrm{J} = [a_{\mathrm{J}1}, \ldots, a_{\mathrm{J}N}]^T$ is the vector of waves incident to the junction, $\mathbf{b}_\mathrm{J} = [b_{\mathrm{J}1}, \ldots, b_{\mathrm{J}N}]^T$ is the vector of waves reflected by the junction, while $\mathbf{Z}_\mathrm{J} = \mathrm{diag}\,[Z_1, \ldots, Z_N]$ is a diagonal matrix having one-port resistances as diagonal entries. The relation between $\mathbf{a}_\mathrm{J}$ and $\mathbf{b}_\mathrm{J}$ is

$$\mathbf{b}_\mathrm{J} = \mathbf{S} \mathbf{a}_\mathrm{J} \,, \tag{3}$$

where $\mathbf{S}$ is a $N \times N$ scattering matrix. General formulas for computing the scattering matrix of arbitrary reciprocal or non-reciprocal connection networks in the WD domain are discussed in [27, 15].

When two-port circuit elements are present, a WDF structure based on scalar port-wise wave definition is generally affected by computability problems. In Fig. 1, a generic WDF structure based on scalar wave variables includes a (linear or nonlinear) two-port element whose ports are both connected to the same topological junction. As highlighted by the dashed paths, two delay-free loops involving cross-dependencies between wave variables are unavoidably formed in the WDF: they cannot be eliminated through any choice of the free parameters.

2.2. Vector Waves

With the purpose of encompassing both ports of the same two-port element, we introduce the following *vector definition of wave variables* [22], which generalizes (1):

$$\begin{aligned} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} &= \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} \\ \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} &= \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} - \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} . \end{aligned} \tag{4}$$

$[v_1, v_2]^T$ is the vector of the two port voltages, $[i_1, i_2]^T$ is the vector of the two port currents, $[a_1, a_2]^T$ is the vector of the waves incident to the two-port element, $[b_1, b_2]^T$ is the vector of the waves reflected by the two-port element and

$$\mathbf{Z}_{1,2} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \tag{5}$$

is a full-rank 2×2 matrix of real free parameters $Z_{\imath\jmath}$, with $\imath \in 1, 2$ and $\jmath \in 1, 2$, which we refer to as *reference two-port resistance*. Since $\mathbf{Z}_{1,2}$ is full-rank, we have

$$|\mathbf{Z}_{1,2}| = \det\left[\mathbf{Z}_{1,2}\right] = Z_{11} Z_{22} - Z_{12} Z_{21} \neq 0 \,. \tag{6}$$

The inverse mapping from WD variables to Kirchhoff variables can be expressed as

$$\begin{aligned} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \frac{1}{2} \left(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right) \\ \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} &= \frac{1}{2|\mathbf{Z}_{1,2}|} \begin{bmatrix} Z_{22} & -Z_{12} \\ -Z_{21} & Z_{11} \end{bmatrix} \left(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right) . \end{aligned} \tag{7}$$

Analogously to one-port linear elements, a two-port WD element based on the proposed vector definition of waves (4) can be adapted through the assignment of $\mathbf{Z}_{1,2}$, which eliminates the instantaneous dependency between the vector of reflected waves $[b_1, b_2]^T$ from the vector of incident waves $[a_1, a_2]^T$ in the scattering relation .

The approaches described for the WD modeling of topological connection networks can be generalized for the design of topological multi-port WD junctions based on mixed scalar and vector definitions of waves [22]. An N-port topological junction which is connected to other WD blocks through P two-port connections and $N - 2P$ one-port connections is characterized by the same wave variable definition in (2), where $\mathbf{Z}_{\mathrm{J}}$ is in this case a *full-rank block-diagonal matrix* of free parameters (not simply a diagonal matrix as in WDFs solely based on scalar waves). Assuming, without loss of generality, to number the P two-port connections before the $N - 2P$ one-port connections, $\mathbf{Z}_{\mathrm{J}}$ can be written as

$$\mathbf{Z}_{\mathrm{J}} = \begin{bmatrix} \mathbf{Z}_{1,2} & \dots & \mathbf{0} & [0,0]^T & \dots & [0,0]^T \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{Z}_{2P-1,2P} & [0,0]^T & \dots & [0,0]^T \\ [0,0] & \dots & [0,0] & Z_{2P+1} & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ [0,0] & \dots & [0,0] & 0 & \dots & Z_N \end{bmatrix} \tag{8}$$

where $\mathbf{Z}_{1,2}, \dots, \mathbf{Z}_{2P-1,2P}$ are 2×2 full-rank submatrices and $Z_{2P+1}, \dots, Z_N$ are scalar parameters different from zero.

The resulting WD junctions can be used to interconnect one-port elements based on traditional scalar waves and two-port elements based on vector waves, thus solving many of the computability problems that would arise using just scalar definitions, while preserving modularity, i.e., modeling circuit elements and topology in a separate fashion.

2.3. WDFs with a Single Vector Nonlinear Two-Port Element

WDFs with a single nonlinearity can be organized into tree-like structures called *connection trees* [10, 8]. Three types of constitutive blocks can be identified in such WD connection trees: the *root*, i.e., the nonlinearity, which has no upward-facing ports and whose downward-facing ports cannot be adapted; *nodes* (typically multi-port topological junctions), which have one upward-facing port and one or more downward-facing ports; *leaves*, which have upward-facing ports and no downward-facing ports [10, 8]. A WD structure of the sort can be solved without employing any iterative solver if the nonlinearity is characterized by an explicit mapping in the WD domain. In fact, delay-free loops can be removed through the *adaptation* of all the leaves and upward-facing ports of junctions. It follows that a WDF based on traditional scalar waves and containing a single nonlinear one-port element with an explicit scattering relation can be implemented in a fully explicit fashion. As outlined in the above description of Fig. 1, a WDF based on scalar waves with a single nonlinear two-port placed at the root would instead be characterized by unavoidable delay-free loops. However, the adoption of a vector definition of waves for modeling the nonlinear two-port element allows us to make also this kind of WD structures *fully explicit*, again, under the assumption that the nonlinearity is characterized by an explicit WD mapping. In fact, as shown in Fig. 2, through proper assignment of the free parameters Z_{11}, Z_{12}, Z_{21} and Z_{22} constituting the reference two-port resistance $\mathbf{Z}_{1,2}$, it is possible to make the pair of junction ports at which the nonlinear two-port element is connected reflection-free. This means that the nonlinear element does not introduce any delay-free loop despite being connected to the junction through a double port connection.

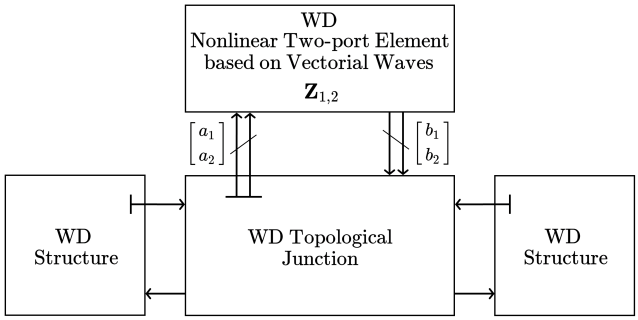


Figure 2: *WD structure featuring a WD nonlinear two-port element based on vector waves connected to a topological junction. The T-shaped stub indicates port adaptation.*

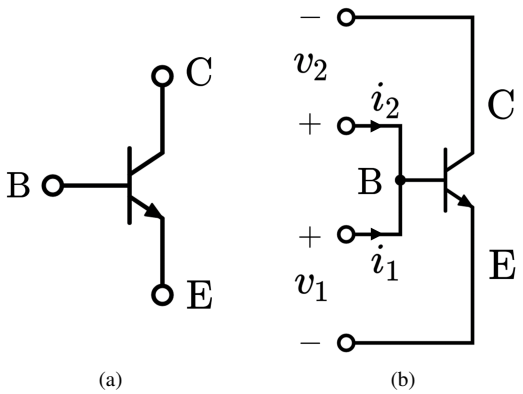


Figure 3: *(a) Symbol of a BJT as a three-terminal device and (b) corresponding two-port element definition.*

3. BJT MODELS BASED ON VECTOR WAVES

A BJT is an electronic device characterized by three terminals called *base*, *emitter* and *collector* - shown in Fig. 3(a) as node B, E and C, respectively. Through the modeling approach discussed in [20], it can be described as a two-port element, as shown in Fig. 3(b). In fact, its behavior is completely described by the voltages v_1 (across base-emitter port) and v_2 (across base-collector port). In the following, we introduce an extension to the Ebers-Moll model (EMM) [23], probably the most widespread large signal BJT model appearing in the literature. As shown in Fig. 4, a series and a parallel resistor are introduced at each of the two ports: $(R_{\mathrm{s}1}, R_{\mathrm{p}1})$ at port 1 and $(R_{\mathrm{s}2}, R_{\mathrm{p}2})$ at port 2. The series resistances encapsulate the contribution of several structural resistances of the device, while the parallel ones are mainly due to current leakage at the two *p-n* junctions forming the BJT. Typically, $R_{\mathrm{p}1}$, $R_{\mathrm{p}2}$ have high values, while $R_{\mathrm{s}1}$, $R_{\mathrm{s}2}$ are very low [28]. The introduced resistors also attenuate numerical problems that might

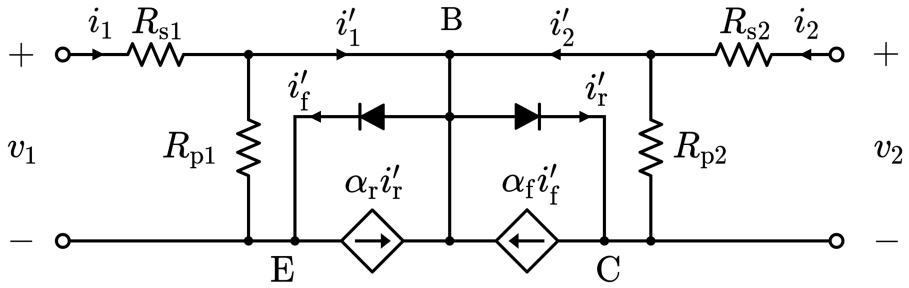


Figure 4: *Proposed extended Ebers-Moll model.*

arise when regions of the v-i characteristic curves of the diodes with extremely high (or extremely low) slopes are visited, especially when dealing with large-amplitude signals [8, 29].

Our extended EMM is mathematically described by the following system of equations

$$\begin{cases} i_1 = i_1' + \frac{v_1 - i_1 R_{s1}}{R_{p1}} \\ i_2 = i_2' + \frac{v_2 - i_2 R_{s2}}{R_{p2}} \end{cases}, \tag{9}$$

where

$$\begin{aligned} i_1' &= I_{s1}\left(e^{\left(\frac{v_1 - i_1 R_{s1}}{\eta_f V_t}\right)} - 1\right) - \alpha_r I_{s2}\left(e^{\left(\frac{v_2 - i_2 R_{s2}}{\eta_r V_t}\right)} - 1\right) \\ i_2' &= I_{s2}\left(e^{\left(\frac{v_2 - i_2 R_{s2}}{\eta_r V_t}\right)} - 1\right) - \alpha_f I_{s1}\left(e^{\left(\frac{v_1 - i_1 R_{s1}}{\eta_f V_t}\right)} - 1\right). \end{aligned} \tag{10}$$

In equations (9) and (10), i_1 is the current through the base-emitter port, i_2 is the current through the base-collector port, α_f is the forward common-base current gain, α_r is the reverse common-base current gain, v_1 is the base-to-emitter voltage, v_2 is the base-to-collector voltage, V_t is the thermal voltage, I_{s1} is the saturation current of the base-emitter *p-n* junction, I_{s2} is the saturation current of the base-collector *p-n* junction, η_f is the ideality factor of the base-emitter *p-n* junction and η_r is the ideality factor of the base-collector *p-n* junction.

3.1. Implicit WD Model based on Modified NR Method

Reformulating (9), the BJT two-port model in Fig. 4 can be characterized by the following system of nonlinear equations in the Kirchhoff domain

$$\mathbf{h}\left(\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}\right) = \begin{bmatrix} i_1' + \frac{v_1 - i_1 R_{s1}}{R_{p1}} - i_1 \\ i_2' + \frac{v_2 - i_2 R_{s2}}{R_{p2}} - i_2 \end{bmatrix} = \mathbf{0}, \tag{11}$$

where $[v_1, v_2]^T$ is the vector of port voltages, $[i_1, i_2]^T$ is the vector of port currents. Since (11) is composed of two coupled transcendental implicit equations, finding an explicit scattering relation in the WD domain is not possible in general, even though some explicit WD mappings based on simplifications have been proposed in some particular cases [30]. To overcome that difficulty, we adjust the NR method to the local resolution of two-port nonlinear WD blocks based on a vector wave definition. To this aim, relationships between port vector variables, (4) and (7), are rewritten as follows [30],

$$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \frac{1}{|\mathbf{Z}_{1,2}|}\begin{bmatrix} Z_{22} & -Z_{12} \\ -Z_{21} & Z_{11} \end{bmatrix}\left(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}\right), \tag{12}$$

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = 2\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} - \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}. \tag{13}$$

Replacing (12) into (11) leads to a vector nonlinear equation $\mathbf{g}(\boldsymbol{\varphi}) = \mathbf{0}$, where $\boldsymbol{\varphi} = [v_1, v_2]^T$. Given the vector wave $[a_1, a_2]^T = \left[a_1^{(k)}, a_2^{(k)}\right]^T$ incident to the nonlinear two-port element and the reference two-port resistance $\mathbf{Z}_{1,2}$, which is set in such a way to make reflection-free the pair of junction ports to which the nonlinear two-port element is connected, the equation $\mathbf{g}(\boldsymbol{\varphi}) = \mathbf{0}$ can be solved using the NR algorithm characterized by the following update rule

$$\boldsymbol{\varphi}^{(k+1)} = \boldsymbol{\varphi}^{(k)} - \left[\mathbf{J}\left(\boldsymbol{\varphi}^{(k)}\right)\right]^{-1}\mathbf{g}\left(\boldsymbol{\varphi}^{(k)}\right), \tag{14}$$

where $\boldsymbol{\varphi}^{(k)} = \left[v_1^{(k)}, v_2^{(k)}\right]^T$ and $\boldsymbol{\varphi}^{(k+1)} = \left[v_1^{(k+1)}, v_2^{(k+1)}\right]^T$ indicate the values of $\boldsymbol{\varphi}$ evaluated at iterations k and $k+1$, respectively, and $\left[\mathbf{J}\left(\boldsymbol{\varphi}^{(k)}\right)\right]^{-1}$ is the inverse of the Jacobian matrix of $\mathbf{g}$ evaluated at $\boldsymbol{\varphi}^{(k)}$. Solving $\mathbf{g}(\boldsymbol{\varphi}) = \mathbf{0}$ requires:

- taking a suitable initial guess $\boldsymbol{\varphi}^{(0)}$;
- repeating (14) up to convergence, i.e., up to the case in which $||\boldsymbol{\varphi}^{(k+1)} - \boldsymbol{\varphi}^{(k)}|| < \epsilon_v$ and $||\mathbf{g}\left(\boldsymbol{\varphi}^{(k+1)}\right)|| < \epsilon_g$, where ϵ_v and ϵ_g are small positive scalar thresholds.

Iterative solvers based on the NR method are arbitrarily accurate and generally more efficient than tabulation methods [31]. The main drawback of such iterative solvers is that their convergence is generally not ensured, since it strongly depends on the chosen initial guess $\boldsymbol{\varphi}^{(0)}$ and it is further compromised by numerical issues related to finite word length representation. To mitigate the effect of such problems, the modified NR (MNR) method introduced in [20] enforces control over the variables $v_1^{(k)}$ and $v_2^{(k)}$ iteration by iteration, through the use of compensation functions, ϕ_1 and ϕ_2, designed to prevent overshooting and to increase the NR method convergence rate (we hereby omit their definition for reasons of space; the reader is kindly referred to [20] for a detailed analysis on the topic). The update equation (14) is thus modified according to

$$\tilde{\boldsymbol{\varphi}}^{(k+1)} = \boldsymbol{\varphi}^{(k)} - \left[\mathbf{J}\left(\boldsymbol{\varphi}^{(k)}\right)\right]^{-1}\mathbf{g}\left(\boldsymbol{\varphi}^{(k)}\right), \tag{15}$$

where $\tilde{\boldsymbol{\varphi}}^{(k+1)} = \left[\tilde{v}_1^{(k+1)}, \tilde{v}_2^{(k+1)}\right]^T$, $\boldsymbol{\varphi}^{(k)} = \left[v_1^{(k)}, v_2^{(k)}\right]^T$, and the two components of the vector $\boldsymbol{\varphi}^{(k+1)}$ are given by

$$v_1^{(k+1)} = \phi_1\left(\tilde{\boldsymbol{\varphi}}^{(k+1)}\right), \quad v_2^{(k+1)} = \phi_2\left(\tilde{\boldsymbol{\varphi}}^{(k+1)}\right). \tag{16}$$

Once the NR convergence condition is met, the two-port voltages are set to $[v_1, v_2]^T = \left[v_1^{(k)}, v_2^{(k)}\right]^T = \boldsymbol{\varphi}^{(k)}$ and the vector of waves $[b_1, b_2]^T$ reflected by the two-port nonlinear element is computed by means of (13).

3.2. Explicit WD Model based on Neural Network

Even though a vector definition of waves allows us to eliminate all the delay-free loops of WD structures containing up to a single nonlinear two-port element, in the previous subsection, we have shown that we still need a local iterative solver to compute the implicit vector scattering equation of the nonlinear WD block characterized by the extended EMM. In this subsection we aim at designing an explicit WD BJT model, thus obtaining a fully explicit WD structure with a single two-port nonlinearity. We propose a data-driven model whose WD mapping approximates the nonlinear scattering relation

$$\mathbf{b} = \mathbf{f}\left(\mathbf{a}\right), \tag{17}$$

which relates vectors of incident waves $\mathbf{a} = [a_1, a_2]^T$ to vectors of reflected waves $\mathbf{b} = [b_1, b_2]^T$ in a fully explicit fashion.

The two-port element behavior can be characterized through suitable simulation or measurement campaigns: sets of port voltages $[v_1, v_2]^T$ and port currents $[i_1, i_2]^T$ measures can be collected forming a Kirchhoff domain dataset for the BJT model. The vector definition of wave variables (4), where, again, $\mathbf{Z}_{1,2}$ is set in such a way to make reflection-free the pair of junction ports to which the nonlinear two-port is connected, allows us to transform the Kirchhoff domain dataset into a vector WD domain dataset of known solutions to the nonlinear vector function (17). The task of modeling the function $\mathbf{f}\left(\cdot\right)$ can be consequently seen as a regression problem, i.e.,

$$\hat{\mathbf{b}} = \tilde{\mathbf{f}}(\mathbf{a}; \theta_{\mathrm{WD}}) \tag{18}$$

where $\tilde{\mathbf{f}}$ is the function approximation given by a suitable neural network architecture [32, 33] whose parameters θ_{WD} are obtained by training the network to predict the current value of the reflected vector waves $\hat{\mathbf{b}}$ given the current value of the incident vector waves $\mathbf{a}$ as input.

Since the adopted reference model of the BJT is characterized by an instantaneous (static) nonlinearity, it is not necessary to rely on neural network structures with memory, such as Recurrent Neural Networks [34] or LSTM networks [35]. For this reason, in the next section, we employ a Multi-Layer Perceptron (MLP) network composed of 2 fully-connected layers with 16 hidden units each and Rectified Linear Unit (ReLU) activation functions (354 trainable parameters).

4. CASE STUDY

As a case study, we develop a Virtual Analog model of the common-emitter amplifier constituting the input stage of the Big Muff Pi distortion pedal - see Fig. 5. The circuit is characterized by a single nonlinear element, namely a 2N5089 BJT, whose extended EMM parameters are reported in Table 1.

The implemented WD structure is shown in Fig. 6 and it is characterized by series/parallel adaptors ($\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$) designed according to [7] and a 6-port WD topological junction $\mathcal{R}_1$ implemented following [15], based on the vectorial definition of waves at the ports of the BJT. Indeed, the matrix of free parameters $\mathbf{Z}_{\mathcal{R}_1}$ of the junction $\mathcal{R}_1$ has a 2×2 submatrix $\mathbf{Z}_{1,2}$ positioned at the ports where the BJT is connected. All the linear elements at the leaves and all the upward facing ports of the junctions are adapted according to traditional WDF principles [7]. The nonlinear two-port element Q_1 is the root of the designed connection tree structure. The free parameters of the port resistance matrix $\mathbf{Z}_{1,2}$ are set to zero out the four entries of the first 2×2 block on the diagonal of the junction scattering matrix $\mathbf{S}_{\mathcal{R}_1}$. As previously mentioned in Section 2.3, this makes the pair of junction ports at which Q_1 is connected reflection-free.

Table 1: *2N5089 BJT: values of the extended EMM parameters.*

Parameter	Value	Description
V_{t}	25.85 mV	thermal voltage
α_{f}	0.9993	BJT forward current gain
α_{r}	0.5579	BJT reverse current gain
$R_{\mathrm{s1}}, R_{\mathrm{s2}}$	$10^{-5}\ \Omega$	BJT series port resistance
R_{p1}, R_{p2}	10^{11}, $10^{8}\ \Omega$	BJT parallel port resistance
$\eta_{\mathrm{f}}, \eta_{\mathrm{r}}$	1	BJT ideality factor
I_{s1}	5.9151 fA	B-E junction saturation current
I_{s2}	10.595 fA	B-C junction saturation current

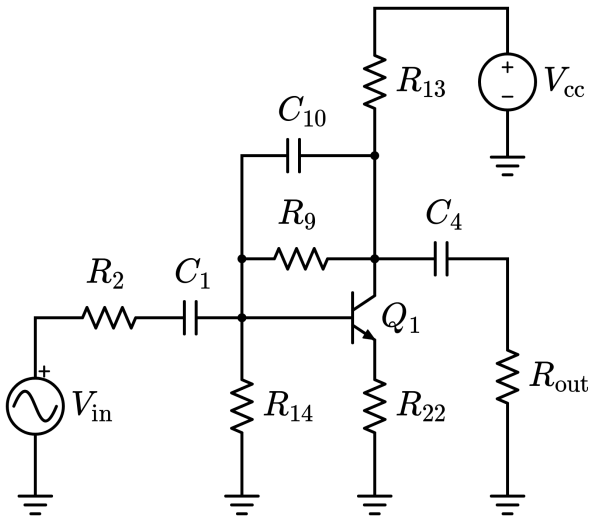


Figure 5: *Big Muff Pi input stage.*

4.1. Dataset and Model Training

In order to generate a suitable Kirchhoff domain dataset for the 2N5089 BJT, we implement the extended EMM using Mathworks Simscape and the model parameters reported in Table 1. The nonlinear characteristic of the model is sampled devising specific port voltage signals to reproduce common device operating ranges. In particular, the base-to-collector voltage v_2 is assumed to vary in the range $[-10, 0]$ V, while the base-to-emitter voltage v_1 varies in the range $[-0.8, 0.8]$ V. To sample the device, v_2 is set to 10^3 different equally spaced DC values extracted from its operating range and, for each of these values, v_1 is linearly increased for a duration of 0.1 seconds to cover its entire operating range; the corresponding base-to-collector current i_2 and base-to-emitter current i_1 are acquired with a sample rate $f_s = 96$ kHz. The

Table 2: *Values of the parameters of the Big Muff Pi input stage circuit shown in Fig. 5.*

R_2	R_9	R_{13}	R_{14}	R_{22}	R_{out}	C_1	C_4	C_{10}	V_{cc}
39 kΩ	470 kΩ	10 kΩ	47 kΩ	100 Ω	101 kΩ	1 µF	1 µF	470 pF	9 V

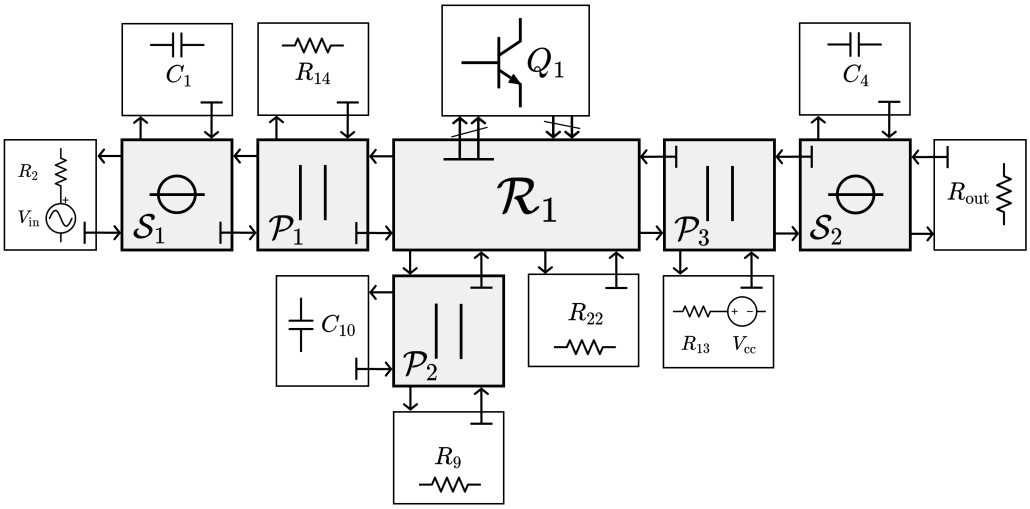


Figure 6: *WDF realization of the circuit shown in Fig. 5.*

complete simulation of the Kirchhoff domain signals v_1, i_1, v_2, i_2 amounts to 100 seconds.

The formed dataset can be expressed in the WD domain by using the vector wave transformation (4), where the free parameters Z_{11}, Z_{12}, Z_{21} and Z_{22} are set according to the already discussed port adaptation condition. From the total amount of data, the 80% are used for training, while the 20% are held out for evaluation purposes. The pairs of corresponding input and output wave variables $\left([a_1, a_2]^T, [b_1, b_2]^T\right)$ are assembled in batches containing 256 elements.

The MLP network described in Section 3.2 is implemented in Python using Pytorch [36] and it is trained for 500 epochs using Adam [37] to minimize the following loss function:

$$\mathcal{L} = \mathcal{E}\left(b_1, \hat{b}_1\right) + \mathcal{E}\left(b_2, \hat{b}_2\right) \tag{19}$$

where

$$\mathcal{E}\left(y, \hat{y}\right) = \frac{\sum_k \left(y_k - \hat{y}_k\right)^2}{\sum_k y_k^2} \tag{20}$$

is the Normalized Mean Squared Error (NMSE). Notably, the two terms summed in the loss function (19) are related to the two components of the output vector. To evaluate the model, we compute the model predictions over the evaluation set. The NMSE computed over those predictions is equal to 1.02×10^{-8}.

4.2. Results

In this subsection, we discuss the numerical results obtained from the simulation of the nonlinear WD structure shown in Fig. 6 using the two different WD BJT models presented in Sec. 3.1 and Sec. 3.2, respectively. Remarkably, thanks to the modularity of WDFs, it is possible to test both the WD BJT implementations employing the same WD structure and just substituting the nonlinear WD block at the root. In case the MNR method-based model is used, an iterative solver is needed to solve the two-port nonlinearity, while, in case the neural network-based model is used, the WD structure can be implemented in fully explicit fashion, since the nonlinearity is also expressed as an explicit wave mapping.

The circuit is tested with an input signal $V_{\text{in}} = A\sin(2\pi k\, f_0/f_s)$, where k is the sampling index, while $f_s = 96$ kHz and $f_0 = 1$ kHz are the sampling frequency and the fundamental frequency, respectively. According to the circuit analysis in [38], we set the input signal amplitude to $A = 0.7$ V to force the input stage into saturation, therefore causing asymmetric clipping. The input signal has a duration of one second. All the simulation algorithms of the WD structures are implemented as MATLAB scripts and are run on a laptop-mounted Intel Core i5-1240P 1.70 GHz CPU.

The simulation results related to the last 5 periods of the input signal are reported in Fig. 7(a) and Fig. 7(b). Both the WD implementations of the BJT closely match the same reference signal resulting from a Mathworks Simscape simulation of the circuit in Fig. 5, where the BJT has been modeled employing the extended EMM. The deviation between the reference Simscape signal and the neural network-based WD simulation is quantifiable by a NMSE of 2.32×10^{-5}, while the MNR method-based WD simulation is able to achieve an NMSE of 1.88×10^{-6}.

For measuring the computational cost associated to the two different WD implementations, $\Gamma = 1000$ identical simulations of the two WD structures are executed and the Real Time Ratio (RTR) is computed. The RTR is a dimensionless quantity indicating how fast the simulation is with respect to real time. If we consider Γ simulations having X samples as input and an execution time t_{c_i} for a single iteration over the input, the RTR can be computed as $\text{RTR} = (1/\Gamma)\sum_{i=1}^{\Gamma}\left(t_{c_i}/\left(X/f_s\right)\right)$, where f_s is the sampling frequency. The algorithm runs in real time if $\text{RTR} < 1$.

We obtain $\text{RTR} \simeq 1.65$ for the MNR method-based WD implementation and $\text{RTR} \simeq 0.21$ for the neural network-based WD implementation. While the MNR method guarantees arbitrarily

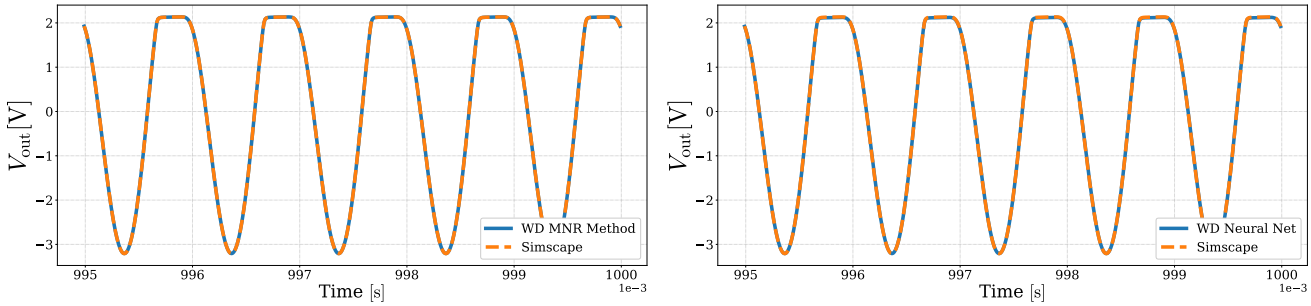


Figure 7: *Voltage V_{out} measured across the resistor R_{out}. (a) Comparison between the MNR method-based BJT WD implementation and Mathworks Simscape. (b) Comparison between the neural network-based BJT WD implementation and Mathworks Simscape.*

good results in terms of accuracy, the iterative solver considerably increases the algorithm execution time. However, with almost comparable accuracy results, the WD structure containing the explicit neural network-based WD BJT model proves to be far more efficient as far as execution time is concerned, being able to run in real time in the MATLAB environment.

5. CONCLUSIONS

In this paper, we have shown that, by adopting a vector definition of waves, it is possible to design WDFs which contain a single two-port nonlinearity connected to a topological junction through a pair of ports with no delay-free loops. It follows that, when the two-port nonlinearity is characterized by an explicit WD mapping, the resulting WDF is fully explicit. The proposed approach allows us to preserve the desirable modularity property of traditional scalar WDFs, according to which circuit elements and connection networks are modeled with separate input-output blocks.

We proposed two vector wave-based models of BJT transistors described with an extended EMM. The first one is implicit, and it relies on a MNR method to solve its nonlinear scattering relation. The second is fully explicit, as it relies on a neural network model which approximates the explicit scattering equation relating the input vector of incident waves to the corresponding vector of reflected waves. We finally provided an accuracy and performance comparison of the two WD BJT models with reference to a specific VA application scenario. The fully explicit WD structure including as root a neural network-based model of a two-port BJT proved to be an excellent compromise between accuracy and computational cost. This shows that the integration of efficient data-driven blocks of nonlinear devices into the WDFs framework might lead the way towards the real time implementation of increasingly complex nonlinear circuits.

Future work might concern the application of vector WDFs for the modeling of generic N-port nonlinear elements. A noteworthy extension would also be considering data-driven methods for the characterization of multi-port nonlinearities with memory, leveraging models or experimental measurements of transistors or vacuum tubes. Finally, the proposed strategy should be validated from a broader perspective, comparing it to other approaches developed in the literature.

6. REFERENCES

[1] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, *Virtual analog effects*, pp. 473–522, John Wiley & Sons, United Kingdom, 2nd edition, 2011.

[2] D. Bouvier, T. Hélie, and D. Roze, "Phase-based order separation for Volterra series identification," *International Journal of Control*, vol. 94, no. 8, pp. 2104–2114, 2021.

[3] A. Wright, E.P. Damskägg, V. Välimäki, et al., "Real-time black-box modelling with recurrent neural networks," in *Proc. 22nd Int. Conf. Digital Audio Effects (DAFx-19)*, Birmingham, UK, 2019, University of Birmingham.

[4] G. Borin, G. De Poli, and D. Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 5, pp. 597–605, 2000.

[5] A. Falaize-Skrzek and T. Hélie, "Simulation of an analog circuit of a wah pedal: a port-Hamiltonian approach," in *Proc. 135th Audio Engineering Society Convention*, New York, USA, 2013, Audio Engineering Society.

[6] G. De Sanctis and A. Sarti, "Virtual analog modeling in the wave-digital domain," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 715–727, 2010.

[7] A. Fettweis, "Wave digital filters: Theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986.

[8] A. Bernardini, P. Maffezzoni, and A. Sarti, "Linear multistep discretization methods with variable step-size in nonlinear wave digital structures for virtual analog modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1763–1776, 2019.

[9] K. Meerkotter and R. Scholz, "Digital simulation of nonlinear circuits by wave digital filter principles," in *1989 IEEE International Symposium on Circuits and Systems (ISCAS)*, Portland, USA, 1989, vol. 1, pp. 720–723, IEEE.

[10] A. Sarti and G. De Sanctis, "Systematic methods for the implementation of nonlinear wave-digital structures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 2, pp. 460–472, 2009.

[11] T. Felderhoff, "A new wave description for nonlinear elements," in *Proc. IEEE Int. Symp. Circuits Syst.* Sep, 1996, vol. 3, pp. 221–224, IEEE.

[12] A. Sarti and G. De Poli, "Toward nonlinear wave digital filters," *IEEE Transactions on Signal Processing*, vol. 47, no. 6, pp. 1654–1668, 1999.

[13] H. Carlin, "Singular network elements," *IEEE Transactions on Circuit Theory*, vol. 11, no. 1, pp. 67–72, 1964.

[14] A. Bernardini and A. Sarti, "Biparametric wave digital filters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. PP, pp. 1–13, 03 2017.

[15] A. Bernardini, K. J. Werner, J. O. Smith, III, and A. Sarti, "Generalized wave digital filter realizations of arbitrary reciprocal connection networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 694–707, 2019.

[16] K. J. Werner, V. Nangia, J. O. Smith III, and J. S. Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," in *Proc. 18th Int. Conf. Digital Audio Effects (DAFx-15)*, Trondheim, Norway, 2015, pp. 387–394, Norwegian University of Science and Technology.

[17] D. T. Yeh, J. S. Abel, and J. O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part I: Theoretical development," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 728–737, 2010.

[18] D. T. Yeh, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part II: Bjt and vacuum tube examples," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1207–1216, 2012.

[19] M. J. Olsen, K. J. Werner, and J. O. Smith III, "Resolving grouped nonlinearities in wave digital filters using iterative techniques," in *Proc. 19th Int. Conf. Digital Audio Effects (DAFX-16)*, Brno, Czech Republic, 2016, pp. 279–286, Brno University of Technology.

[20] A. Bernardini, A. E. Vergani, and A. Sarti, "Wave digital modeling of nonlinear 3-terminal devices for virtual analog applications," *Circuits, Systems, and Signal Processing*, vol. 39, no. 7, pp. 3289–3319, 2020.

[21] L. Kolonko, B. Musiol, J. Velten, and A. Kummert, "A split-modular approach to wave digital filters containing bipolar junction transistors," in *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Lansing, 2021, pp. 840–843, IEEE.

[22] A. Bernardini, P. Maffezzoni, and A. Sarti, "Vector wave digital filters and their application to circuits with two-port elements," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 3, pp. 1269–1282, 2021.

[23] J. J. Ebers and J. L. Moll, "Large-signal behavior of junction transistors," *Proceedings of the IRE*, vol. 42, no. 12, pp. 1761–1772, 1954.

[24] J. Chowdhury and C. J. Clarke, "Emulating diode circuits with differentiable wave digital filters," in *19th Sound and Music Computing Conference*, Saint-Étienne, France, 2022, pp. 2–9, SMC Network.

[25] C. C. Darabundit, D. Roosenburg, and J. O. Smith, "Neural net tube models for wave digital filters," in *Proc. 25th Int. Conf. Digital Audio Effects (DAFx20in22)*, Vienna, Austria, 2022, pp. 153–160, Vienna University of Music and Performing Arts.

[26] O. Massi, A. I. Mezza, R. Giampiccolo, and A. Bernardini, "Deep learning-based wave digital modeling of rate-dependent hysteretic nonlinearities for virtual analog applications," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2023, no. 1, pp. 12, 2023.

[27] K. J. Werner, A. Bernardini, J. O. Smith, and A. Sarti, "Modeling circuits with arbitrary topologies and active linear multiports using wave digital filters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4233–4246, 2018.

[28] M. G. Villalva, J. R. Gazoli, and E. R. Filho, "Comprehensive approach to modeling and simulation of photovoltaic arrays," *IEEE Transactions on Power Electronics*, vol. 24, no. 5, pp. 1198–1208, 2009.

[29] A. Bernardini, K.J Werner, P. Maffezzoni, and A. Sarti, "Wave digital modeling of the diode-based ring modulator," in *Proc. 144th Audio Engineering Society Convention*, Milan, Italy, 2018, Audio Engineering Society.

[30] A. Bernardini, K. J. Werner, A. Sarti, and J. O. Smith, III, "Modeling nonlinear wave digital elements using the Lambert function," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 8, pp. 1231–1242, 2016.

[31] A. Bernardini, E. Bozzo, F. Fontana, and A. Sarti, "A wave digital Newton-Raphson method for virtual analog modeling of audio circuits with multiple one-port nonlinearities," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2162–2173, 2021.

[32] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[33] C. Huang, "Relu networks are universal approximators via piecewise linear or constant functions," *Neural Computation*, vol. 32, no. 11, pp. 2249–2278, 2020.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] Electrosmash, "Big Muff Pi analysis," Available at https://www.electrosmash.com/big-muff-pi-analysis, accessed March 31, 2023.

ANTIALIASING PIECEWISE POLYNOMIAL WAVESHAPERS

Kurt James Werner *

Soundtoys, Inc.
Burlington, VT
kurt.james.werner@gmail.com

Emma Azelborn

Native Instruments, GmbH.
Boston, MA
emma.azelborn@native-instruments.com

ABSTRACT

Memoryless waveshapers are commonly used in audio signal processing. In discrete time, they suffer from well-known aliasing artifacts. We present a method for applying antiderivative antialising (ADAA), which mitigates aliasing, to any waveshaping function that can be represented as a piecewise polynomial. Specifically, we treat the special case of a piecewise linear waveshaper. Furthermore, we introduce a method for for replacing the sharp corners and jump discontinuities in any piecewise linear waveshaper with smoothed polynomial approximations, whose derivatives match the adjacent line segments up to a specified order. This piecewise polynomial can again be antialiased as a special case of the general piecewise polynomial. Especially when combined with light oversampling, these techniques are effective at reducing aliasing and the proposed method for rounding corners in piecewise linear waveshapers can also create more "realistic" analog-style waveshapers than standard piecewise linear functions.

1. INTRODUCTION

Memoryless nonlinearities are commonly used in audio signal processing algorithms, as waveshapers and wavefolders used in synthesizers [1–3], to model guitar amplifier distortion [4, 5], ring modulators [6], as models of electrical elements in virtual analog modeling (for instance, real electrical components like Shockley's diode model [7] or imaginary electrical components [8]), as gain computers in dynamic range controllers, as utility saturators to restrict a signal output to a certain range, etc. Given a continuous-time input signal $x(t)$, a memoryless nonlinearity $f(\cdot)$ produces a continuous-time output signal $y(t)$ by

$$y(t) = f\left(x(t)\right) \ . \tag{1}$$

In a digital signal processing context, with discrete time index n, a naive implementation is

$$y[n] = f\left(x[n]\right) \ . \tag{2}$$

This naive implementation suffers from one well-known artifact: aliasing distortion. The nonlinear function $f(\cdot)$ expands the bandwidth of the input signal $x[n]$, and if any frequency component arises that is above the Nyquist rate (half of the sampling rate: $f_s/2$), it will end up aliased down into the baseband. Although aliasing is sometimes introduced deliberately as a creative effect or to mimic a hardware device with audible aliasing distortion [9], it is usually considered detrimental and bad-sounding. As such, audio DSP designers typically use several types of approaches to mitigate aliasing distortion: 1. Oversampling, 2. Non-oversampled antialiasing (such as ADAA), and/or 3. Altering the waveshape to expand the signal bandwidth less. Oversampling is a classic and effective approach [4, 10], but can be quite costly if a large oversampling ratio is needed. Non-oversampled antialiasing is a class of techniques that include those specialized to waveform synthesis and for memoryless nonlinearities, including the recent and well-known Antiderivative Antialiasing (ADAA) technique [11–17]. In fact, we do not have to choose just one technique, but can often combine them. For instance, ADAA works best when combined with modest (at least $2\times$) oversampling.

If a designer is willing to allow small changes to the shape of their memoryless nonlinearity, the last option should be considered quite attractive. First of all, sacrificing a small bit of accuracy for increased sound quality is often a valid tradeoff. Second of all, in certain circumstances, smoothing out and/or rounding corners on a memoryless nonlinearity is often more "realistic," in the sense that waveshapers created with analog electronic circuits (including diodes, tubes, transistors) typically have very soft corners, whereas some of the classic techniques for creating digital memoryless nonlinearities, such as piecewise-linear representations (PWL) have sharp corners. For this reason, it can be useful to specify a PWL model augmented with smoothness controls for each corner. In fact, this approach is commonly used in digital dynamic range control systems, which typically offer "knee" parameters on their gain computers.

In this paper, we will present a method for applying ADAA to any piecewise-polynomial (PWP) waveshaping function. The special case of a piecewise linear (PWL) waveshaper is discussed, including how to convert from the more convenient breakpoint representation to the standard line-segment representation, for which ADAA can be applied in the identical fashion. Finally, we introduce a technique for replacing any rounded corners and jump discontinuities in a PWL waveshaper with rounded corners, making it a technique that combines *all three* of the aforementioned approaches. This method has a number of attractive features: 1. It's parameterized by piecewise line segments and finite jump discontinuities. Jump discontinuities should be attended to, since they appear commonly in memoryless nonlinearities, such as "dead-zone" and bitcrushers; 2. It has smooth corners made of finite-order polynomials; and 3. It has controllable finite width of corners and degree of smoothness at splice points.

This method combines the well-known ADAA approach with aspects of the canonical piecewise linear representations (CPLR) [18, 19] and the smoothed corners are based on a piecewise polynomial model of certain soft clipper curve, whose smoothness can be set to an arbitrary level, as introduced and discussed at length

* Early stages of this research were conducted during the 1st author's employment at iZotope, Inc., Boston, MA.

Table 1: *Polynomial coefficients for the piecewise polynomial from* (5) *and Fig. 1 and its antiderivative.*

j	$p_{j,0}$	$p_{j,1}$	$p_{j,2}$	$\overline{C}_j$	$\hat{C}_j$	C_j	$P_{j,1}$	$P_{j,2}$	$P_{j,3}$
0	3	2	0	0	0	$\frac{5}{3}$	3	2	0
1	0	0	1	0	$-\frac{5}{3}$	0	0	0	$\frac{1}{3}$
2	$\frac{1}{2}$	0	0	0	$-\frac{11}{6}$	$-\frac{1}{6}$	$\frac{1}{2}$	0	0

by Robert Bristow-Johnson, Olli Niemitalo, et al. [20–22], which stands in as a smoothed approximation to the sign function, and a related curve which is a smooth approximation of the absolute value function. The first curve is related to a family of curves known in the image processing field, including the simplest case, SmoothStep [23] and higher-order generalizations, and its lowest-order case is identical to a cubic soft clipper that is well-known in audio signal processing [24, 25].

The paper is structured as follows. §2 reviews piecewise polynomial (PWP) waveshaping functions and shows how to apply Antiderivative Antialiasing (ADAA) to them. §3 details a special case of the PWP waveshaper, the piecewise linear (PWL) waveshaper, again showing how ADAA can be performed. §4 shows how to round the corners of a PWL waveshaper using polynomial segments, which can be parameterized by a PWL representation, corner widths, and the order of smoothness enforced at each corner, again showing how ADAA can be applied. §5 presents several different ways to arrive at the same method for rounding the corners, based on ensuring smoothness up to a certain degree. §6 gives two brief case studies. §7 concludes.

2. PIECEWISE POLYNOMIALS WAVESHAPERS

A piecewise polynomial (PWP) with $J+1$ segments is given by

$$f(x) = \begin{cases} p_0(x), & x < x_1 \\ p_1(x), & x_1 \le x < x_2 \\ \cdots & \cdots \\ p_{J-1}(x), & x_{J-1} \le x < x_J \\ p_J(x), & x_J \le x \end{cases} \tag{3}$$

each of the $J+1$ polynomials has an order Φ_j and is defined as

$$p_j(x) = \sum_{\rho=0}^{\Phi_j} p_{j,\rho} x^\rho = p_{j,0} + p_{j,1}x + \cdots + p_{j,\Phi_j} x^{\Phi_j} \,. \tag{4}$$

An illustrative piecewise polynomial

$$f(x) = \begin{cases} p_0(x) = 2x + 3, & x < (x_1 = -1) \\ p_1(x) = x^2, & (x_1 = -1) \le x < (x_2 = 1) \\ p_2(x) = \frac{1}{2}, & (x_2 = 1) \le x \,. \end{cases} \tag{5}$$

is shown in the top of Fig. 1.[1]

[1] It is worth mentioning two efficient methods for evaluating polynomials: Horner's method and Estrin's scheme. Horner's method uses fewer operations but Estrin's scheme is more parallelizable, so the more efficient one will depend on implementation context. For higher-order polynomials, it may be more efficient to bake the waveshapes into lookup tables rather than evaluating the polynomials directly.

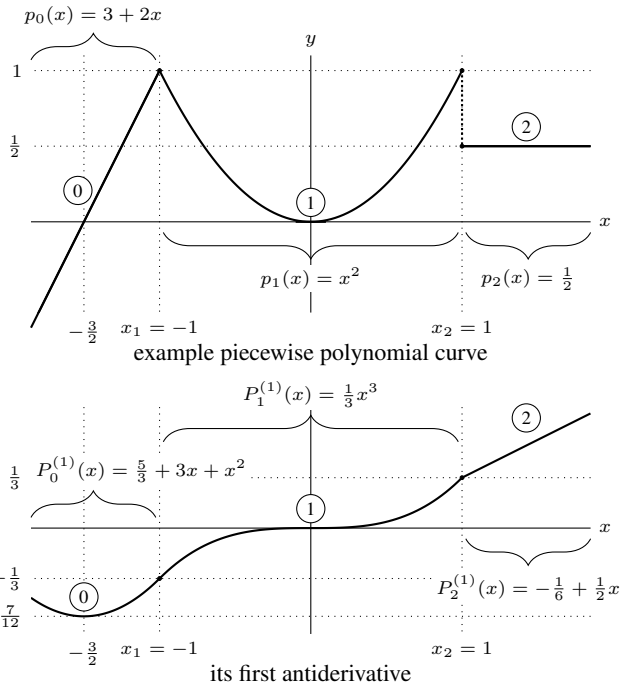


Figure 1: *An example of a piecewise polynomial curve (top) and its first antiderivative (bottom), with constants of integration chosen to enforce C^0 smoothness.*

2.1. Antialiasing Piecewise Polynomial Waveshapers

We can antialias any piecewise polynomial using the well-known Antiderivative Antialiasing (ADAA) method [11], which works by approximating the process of upsampling, applying a nonlinearity, and downsampling without actually doing anything at an upsampled rate. For a nonlinearity $f()$, ADAA produces expressions that use additions, multiplications, and divisions of various antiderivatives $F^{(N)}()$, where N indicates the antiderivative order. This results in potential divisions by zero (or near zero, which can cause numerical issues), which must be dealt with using "escape conditions": special cases where a small signal linearization that avoids the problematic division substitutes for the original expression.

The simplest form is first-order ADAA

$$y[n] = \begin{cases} \frac{F^{(1)}(u[n]) - F^{(1)}(u[n-1])}{u[n] - u[n-1]} \,, & \epsilon < |u[n] - u[n-1]| \\ f\left(\frac{u[n]+u[n+1]}{2}\right) \,, & \text{otherwise} \,, \end{cases} \tag{6}$$

where ϵ is a very small number (we use $\epsilon = 10^{-8}$) and the second case is used to escape numerical ill-conditioning issues. Again, $F^{(1)}()$ is the first antiderivative of $f()$. u is the input signal.[2]

The antiderivative of a polynomial of order Φ is another polynomial of order $\Phi + 1$

$$P_j^{(1)}(x) = C_j + \sum_{\rho=1}^{\Phi_j+1} \frac{p_{j,\rho-1}}{\rho} x^\rho \,. \tag{7}$$

[2] Note that we draw a distinction between the input to the waveshaping function or its antiderivate (which we always call x) and the input signal itself, which we call u. This is because the input to the waveshaping functions or antiderivatives (x) are sometimes filtered versions of the input signal itself (u), i.e., during the escape condition.

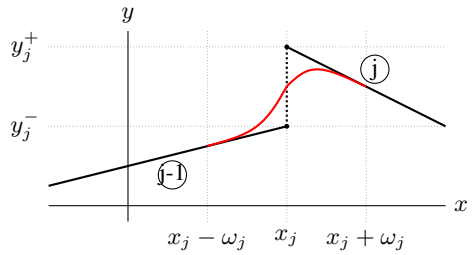


Figure 2: *Naming conventions around the single jth breakpoint, where ω_j is the "double-width" of the curved segment for the cases of a PWL waveshaper with rounded corners, as discussed in §4.*

However, we must be careful in setting the constants of integration C_j. In general, leaving them at zero will lead to large jump discontinuities in the antiderivative. Instead, we must set the constants of integration C_j so that the $P_{j-1}^{(1)}(x_j) = P_j^{(1)}(x_j)$, $j \in \{1, 2, \cdots, J\}$. We can also consider one last degree of freedom, the "global" vertical offset of the function. This can be set entirely arbitrarily, since it will cancel out immediately in the numerator of the antiderivative antialiasing equation (6). We choose it so that $F^{(1)}(0) = 0$, mainly for visual appeal on our plots.

To accomplish this, we start with a piecewise polynomial $\overline{F}^{(1)}(x)$ with all constants of integration zero: $\overline{C}_j = 0, j \in \{0, 1, \cdots, J\}$. Next, we form a piecewise polynomial $\hat{F}^{(1)}(x)$ by first setting $\hat{C}_0 = 0$. Then, for all other values of $j \in \{1, 2, \cdots, J\}$, we choose $\hat{C}_j$ so that the segments line up with C^0 smoothness:

$$\hat{C}_j = \hat{C}_{j-1} + \sum_{\rho=1}^{\Phi_{j-1}+1} \left(\frac{p_{j-1,\rho-1}}{\rho} x^\rho \right) - \sum_{\rho=1}^{\Phi_j+1} \left(\frac{p_{j,\rho-1}}{\rho} x^\rho \right). \quad (8)$$

This can be accomplished by finally setting

$$C_j = \hat{C}_j - \hat{F}^{(1)}(0), \; j \in 0, 1, 2, \cdots, J \, . \quad (9)$$

Which finally gives us our polynomial $F^{(1)}(x)$. All of the polynomial coefficients and intermediate steps of finding the constants of integration for this example PWP curve are shown in Tab. 1 and the resulting curve is shown in the bottom of Fig. 1.

2.2. An example

Starting with our example piecewise polynomial (5), we can antidifferentate them to form the intermediate polynomial

$$\overline{F}^{(1)} = \begin{cases} \overline{P}_0^{(1)}(x) = 3x + 2x^2, & x < -1 \\ \overline{P}_1^{(1)}(x) = \frac{1}{3}x^3, & -1 \le x < 1 \\ \overline{P}_2^{(1)}(x) = \frac{1}{2}x, & x \le 1 \end{cases}, \quad (10)$$

i.e., one where the unadjusted constants of integration are $\overline{C_0} = 0$, $\overline{C_1} = 0$, and $\overline{C_2} = 0$.

Using (8), we can form a piecewise polynomial of the antiderivative with no jump discontinuities, where $\hat{C}_0 = 0$, $\hat{C}_1 = -\frac{5}{3}$, and $\hat{C}_2 = -\frac{11}{6}$,

$$\hat{F}^{(1)} = \begin{cases} \hat{P}_0^{(1)}(x) = 3x + 2x^2, & x < -1 \\ \hat{P}_1^{(1)}(x) = -\frac{5}{3} + \frac{1}{3}x^3, & -1 \le x < 1 \\ \hat{P}_2^{(1)}(x) = -\frac{11}{6} + \frac{1}{2}x, & x \le 1 \end{cases} . \quad (11)$$

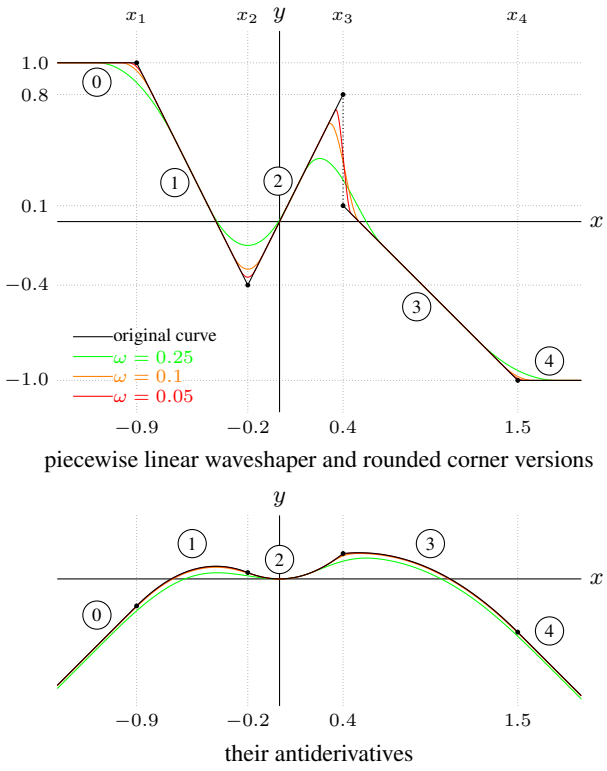


Figure 3: *An example of a piecewise linear curve (top) and its antiderivative (bottom), including three versions with spliced-in rounded corners of "double-width" ω, as discussed in §4. Here we have $\omega_j = \omega, \forall j$.*

Finally, by evaluating $\hat{F}^{(1)}(0) = -\frac{5}{3}$, we form the actual constants of integration $C_0 = \frac{5}{3}$, $C_1 = 0$, and $\hat{C}_2 = -\frac{1}{6}$ and arrive at the actual constants of integration

$$F^{(1)} = \begin{cases} P_0^{(1)}(x) = \frac{5}{3} + 3x + 2x^2, & x < -1 \\ P_1^{(1)}(x) = \frac{1}{3}x^3, & -1 \le x < 1 \\ P_2^{(1)}(x) = -\frac{1}{6} + \frac{1}{2}x, & x \le 1 \end{cases} . \quad (12)$$

This is now suitable for use in the first-order ADAA equation (6) or higher-order generalizations.

2.3. Higher-order antiderivatives

Higher-order [13, 26, 27] and other variations [12, 14, 16] on antiderivatives antialiasing algorithms exist, for instance 2nd or 3rd order ADAA, which involve expressions involving higher order antiderivatives such as $F^{(2)}(x)$, $F^{(3)}(x)$, etc., as well as more complex expressions and escape conditions. The process of defining $F^{(N)}(x)$ from $F^{(N-1)}(x)$ for any order N is identical to the process of forming $F^{(1)}(x)$ from $f(x)$, so needs not be discussed in detail here. Because the antiderivative of any polynomial is another polynomial of one higher order, we are guaranteed that we can analytically produce as many antiderivatives as are required. For other types of functions, analytic antiderivatives do not always exist (although they can always be approximated numerically).

Table 2: *Breakpoint locations and extremal slopes and derived slope and offsets for the example curve shown in Fig. 3. The y-values and σ value associated with jump discontinuities are shaded.*

j	x_j	y_j^-	y_j^+	b_j	m_j	β_j	μ_j	α_j	σ_j	$\hat{C}_j$	$\overline{C}_j$	C_j	$P_{j,1}$	$P_{j,2}$
0	——	——	——	1.0	0.0	——	——	——	——	0	0	0.73	1.0	0.0
1	−0.9	1.0	1.0	−0.8	−2.0	−0.8	−1.0	−1.0	0.0	0	−0.81	−0.08	−0.8	−1.0
2	−0.2	−0.4	−0.4	0.0	2.0	0.0	0.0	2.0	0.0	0	−0.73	0.0	0.0	1.0
3	0.4	0.8	0.1	0.5	−1.0	0.0	0.5	−1.5	−0.35	0	−0.69	0.04	0.5	−0.5
4	1.5	−1.0	−1.0	−1.0	0.0	0.5	−0.5	0.5	0.0	0	0.435	1.165	−1.0	0.0

3. PIECEWISE LINEAR WAVESHAPERS

A special case that often arises is the case of piecewise linear (PWL) waveshapers. A graphical representation of an example piecewise linear curve is shown in Fig. 3. These are in fact a special case of the previously discussed peicewise-polynomial waveshapers, where, again for J breakpoints x_j, $j \in 1, 2, \cdots, J$ and $J+1$ segments, each polynomial segment is just a line segment

$$p_j(x) = p_{j,0} + p_{j,1}x \tag{13}$$

To be more specific that it is a line we will call these $\ell(x)$ and to match the conventional notation for a line, we will use m for slope and b for offset. Now, the entire PWL function is given by

$$f(x) = \begin{cases} \ell_0(x), & x < x_1 \\ \ell_1(x), & x_1 \le x < x_2 \\ \cdots & \cdots \\ \ell_{J-1}(x), & x_{J-1} \le x < x_J \\ \ell_J(x), & x_J \le x \end{cases} \tag{14}$$

where each line segment is defined by

$$\ell_j(x) = b_j + m_j x_j, \quad j \in \{0, 1, 2, \cdots, J\} . \tag{15}$$

We require that the x_j, coordinates be ordered as

$$x_1 < x_2 < x_3 < \cdots < x_J . \tag{16}$$

It's often convenient, in defining a PWL function, to not deal with the slopes and offsets directly, but rather to specify the endpoints of each line segments. A piecewise linear function with J breakpoints and $J+1$ line segments (which may have jump discontinuities between them) can be fully specified by

1. the slope of the furthest-left line segments: m_0.
2. J breakpoints, $j \in \{1, 2, \cdots, J\}$. They are written in the form (x_j, y_j), with a single y coordinate, when there is no jump discontinuity. They are written in the form (x_j, y_j^-, y_j^+) when there is a jump discontinuity, where the superscript indicates which y coordinate is associated with the $(-)$ or right $(+)$ line segment. When there is no jump discontinuity, we will still need to refer to the y coordinate associated with each line segment, but in that case we simply have $y_j^- = y_j^+ = y_j$.
3. the slope of the furthest-right line segments: m_J.

Again, the x coordinates must be ordered by $x_1 < x_2 < x_3 < \cdots < x_J$, essentially meaning that none of the line segments may be vertical—instead, an instantaneous vertical jump should be represented as a jump discontinuity between two line segments—and also that the line segments are indexed in increasing order from left to right. Finally, all quantities should have finite values. These naming conventions are shown graphically in Fig. 2.

Given this, the slopes m and offsets b are determined from the breakpoints and extremal slopes by

$$m_j = \frac{y_{j+1}^- - y_j^+}{x_{j+1} - x_j} , \qquad j \in \{1, \cdots, J-1\} \tag{17}$$

$$b_0 = y_1^- - m_0 x_1 \tag{18}$$

$$b_j = y_j^- - m_{j-1} x_j = y_j^+ - m_j x_j, \ j \in \{1, \cdots, J-1\} \tag{19}$$

$$b_J = y_J^+ - m_J x_J . \tag{20}$$

and m_0 and m_J are already given. The resulting slopes and offsets for the example piecewise linear curve (Fig. 3) are tabulated alongside the original breakpoint locations in Tab. 2.

3.1. Antialiasing Piecewise Linear Waveshapers

Applying antialiasing to the PWL waveshaper is done identically to the more general piecewise polynomial. For this special case, the antiderivative of the PWL function is a piecewise-quadratic function, i.e., the order of each line segment in the PWL is $\Phi_j = 1$, so the order of the each segment of its antiderivative is $\Phi_j = 2$ and

$$P_j^{(1)}(x) = C_j + \frac{p_{j,0}}{1}x + \frac{p_{j,1}}{2}x^2 = C_j + \frac{b_j}{1}x + \frac{m_j}{2}x^2 . \tag{21}$$

The constants of integration are set identically to the PWP case.

4. ROUNDED PIECEWISE LINEAR WAVESHAPERS

Now we will introduce a formulation that splices a rounded corner function $g_j(x)$, $j \in \{1, 2, \cdots, J\}$ between each line segment $\ell_{j-1}(x)$ and $\ell_j(x)$. Without yet saying much the exact shape of these rounded corners, the main quantity that they must be parameterized by is the double-width of each corner, denoted by $\omega_j > 0$, which may be different for each corner. These must obey

$$x_j + \omega_j < x_{j+1} - \omega_{j+1}, \quad j \in \{1, 2, \cdots, J-1\} \tag{22}$$

essentially guaranteeing that the corners do not overlap and that the line segments do not shrink to have empty domains.

This representation is given by

$$\widetilde{f}(x) = \begin{cases} \ell_0(x), & x < x_1 - \omega_1 \\ g_1(x), & x_1 - \omega_1 \le x < x_1 + \omega_1 \\ \ell_1(x), & x_1 + \omega_1 \le x < x_2 - \omega_2 \\ \cdots & \cdots \\ \ell_{J-1}(x), & x_{J-1} + \omega_{J-1} \le x < x_J - \omega_J \\ g_J(x), & x_J - \omega_J \le x < x_J + \omega_J \\ \ell_J(x), & x_J \le x \end{cases} \tag{23}$$

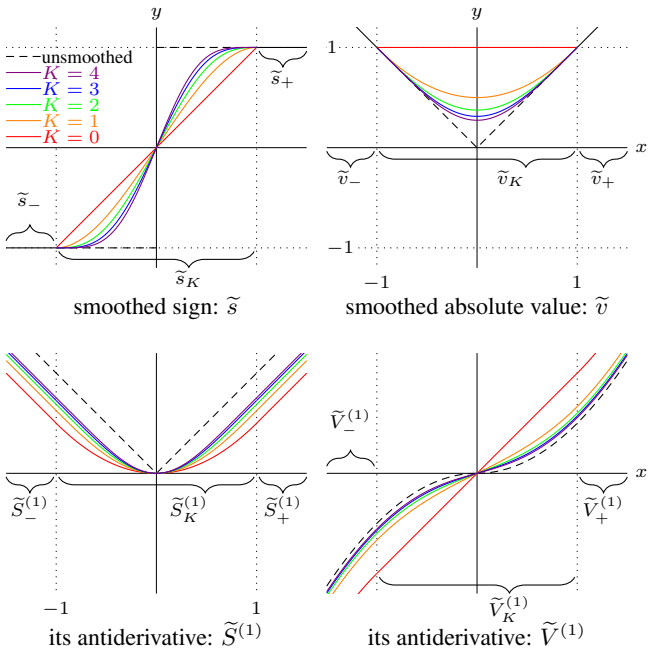


Figure 4: *Plots of the sign function* $\operatorname{sgn}(x)$ *and its smoothed variant* $\widetilde{s}_K(x)$ *(top left), the absolute value function* $|x|$ *and its smoothed variants* $\widetilde{v}_K(x)$ *(top right), as well as their antiderivatives* $\widetilde{S}_1^{(x)}$ *(bottom left) and* $\widetilde{V}_1^{(x)}$ *(bottom right), all for* $K \in \{0, 1, 2, 3, 4\}$.

where each line segment $\ell_j(x)$ is defined as in (15), and where, still without saying much about their shape, each rounded corner is given by an equation of the form [18]

$$g_j(x) = \beta_j + \mu_j x + \alpha_j \omega_j \, \widetilde{v}_j\left(\tfrac{x - x_j}{\omega_j}\right) + \sigma_j \, \widetilde{s}_j\left(\tfrac{x - x_j}{\omega_j}\right) \quad (24)$$

where

$$\beta_j = g_j(0) - \mu_j |x_j| = \begin{cases} y_j^- - m_j x_j, & x_j < 0 \\ \frac{y_j^+ + y_j^-}{2}, & x_j = 0 \\ y_{j-1}^+ - m_{j-1} x_j, & 0 < x_j \end{cases} \quad (25)$$

$$\mu_j = \frac{m_j + m_{j-1}}{2}, \qquad \alpha_j = \frac{m_j - m_{j-1}}{2} \quad (26)$$

$$\sigma_j = \frac{y_j^+ - y_j^-}{2}, \quad j \in \{1, 2, \ldots, J\} \, . \quad (27)$$

where each slope m_j is defined as in (17).

$\widetilde{v}(x)$ is a "smooth" approximation of the absolute value

$$\widetilde{v}(x) \approx |x| = \begin{cases} -x \, , & x \le 0 \\ x \, , & 0 \le x \end{cases} \quad (28)$$

and $\widetilde{s}(x)$ is a "smooth" approximation of the sign function

$$\widetilde{s}(x) \approx \operatorname{sign}(x) = \begin{cases} -1 \, , & x < 0 \\ 0 \, , & x = 0 \\ 1 \, , & 0 < x \end{cases} . \quad (29)$$

In the case where $\widetilde{v}(x) := |x|$ and $\widetilde{s}(x) := \operatorname{sign}(x)$, the representation (24) would be identical to the standard PWL representation (with no rounded corners). The idea of decomposing a corner with a possible jump continuity into a constant, linear, scaled absolute value, and scaled sign function comes from the "canonical piecewise linear representation" literature [28]. The advantage of performing this decomposition is that we only need to deal with defining $\widetilde{v}(x)$ and $\widetilde{s}(x)$ once, rather than defining a unique new function for each corner, and can then shift and scale them appropriately using the α_j, σ_j, ω_j, and x_j parameters used in (24)–(27). The resulting constants for the example piecewise linear curve (Fig. 3) are tabulated in Tab. 2. Again, we have $\omega_j = \omega, \forall j$ for these curves, where Fig. 3 shows plots for three values of ω whereas the values in Tab. 2 are calculated for the case of $\omega = 0.25$.

5. DERIVATIONS FOR SMOOTHED CORNERS

We could imagine defining $\widetilde{v}(x)$ and $\widetilde{s}(x)$ in many ways. Here, we will propose a specific way of defining these functions based on polynomials that match the values and one or more derivatives of the linear functions they are spliced to.

Now we will give details on how to derive functions that are suitable for $\widetilde{v}(x)$ and $\widetilde{s}(x)$. The key concept here is that these functions are both piecewise polynomials, comprising three segments each, with the outer two segments as straight lines and the inner segment, defined on the open interval $x \in \,]-1, 1[$, is a polynomial that is designed to match the value and a certain number of derivatives of the outer segments at the splice points $x = \pm 1$.

The smoothed absolute value function is

$$\widetilde{v}(x) = \begin{cases} \widetilde{v}_-(x) = -x, & x \le -1 \\ \widetilde{v}_K(x), & -1 \le x \le 1 \\ \widetilde{v}_+(x) = x, & 1 \le x \end{cases} \quad (30)$$

and the smoothed sign function is

$$\widetilde{s}(x) = \begin{cases} \widetilde{s}_-(x) = -1, & x \le -1 \\ \widetilde{s}_K(x), & -1 \le x \le 1 \\ \widetilde{s}_+(x) = 1, & 1 \le x \end{cases} . \quad (31)$$

$\widetilde{v}(x)$ and $\widetilde{s}(x)$ with no subscripts refer to the entire smoothed absolute value and sign functions, whereas the subscripted versions $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ refer to the inner polynomial segments. The subscript K refers to the degree of smoothness—Our goal will be to choose the polynomial coefficients so that $\widetilde{v}(x)$ and $\widetilde{s}(x)$ have a specified degree of differentiability C^K. Specifically, C^k smoothness means that all derivatives up to order k exist and are continuous. Since $\widetilde{v}_-(x)$, $\widetilde{v}_K(x)$, $\widetilde{v}_+(x)$, $\widetilde{s}_-(x)$, $\widetilde{s}_K(x)$, and $\widetilde{v}_+(x)$ (and all polynomials) all have C^∞ smoothness, this amounts choosing the $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ coefficients so that their values and first K derivatives match those of $\widetilde{v}_\pm(x)$ and $\widetilde{s}_\pm(x)$ at $x = \pm 1$. For a given degree of smoothness K, these constraints are:

$$\widetilde{v}_K(\pm 1) = \widetilde{v}_\pm(\pm 1) \quad = 1 \quad (32)$$

$$\frac{\mathrm{d}}{\mathrm{d}x}\widetilde{v}_K(\pm 1) = \frac{\mathrm{d}}{\mathrm{d}x}\widetilde{v}_\pm(\pm 1) \quad = \pm 1 \quad (33)$$

$$\frac{\mathrm{d}^k}{\mathrm{d}x^k}\widetilde{v}_K(\pm 1) = \frac{\mathrm{d}^k}{\mathrm{d}x^k}\widetilde{v}_\pm(\pm 1) = 0, \quad k \in \{2, 3, \cdots, K\} \quad (34)$$

$$\widetilde{s}_K(\pm 1) = \widetilde{s}_\pm(\pm 1) \quad = \pm 1 \quad (35)$$

$$\frac{\mathrm{d}}{\mathrm{d}x}\widetilde{s}_K(\pm 1) = \frac{\mathrm{d}}{\mathrm{d}x}\widetilde{s}_\pm(\pm 1) \quad = 0 \quad (36)$$

$$\frac{\mathrm{d}^k}{\mathrm{d}x^k}\widetilde{s}_K(\pm 1) = \frac{\mathrm{d}^k}{\mathrm{d}x^k}\widetilde{s}_\pm(\pm 1) = 0, \quad k \in \{2, 3, \cdots, K\} \, . \quad (37)$$

Table 3: *Tabulated coefficients for smoothed signum $\widetilde{s}_K(x)$ and absolute value $\widetilde{v}_K(x)$ for the lowest 7 values of K.*

K	N	s_1	s_3	s_5	s_7	s_9	s_{11}	s_{13}
0	1	1						
1	3	1.5	−0.5					
2	5	1.875	−1.25	0.375				
3	7	2.1875	−2.1875	1.3125	−0.3125			
4	9	2.4609375	−3.28125	2.953125	−1.40625	0.2734375		
5	11	2.70703125	−4.51171875	5.4140625	−3.8671875	1.50390625	−0.24609375	
6	13	2.9326171875	−5.865234375	8.7978515625	−8.37890625	4.8876953125	−1.599609375	0.2255859375
K	N	v_0	v_2	v_4	v_6	v_8	v_{10}	v_{12}
0	0	1						
1	2	0.5	0.5					
2	4	0.375	0.75	−0.125				
3	6	0.3125	0.9375	−0.3125	0.0625			
4	8	0.2734375	1.09375	−0.546875	0.21875	−0.0390625		
5	10	0.24609375	1.23046875	−0.8203125	0.4921875	−0.17578125	0.02734375	
6	12	0.2255859375	1.353515625	−1.1279296875	0.90234375	−0.4833984375	0.150390625	−0.0205078125

The polynomials are defined as

$$\widetilde{v}_K(x) = \sum_{\rho=0,2,\cdots}^{\Phi_\mathrm{e}} v_\rho x^\rho = v_0 + v_2 x^2 + \cdots + v_{\Phi_\mathrm{e}} x^{\Phi_\mathrm{e}} \tag{38}$$

$$\widetilde{s}_K(x) = \sum_{\rho=1,3,\cdots}^{\Phi_\mathrm{o}} s_\rho x^\rho = s_1 x + s_3 x^3 + \cdots + s_{\Phi_\mathrm{o}} x^{\Phi_\mathrm{o}} \,. \tag{39}$$

Because $|x|$ is an even function, $\widetilde{v}(x)$ and hence $\widetilde{v}_K(x)$ must be even functions, so all of the odd coefficients of $\widetilde{v}_K(x)$ are zero and its polynomial order Φ_e must be an even ("e") positive integer. Because $\mathrm{sign}(x)$ is an odd function, $\widetilde{s}(x)$ and hence $\widetilde{s}_K(x)$ must be odd functions, so all of the even coefficients of $\widetilde{s}_K(x)$ are zero, and its polynomial order Φ_o must be an odd ("o") positive integer.

In practice, since both $\widetilde{s}(x)$ and $\widetilde{v}(x)$ may contribute to the shape of the curve at each corner, we set their orders together by

$$\Phi_\mathrm{e} = 2K, \qquad \Phi_\mathrm{o} = 2K + 1 \,. \tag{40}$$

The first five ($K \in \{0, 1, 2, 3, 4\}$) instances of $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ are shown in Fig. 4. Note that the $K = 0$ case is only shown as an aid to understanding—it is not actually of any real use, since it is equivalent to just making a PWL with twice as many line segments, and does not actually increase the smoothness compared to the original PWL representation.

Our final task is to come up with the polynomial coefficients $v_0, v_2, \cdots, v_{\Phi_\mathrm{e}}$ and $s_1, s_3, \cdots, s_{\Phi_\mathrm{o}}$ of $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$.

5.1. Linear algebra

The simplest conceptual approach is to write all of the constraints from (35)–(37) and (32)–(34) as two systems of equations, which can each written as a single linear algebra equation and solved with standard methods (matrix inversion, etc.).

For example, the $\widetilde{s}_0(x)$ setup would be

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \,, \tag{41}$$

and the $\widetilde{s}_1(x)$ setup would be

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & -2 & 3 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} \,. \tag{42}$$

This process can be repeated to systematically form a linear equation of arbitrary order, by forming each row of the matrix as

$$\begin{bmatrix} 1 & (\pm 1) & (\pm 1)^2 & (\pm 1)^3 & \cdots & (\pm 1)^{\Phi_\mathrm{o}} \end{bmatrix} \left(\mathbf{D}^k \right)^\top \tag{43}$$

where the matrix $\mathbf{D}$ represents differentiation

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \Phi_\mathrm{o} - 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \,. \tag{44}$$

The whole system can then be solved as-is, or we can preferably enforce the odd symmetry of the smoothed sign function by eliminating the even columns (since the even coefficients are guaranteed to be zero) and eliminating the even rows (since the constraints at $x = -1$ are now redundant, since odd symmetry is enforced).

The process for the smoothed absolute value function is nearly identical, although the constraints on the right hand side will need to match those in (32)–(34) instead, and even symmetry is instead enforced by eliminating the odd columns.

In both cases, the coefficients $v_0, v_2, \cdots, v_{\Phi_\mathrm{e}}$ and $s_1, s_3, \cdots, s_{\Phi_\mathrm{o}}$ of $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ can be found with standard linear algebra, such as matrix inversion. Tabulated coefficients of $\widetilde{s}_K(x)$ and $\widetilde{v}_K(x)$ for $K \in \{0, 1, \cdots, 6\}$ are given in Tab. 3.

5.2. Closed-form

As an alternative to using linear algebra, we can used a closed-form equation that arises from a recursive construction. Robert Bristow-Johnson has discussed a soft clipper which is identical to our smoothed sign function [20–22].

In [21], Olli Niemitalo gives a useful recursive construction

$$\widetilde{s}_0(x) = x \tag{45}$$

$$\widetilde{s}_K(x) = \widetilde{s}_{K-1}(x) + \frac{(2K)!}{4^K (K!)^2} (1 - x^2)^K x \tag{46}$$

$$= \sum_{k=0}^{K} \frac{(2k)!}{4^k (k!)^2} (1 - x^2)^k x \,. \tag{47}$$

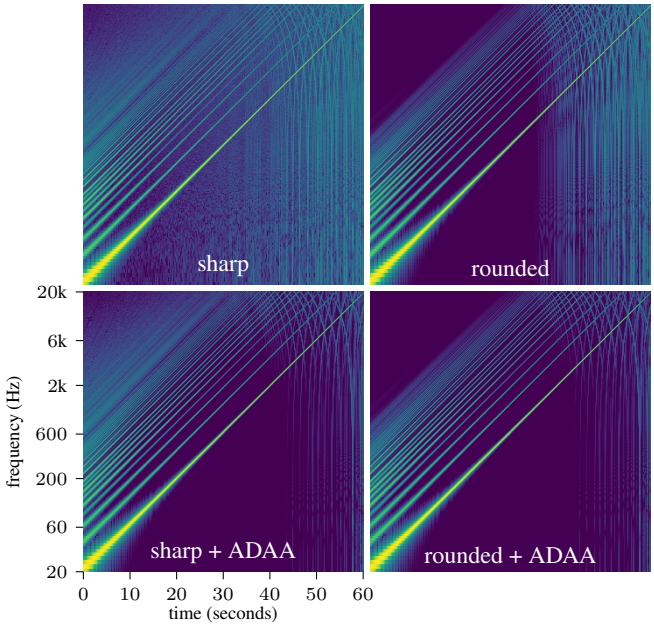


Figure 5: *Chirp response spectrograms showing the effect of rounding corners and/or applying first-order ADAA.*

The coefficients of $\widetilde{s}_K(x)$ are given in closed form by [21]

$$s_{2k+1} = \frac{(-1)^k(2K+1)!}{4^K K!(2k+1)k!(K-k)!},\ k \in \{0, 1, \cdots, K\}\ . \quad (48)$$

We can use this same technique to generate the smoothed absolute value function $\widetilde{v}_K(x)$ by recognizing that—in the same way that $|x|$ is an antiderivative of $\operatorname{sgn}(x)$—the smoothed absolute value function is itself an antiderivative of the smoothed sign function $\widetilde{s}_K(x)$, as can be seen in Fig. 4. This means that we can produce $\widetilde{s}_K(x)$ by first producing $\widetilde{v}_K(x)$ using Niemitalo's method and then integrating it. Since these are piecewise functions, we need to attend to the constants of integration in the same way that we discussed in §2. The only difference is that in the final step, we should not have $\widetilde{v}_K(0) = 0$, but rather $\widetilde{v}_K(\pm 1) = 1$.

5.3. Shifting polynomials

When we are not applying ADAA, we can shift and scale the normalized corner components $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ without any further effort, as in (24). However, when we are applying ADAA, we may prefer to deal with the raw polynomial coefficients directly. In this case, we need to be able to apply shifting and scaling operations to the polynomials. Vertically scaling a polynomial is trivial

$$Ap(x) = Ap_0 + Ap_1 x + Ap_2 x^2 + \cdots + Ap_\Phi x^\Phi\ . \quad (49)$$

Horizontal scaling is also fairly simple

$$p(Ax) = p_0 + Ap_1 x + A^2 p_2 x^2 + \cdots + A^\Phi p_\Phi x^\Phi\ . \quad (50)$$

Horizontal shifts are more complicated; we use the synthetic division based algorithm from [29].

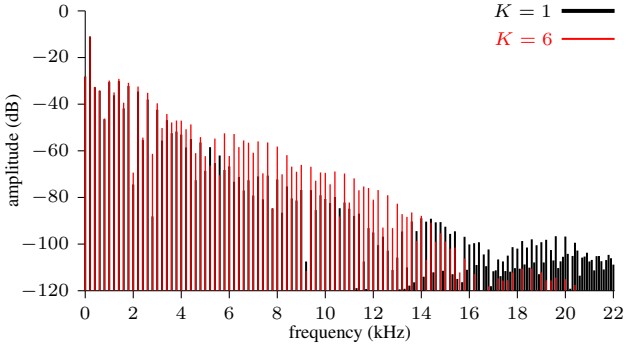


Figure 6: *Overtone amplitudes for* $K \in \{1, 6\}$ *for a* 5*-second-long* 200 *Hz sinusoid with an amplitude of* 2.0*.*

6. CASE STUDIES

We will now look at the chirp response for the PWL waveshaper shown in Fig. 3. Our input signal is a 60 second long logarithmic sinusoidal chirp from 20 Hz to 20 kHz, with an amplitude of 2.0 to drive it to hit all of the segments. Fig. 5 shows spectrograms of the chirp response of the "sharp" PWL waveshaper, a rounded version (all corners with $\omega = 0.25$), and versions of both with 1st-order ADAA applied. In all cases, the sampling rate is $f_s = 44100$ Hz (with no oversampling), and the curve smoothness is $K = 2$.

The "sharp" PWL waveshaper produces excessive aliasing, whereas the rounded version has far less, due to generally producing fewer overtones. For both the "sharp" PWL and rounded cases, applying ADAA is successful at eliminating most of the aliasing, coherent with results [11] for other waveshapes presented in the literature. In practice, due to the inherent filtering of ADAA, it should be used with at least $2\times$ oversampling, which will mitigate the filtering as well as adding extra alias suppression.

A second case study looks at what happens when the order of smoothness K is varied. Fig. 6 shows the strength of the overtones for a 5-second-long static 200 Hz sine wave with an amplitude of 2.0 as input, fed into the same waveshaper from shown in Fig. 3, again with $\omega = 0.25$, and with 1st-order ADAA applied. We can glean a few things from this plot. First, the amplitudes of the lower overtones are somewhat unaffected by K. Second, we can see that increasing K raises the amplitudes of the middle-frequency overtones. Third, increasing K decreases the amplitudes of the higher-frequency overtones.

7. CONCLUSION

In this paper, we've shown how to apply the ADAA technique to any piecewise polynomial waveshaper, including the crucial step of manipulating each segment's constant of integration to ensure at least C^0 smoothness of each antiderivative used. This greatly expands the class of waveshaping functions that can be used with ADAA, while also providing a template for how to handle other piecewise waveshaping functions. We looked specifically into two special cases: the classic case of piecewise linear waveshaping functions and a proposed technique for rounding the corners on a piecewise linear waveshaper using special polynomial corners that enforce smoothness up to a certain order.

For these smoothed corners, we've shown how to derive $\widetilde{s}_K(x)$

and $\widetilde{v}_K(x)$ for positive integer K, which enforces C^K smoothness on our rounded corners. If one wanted to use K as a smoothly controllable parameter, the discrete nature of the set of curves that make up $\widetilde{s}_K(x)$ and $\widetilde{v}_K(x)$ (as shown in Fig. 4) will be disappointing. However, there is an easy way to form a set of polynomials with continuous K with smoothness $C^{\lfloor K \rfloor}$,

$$\widetilde{s}_K(x) = (\lceil K \rceil - K)\widetilde{s}_{\lfloor K \rfloor}(x) + (K - \lfloor K \rfloor)\widetilde{s}_{\lceil K \rceil}(x)\,, \quad (51)$$

which would allow the corner shape to be varied smoothly.

For the case of the piecewise linear waveshaper with rounded corners, an obvious question arises: what order of smooothnes should we choose? Increasing the order of a polynomial waveshaper increases the number of overtones it produces, so it may seem that increasing the order of the smoothness would increase the number of overtones. However, in fact the *number* of overtones in a non-trivial piecewise polynomial waveshaper is infinite. So, reasoning about the number of overtones is not the right approach. Thinking instead of the relative amplitudes of the overtones, we saw in the second case study that increasing the smoothness did not greatly affect the low-frequency overtones, increased the amplitude of middle-frequency overtones, and decreased energy for the higher overtones. It is hard to say conclusively for all situations whether this is good or bad—What we can say for sure is that being able to control the spectral profile in this way is a useful timbral control that has implications for the amount of aliasing, since suppressing the amplitude of high-frequency overtones (beyond the Nyquist limit) reduces aliasing. One downside of increasing the order is that it can increases numerical error in the calculation of the coefficients and evaluation of the polynomials. In practice, algorithm designers will have to be careful not to create a situation where this numerical error, which manifests as harsh buzzing, is audible. So, for now, we can only recommend choosing the order to taste—a more rigorous evaluation and recommendation could be the subject of future work.

8. ACKNOWLEDGMENTS

Thanks to Russell McClellan and Evan Lynch for helpful talks.

9. REFERENCES

[1] F. Esqueda, H. Pöntynen, V. Välimäki, and J. D. Parker, "Virtual analog Buchla 259 wavefolder," in *Proc. Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, Sept. 5–9 2017, pp. 192–199.

[2] G. Gormond, F. Esqueda, H. Pöntynen, and J. D. Parker, "Waveshaping with Norton amplifiers: Modeling the Serge triple waveshaper," in *Proc. Int. Conf. Digital Audio Effects (DAFx-18)*, Aveiro, Portugal, Sept. 2016, pp. 288–295.

[3] F. Esqueda, H. Pöntynen, and J. D. Parker, "Virtual analog models of the Lockhart and Serge wavefolders," *Appl. Sci.*, vol. 7, no. 12, 2017, Art. #1328.

[4] H. Thornburg, "Antialiasing for nonlinearities: Acoustic modeling and synthesis applications," in *Proc. Int. Comput. Music Conf.*, Beijing, China, Oct. 22–27 1999, pp. 66–69.

[5] J. Pakarinen and D. T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Comput. Music J.*, vol. 33, no. 2, pp. 85–100, Summer 2009.

[6] A. Bernardini, K. J. Werner, P. Maffezzoni, and A. Sarti, "Wave digital modeling of the diode-based ring modulator," in *Proc. 144th Audio Eng. Conv.*, Milan, Italy, May 2018, pp. 66–69.

[7] K. J. Werner, V. Nangia, A. Bernardini, and J. O. Smith III A. Sarti, "An improved and generalized diode clipper model for wave digital filters," in *Proc. 139th Audio Eng. Conv.*, New York, NY, Oct.–Nov. 2015.

[8] D. J. Gillespie and S. Schachter, "Model bending: Teaching circuit models new tricks," in *Proc. Int. Conf. Digital Audio Effects (DAFx20in22)*, Vienna, Austria, Sept. 2022.

[9] D. T. Yeh, D. Nolting, and J. O. Smith III, "Physical and behavioral circuit modeling of the SP-12 sampler," in *Proc. Int. Comput. Music Conf.*, Copenhagen, Denmark, Aug. 2007, pp. 299–302.

[10] J. Kahles, F. Esqueda, and V. Välimäki, "Oversampling for nonlinear waveshaping: Choosing the right filters," *J. Audio Eng. Soc.*, vol. 67, no. 6, pp. 440–449, June 2019.

[11] J. D. Parker, V. Zavalishin, and E. Le Bivec, "Reducing the aliasing of nonlinear waveshaping using continuous-time convolution," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 2016, pp. 137–144.

[12] D. Albertini, A. Bernardini, and A. Sarti, "Antiderivative antialiasing techniques in nonlinear wave digital structures," *J. Audio Eng. Soc.*, vol. 69, pp. 448–464, July/Aug. 2021.

[13] K. J. Werner, "An equivalent circuit interpretation of antiderivative antialiasing," in *Proc. Int. Conf. Digital Audio Effects (DAFx20in21)*, Vienna, Austria, Sept. 2021, pp. 17–24.

[14] M. Holters, "Antiderivative antialiasing for stateful systems," in *Proc. Int. Conf. Digital Audio Effects (DAFx-19)*, Brimingham, U.K., Sept. 2019.

[15] M. Holters, "Combined derivative/antiderivative antialiasing," in *Proc. Int. Conf. Digital Audio Effects (DAFx20in22)*, Vienna, Austria, Sept. 2022, pp. 86–93.

[16] P. P. La Pastina, S. D'Angelo, and L. Gabrielli, "Arbitrary-order IIR antiderivative antialiasing," in *Proc. Int. Conf. Digital Audio Effects (DAFx20in21)*, Vienna, Austria, Sept. 2021, pp. 9–16.

[17] F. G. Germain, "Periodic analysis of nonlinear virtual analog models," in *Proc. IEEE Work. Appl. Signal Process. Audio Acoust.*, New Paltz, NY, Oct. 2021, pp. 321–325.

[18] J.-N. Lin and R. Unbehauen, "Canonical representation: From piecewise-linear function to piecewise-smooth functions," *IEEE Trans. Circuits Syst.—I: Fundamental Theory Appl.*, vol. 40, no. 7, pp. 461–468, July 1993.

[19] A. Bernardini and A. Sarti, "Canonical piecewise-linear representation of curves in the wave digital domain," in *Proc. Europe. Signal Process. Conf.*, Kos, Greece, Aug.–Sept. 2017, pp. 1165–1169.

[20] R. Bristow-Johnson, "Family of soft clipping functions," post on `music-dsp@music.columbia.edu` mailing list, 2014, `https://music-dsp.music.columbia.narkive.com/1JNTzu03/family-of-soft-clipping-functions#post1`.

[21] R. Bristow-Johnson, "Monotonic, symmetrical soft-clipping polynomial," post on DSP Stack Exchange, Dec. 11 2016, `https://dsp.stackexchange.com/questions/36202/monotonic-symmetrical-soft-clipping-polynomial`.

[22] R. Bristow-Johnson, "Smoothstep sigmoid-like function: Can anyone prove this relation?," post on Math Stack Exchange, Apr. 24 2017, `https://math.stackexchange.com/questions/2249296/smoothstep-sigmoid-like-function-can-anyone-prove-this-relation`.

[23] D. S. Ebert, Ed., *Texturing and modeling: A procedural approach*, AP Professional, Boston, 1994.

[24] J. O. Smith III, *Physical Audio Signal Processing*, W3K Publ., 2010.

[25] S. Enderby and Z. Baracskai, "Harmonic instability of digital soft clipping algorithms," in *Proc. Int. Conf. Digital Audio Effects (DAFx-12)*, York, U.K., Sept. 2012.

[26] S. Bilbao, F. Esqueda, J. D. Parker, and V. Välimäki, "Antiderivative antialiasing for memoryless nonlinearities," *IEEE Signal Process. Lett.*, vol. 24, no. 7, pp. 1049–1053, July 2017.

[27] A. Carson, "Aliasing reduction in virtual analogue modelling," MSc. thesis, U. Edinburgh, Edinburgh, U.K., Sept. 2020.

[28] L. O. Chua and S. M. Kang, "Section-wise piecewise-linear functions: Canonical representation, properties, and applications," *Proc. IEEE*, vol. 65, no. 6, pp. 915–929, June 1977.

[29] A. G. Akritas and S. D. Danielopoulos, "On the complexity of algorithms for the translation of polynomials," *Computing*, vol. 24, pp. 51–60, Mar. 1980.

Oral Session 4:
Machine Learning

AUTOMATIC RECOGNITION OF CASCADED GUITAR EFFECTS

Jinyue Guo *

RITMO, Department of Musicology
University of Oslo
Oslo, Norway
jinyue.guo@imv.uio.no

Brian McFee

Music and Audio Research Laboratory, MARL
New York University
New York, USA
brian.mcfee@nyu.edu

ABSTRACT

This paper reports on a new multi-label classification task for guitar effect recognition that is closer to the actual use case of guitar effect pedals. To generate the dataset, we used multiple clean guitar audio datasets and applied various combinations of 13 commonly used guitar effects. We compared four neural network structures: a simple Multi-Layer Perceptron as a baseline, ResNet models, a CRNN model, and a sample-level CNN model. The ResNet models achieved the best performance in terms of accuracy and robustness under various setups (with or without clean audio, seen or unseen dataset), with a micro F_1 of 0.876 and Macro F_1 of 0.906 in the hardest setup. An ablation study on the ResNet models further indicates the necessary model complexity for the task.

1. INTRODUCTION

The task of guitar effect recognition is to build an algorithm that recognizes which kinds of effects are used in a given piece of guitar audio. It is common to see multiple, nonlinear effects cascaded to produce a rich guitar timbre. This makes it difficult to build an effective recognition algorithm. Additionally, unlike some other music information retrieval (MIR) tasks such as pitch tracking or music tagging, resources for guitar effect recognition are relatively limited. The lack of data and evaluation standards makes it difficult to standardize the setup.

As we review in section 2, previous works on guitar effect recognition have framed it as a typical classfication problem. However, most previous research tend to form the question as a multi-class but single-label classification task, which means the models either work on samples with single effects or consider a group of effects as one label. However, in practice, different types of linear or nonlinear effects are often cascaded to create the final output. A second challenge is that the audio samples only contain a single sound event, either a single pluck or a single sweep, which makes the performance unpredictable on common guitar recordings that usually contain more complex temporal and spectral components.

1.1. Our Contributions

In this paper, we constructed the task of guitar effect recognition as a multi-label classification task by applying two improvements: First, we created a workflow that renders arbitrary guitar recordings with all combinations of effects using `SoX`. Second, we adapted several Neural Network models that can handle the multi-label classification task. The models were modified and benchmarked with several task setups: using the original clean audio as a hint or not, and zero-shot classification on unseen data distributions. As a complement of this paper, source-code and result sheets are provided online[1][2].

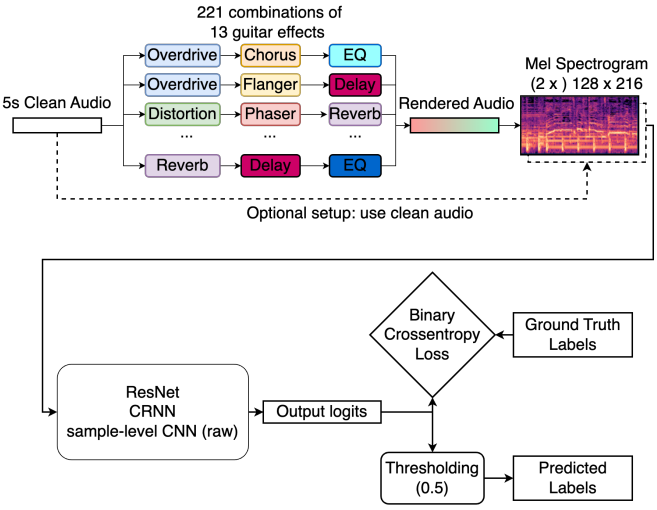


Figure 1: *Proposed method of effect rendering and multi-label classification.*

2. RELATED WORKS

The question of guitar effect recognition was first formed into a classification task by Stein et al. [1, 2]. They formed a single-label classification task by using only one effect at a time [1] or only classifying the group of effects [2]. The classifiers are built with hand-crafted audio features such as spectral centroid and cepstral features, and then feed into a Support Vector machine for classification. They also introduced a new dataset, *IDMT-SMT-Audio-Effects*. The dataset was manually processed using a Digital Audio Workstation, containing 55,000 samples of processed guitar audio files, recorded with two electric guitars and two electric basses. The samples are monophonic or polyphonic, but each includes a single sound event only.

Schmitt and Schuller [3] continued on the direction of hand-crafted audio features with a comprehensive research on features,

* This paper is based on the master's thesis research performed by the first author while studying at New York University

[1] https://github.com/fisheggg/SFXlearner
[2] DOI: 10.5281/zenodo.7973536

Table 1: *Effects used, the corresponding* `pysox` *function, and parameters. Parameters not listed are default values in* `pysox`.

Effect type	**Effect name**	**pysox function**	**parameters**
Non-linear	Overdrive	*sox.overdrive()*	gain_db: 5
	Distortion	*sox.overdrive()*	gain_db: 15
Modulation	Chorus	*sox.chorus()*	n_voices: 5
	Flanger	*sox.flanger()*	depth:5, phase:50
	Phaser	*sox.phaser()*	default
	Tremolo	*sox.tremolo()*	default
Ambience	Reverb	*sox.reverb()*	reverberance: 80
	Feedback delay	*sox.echos()*	n_echos: 3 delays: [200,400,600] decays:[0.4,0.2,0.1] gain_out:0.5
	Slapback delay	*sox.echo()*	n_echos: 3 delays: [200,400,600] decays:[0.4,0.2,0.1] gain_out:0.5
EQs	Low boost	*sox.bass()*	frequency: 200 gain_db: 10
	Low reduct	*sox.bass()*	frequency: 200 gain_db: -10
	High boost	*sox.treble()*	frequency: 8000 gain_db: 20
	High reduct	*sox.treble()*	frequency: 8000 gain_db: -20

including *zero-crossing rate* (ZCR), *root-mean-square* (RMS) energy, etc. They also introduced the concept of *Bag-of-Audio-Words* (BoAW), which generates a codebook of frame-level features. The classification setup and dataset are the same as Stein et al. [1, 2].

Neural network methods were first introduced to the task by Jürgens et al. [4] and Comunità, M. et al. [5]. Both of these works attempted to recognize not only the types of effects but also the parameters. Since the *IDMT-SMT-Audio-Effects* dataset did not provide any parameter information, both of these works have created their own setups. Jürgens et al. [4] continued to use the aforementioned effect settings [1] and built specific *Multi-Layer Perceptron* (MLP) [6] parameter regressor for each class after being classified. On the other hand, Comunità et al. [5] used a different setting. Instead of using the 10 classes [1], they focused on nonlinear effects. Thirteen overdrive, distortion and fuzz effect units are selected and tested on unified parameters (Gain and Tone/EQ). Instead of building specific regressors for each effect, a uniform *Convolutional Neural Network* (CNN) is trained to classify the effects and estimate their parameters.

Meanwhile, some of the latest neural network methods have been introduced to similar Music Information Retrieval tasks. The ResNet model [7] was originally proposed for Computer Vision tasks, but is proven to be also effective on genre recognition [8]. The CRNN model [9] combines the idea of convolutional neural network and recurrent neural network, achieved great results on music classification. Unlike the previous models that use spectrogram as their input, the Sample-Level CNN model [10, 11] applied 1-D convolution on raw waveforms and achieved great results on music auto-tagging. These models have been proven on solving audio-based multi-label classification problems, but are not adapted to guitar effect recognition yet.

3. DATA PREPARATION

3.1. Clean Datasets

Instead of recording sounds from physical guitar effect units, we chose to use clean guitar audio datasets, and manually render audio effects. There are two main benefits of this process. The first is to have huge flexibility in controlling the types, numbers, orders, and parameters of the effects we apply to each audio. The richness of data variation can contribute to the robustness of the model. The second benefit is that we can get a clean version of the processed audio, which can be used as a reference in the model.

We use two different clean guitar datasets in our rendering progress. `GuitarSet` [12] contains more than 10,000 seconds of guitar audio recordings, played by 6 different players in 5 different styles. The variety of this dataset can make our model robust on the timbre difference of the guitars and play styles of different guitarists. However, the instrument and recording setting of `GuitarSet` is fixed. As a complement, the fourth subset of `IDMT-SMT-Guitar` dataset [13] contains 384 guitar samples of 64 different music pieces that are played in 2 different tempi and 3 different guitar models, and recorded with two different setups. The audio files are sliced into 5-second excerpts with the tails dropped. After the process, we have 2004 samples from `GuitarSet`, and 650 samples from `IDMT-SMT-Guitar`.

3.2. Effect Selection and Dataset Rendering

We use `SoX` [14] and its python wrapper `pysox` [15] to create the pipeline and render effects to the clean datasets. We chose 13 often-used sound effects, including nonlinear effects, modulation, and EQs. Compared to the standard effect types proposed by Stein et al. [1], we added EQ effects since they are often used

in pedalboards. The *vibrato* effect is replaced by *chorus*, since it could be considered as a special case of *chorus* when the number of voices equals to 1. One limitation here is that `SoX` only provides one clipping algorithm, hence the *Overdrive* and *Distortion* actually use the same function but with different values of gain. The parameters of effects are hand-picked to provide audible changes to the audio samples and are fixed during the generation process. Table 1 shows the selected effects.

Using the `pysox` API, we can create effect chains that contain combinations of effects with various numbers and orders. The labels are generated as a multi-hot vector in length 13, each element indicates the presence of one effect. For example, an array $[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$ indicates an effect chain with overdrive, chorus, and high-reduct EQ. The labels are used as the ground truth during training and evaluation. We further split the effects into 6 different groups. Effects within a group are mutually exclusive: only one effect in the group is chosen in one effect chain. This is to avoid cancellation, such as applying Low boost and Low reduct at the same time. Table 2 shows the grouping.

Table 2: *Effect groups.*

Group number	Effect name
1	Overdrive
	Distortion
2	Chorus
	Flanger
	Phaser
	Tremolo
3	reverb
4	Feedback delay
	Slapback delay
5	Low boost
	Low reduct
6	High boost
	High reduct

During generation, we can choose the number of groups to apply, and the pipeline iterates over all combinations of groups, and combinations of the effects. Let E_n denote the total number of effect chains when applying n groups of effects, G_k denote the number of effects in the k-th group, and C_k^n denote a k-combination with n elements, we can calculate E_n using equation 1.

$$E_n = \sum_{g_i \in \{C_6^n\}} \sum_{k \in g_i} C_{G_k}^1 \tag{1}$$

For example, when $n = 2$, it iterates over the combinations of two groups: (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6), (4, 5), (4, 6), (4, 5). Within each set of groups, the combinations of effects are also iterated. For example in set $(5, 6)$, the combinations are (*Low boost*, *High boost*), (*Low boost*, *High reduct*), (*Low reduct*, *High boost*), (*Low reduct*, *High reduct*).

Each 5-second clean sample are rendered with all combinations of effects. We choose to use $n = [1, 5]$ in our generation process, where $n = 1$ is the case of single effect recognition and $n = 5$ means only one group is not used. As a result, 221 combinations of effects, 442,884 samples from `GuitarSet` and 143,640 samples from `IDMT-SMT-Guitar` are created. We further split `GuitarSet` samples into a training split of 327,522 samples and a validation split of 115,362 samples. We made sure the samples from the same audio source are in the same split to avoid the 'album effect' [16]. The `IDMT-SMT-Guitar` samples are only used during evaluation.

4. EXPERIMENTAL SETUP

We form two different setups: using the rendered audio with the clean audio ('with_clean'), or using the rendered audio only ('no_clean'). The 'with_clean' setup is a simpler task, since the model can compare the difference between the two signals and ignore unrelated variables, such as the type of guitar or the recording settings. Additionally, to compare with previous works on single-effect recognition, a reduced dataset with only $n = 1$ is used as a simpler setup.

To compare with previous works that use hand-crafted features, we use a traditional *Multi-Layer Perceptron* (MLP) model with *Mel-Frequency Cepstral Coefficients* (MFCC) [17] as input. The MFCCs are extracted from audio samples at $44,100$Hz with $n\_mfcc = 20$, $n\_fft = 4096$ and $hop\_length = 2048$. The MLP has three hidden layers with $hidden\_dim = 4096, 512, 13$ for each layer, and a ReLU function after each layer. The input shape of MLP is $2 \times 20 \times N_f$ for 'with_clean', and $20 \times N_f$ for 'no_clean', where $N_f = 108$ is the number of frames. As for the DC coefficient in MFCC, since it represents the overall volume of the audio, the MLP should be able to learn by itself whether the volume information is related to output labels.

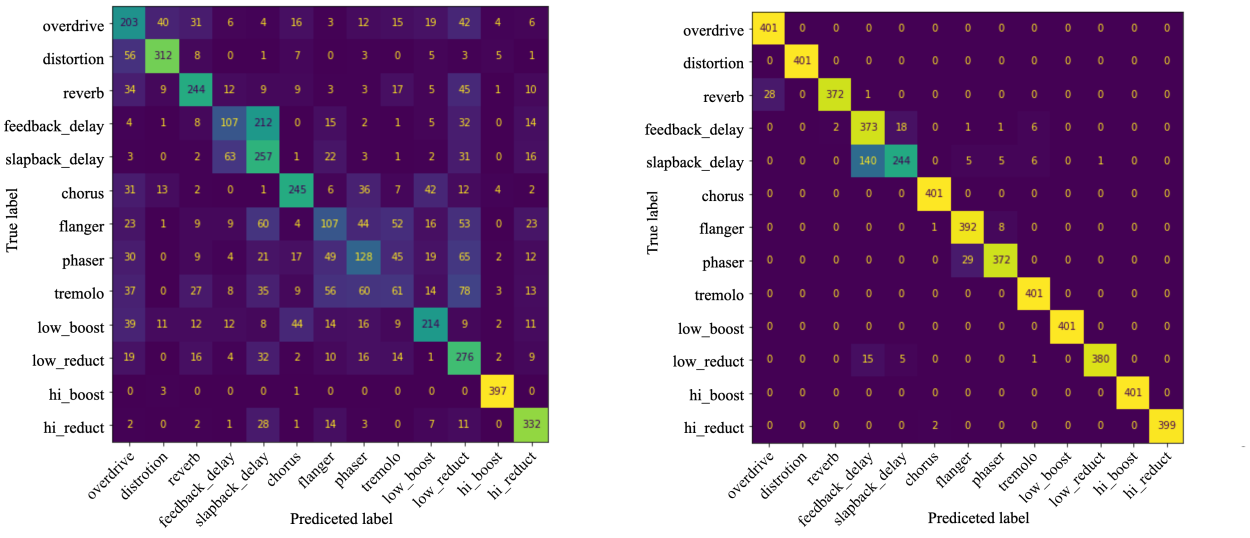
On the other hand, three newly developed Neural Network models are used: the *ResNet* model [7], the *CRNN* model [9] and the *Sample-Level CNN* model [10]. We use Mel spectrogram in dB scale as the input feature map for *ResNet* and *CRNN*, extracted from audio samples at $44,100$Hz with $n\_fft = 2048$, $n\_mels = 128$, $hop\_length = 1024$. The shape of one spectrogram is $(128, 216)$, with each time frame containing the information of 23.2ms. For the 'no_clean' task, the number of channels simply equals 1, while for 'with_clean' it becomes 2. The *Sample-level CNN* model uses raw audio downsampled to $22,050$Hz as input. The kernels and layers are slightly modified to suit the change of input and output size of our task.

The Models are built and trained using the `PyTorch` library [18]. For single-effect recognition, Categorical Cross-Entropy loss is used, while Binary Cross-Entropy loss is used for multi-effect recognition. Model weights are updated using Adam optimizer [19] with a learning rate of 0.001 for a maximum of 500 epochs. The batch size is set to 64 for baseline, *ResNet* and *CRNN*, and 16 for *sample-level CNN* due to memory limitation. The training set is shuffled before each epoch, but the random seed is fixed. An early stopping mechanism is performed if the validation loss over `Guitarset` validation split does not decrease for 2.5 epochs. For inference, the threshold of 0.5 is used. A computer with 16G RAM and one NVIDIA 2060 super GPU is used for training.

5. EVALUATION

5.1. Single-Effect Results

Table 3 shows the F_1 scores of the baseline model and *resnet18* model under the two setups. The *resnet18* model has outperformed the baseline model in both setups. For the single effect setup, the *'with_clean'* setup is relatively simple and both models achieved good performance. However, when the clean signal is removed,



(a) *no_clean setup.*

(b) *with_clean setup.*

Figure 2: *Single-effect confusion matrices of the baseline model under two setups, both on* `GuitarSet` *validation split.*

the baseline model has a significant performance drop, while *resnet18* can still achieve a score above 0.9.

Table 3: *Result of single effect recognition, evaluated on* `GuitarSet` *test split.*

Model	Setup	F_1 score
baseline	with_clean	0.956
resnet18	with_clean	0.999
baseline	no_clean	0.742
resnet18	no_clean	0.924

Figure 2 shows the single-effect confusion matrices of the baseline model under two setups. The characteristics of each effect class are different since they have different mechanisms. In the *'no_clean'* setup, *overdrive* and *distortion* are often confused, the reason might be that we actually used the same `SoX` function, only with different parameters. Since the data samples have different loudness levels, the clipping ratio varies according to the sample's own loudness level, which might cause the confusion. The *feedback delay* and *slapback delay* are similar, while the former has a feedback loop that also delays the previous outputs. More samples of *feedback delay* are classified as *slapback delay*, while fewer *slapback delay* are classified as *feedback delay*. The reason might be for samples without explicit note onsets, the differences between these two delays are too subtle for *MFCC*s to catch. Another confused group is *flanger*, *phaser*, and *tremolo*. Interestingly, it is the *tremolo* but not the *chorus* that is misclassified, since *tremolo* is amplitude modulation while *chorus*, *flanger* and *phaser* are all delay-based modulation. Since the baseline model is a plain MLP with no inductive bias for shift invariance, it did not capture the difference between delay modulation and amplitude modulation, but instead captured the difference in modulation patterns. In the `pysox` implementations, the Low Frequency Oscillators (LFO) are sinusoids in *flanger*, *phaser* and *tremolo*, while for *chorus* the LFO shape of each voice is randomly chosen between triangular and sinusoidal.

With the help of the clean signal, the model gained better performances in most of these groups. The model performed perfectly on *overdrive* and *distortion* when it knows the original loudness level of the signal. The volume information also helps to classify *tremolo* successfully from the other modulations. However, the model still struggles with *feedback delay* and *slapback delay*, which probably indicates that the MFCCs cannot catch such subtle timbre differences. There are still errors between *flanger* and *phaser*, which is more difficult when the audio sample only contains short impulses.

5.2. Multi-Effect Results

For a multi-label classification problem, we use both micro F_1 score and macro F_1 score as our evaluation metric. Figure 3 shows the benchmark of four models under two setups, two datasets. Among all the setups, the *resnet18* and *CRNN* model outperforms the baseline model under every setup, while the sample-level CNN only achieved a better micro F_1 score in one setup. The difference between micro F_1 and macro F_1 is relatively small for all models except sample-level CNN, which indicates that the performance of sample-level CNN is imbalanced between classes.

We can see a performance drop on `IDMT-SMT-Guitar` compared to the `GuitarSet` validation split. This indicates that even though the 'album effect' is avoided when splitting *GuitarSet*, the two splits are still inherently correlated. The F_1 scores of *CRNN* and *resnet18* on `Guitarset` validation split even reached 0.999 under 'with_clean' setup. Therefore, using `IDMT-SMT-Guitar` as the evaluation dataset is more meaningful.

In the hardest and most practical setup (`IDMT-SMT-Guitar`, 'no_clean'), the *resnet18* model achieved the best performance, while *CRNN* is slightly below. The other two models were able to solve the problem in the simpler setups, but failed in this practical setup. However, although the performance of *resnet18* and *CRNN* are similar in the hardest setup, we noticed that *resnet18* converges

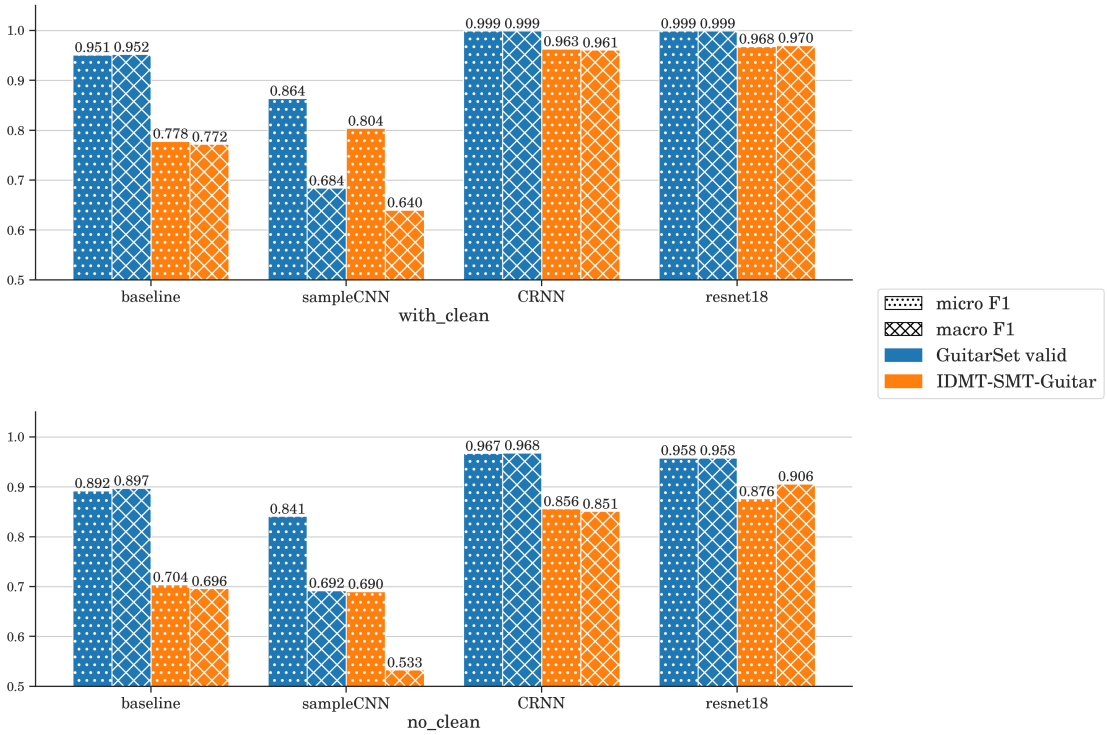


Figure 3: *Model benchmark of multi-effect recognition.*

much faster than *CRNN* during training.

Although the performance on `GuitarSet` validation split cannot directly represent the model's actual performance, we can analyze the robustness of models by comparing their differences in performance among two datasets. In both setups, the baseline model has the biggest performance drop on `IDMT-SMT-Guitar` dataset, while *resnet18* achieves the best robustness among the four models. All models had a bigger drop in 'no_clean' since the task is more difficult.

5.2.1. *Per-class analysis*

We used the best performing model, *resnet18* to analyze the result for each effect class. Figure 4 shows the confusion matrices of *resnet18* model under two setups. The performance under 'no_clean' setup is acceptable in most of the classes, except *overdrive* and *distortion*. The model achieved F_1 scores around 0.93 for these two classes on `GuitarSet` validation split, but dropped significantly on `IDMT-SMT-Guitar` while other classes achieved the same level of performance among two datasets. The reason might be that we used the same `SoX` implementation for these two classes, only with different gain levels. When testing on another dataset with a different distribution of loudness levels, the distribution compression rate also varies significantly. On the other hand, when clean audio is given to the model in 'with_clean' setup, it is easier to learn the relationship between input and output gain, and the model achieved a better performance on the two classes. Other classes benefit from the additional information as well.

5.3. Ablation study

We performed an ablation study on *resnet* by removing groups of convolutional layers at the output end. Since *resnet18* already performed well under both setups and both datasets, we want to investigate the minimum number of layers required to get such performance. The ablation models are trained using the same configuration as described in Section 4. For each ablation model, the last group of convolutional layers is removed, and the number of layers decreases by 4. Metrics are evaluated on `IDMT-SMT-Guitar` dataset.

Table 4: *Model complexity and performance for resnet ablations. Metrics are evaluated on* `IDMT-SMT-Guitar`.

Model	Setup	Number of parameters (k)	Micro F_1	Macro F_1
resnet18	with_clean	714.7	0.968	0.970
resnet14	with_clean	188.3	0.963	0.955
resnet10	with_clean	56.2	0.958	0.950
resnet6	with_clean	34.4	0.926	0.917
resnet18	no_clean	714.3	0.876	0.906
resnet14	no_clean	187.9	0.848	0.832
resnet10	no_clean	55.6	0.860	0.844
resnet6	no_clean	34.0	0.830	0.811

Table 4 shows the complexity and performance of the ablation models. The performance drop has a nonlinear relationship with respect to the number of parameters. We see that *resnet14* and *resnet10* have similar performance, and *resnet10* even slightly outperformed *resnet14* under 'no_clean' setup. On the other hand,

(a) *'with_clean'* setup

(b) *'no_clean'* setup.

Figure 4: *Confusion matrices and class-wise F1 scores of resnet18, multi-effect* `IDMT-SMT-Guitar`.

resnet6 has a performance drop of about 0.03 under both setups. Generally, the performance under 'no_clean' setup is a fair trade-off between complexity and performance. For the 'with_clean' setup, the smallest model is still able to perform relatively well, since the task is much simpler with the clean signal.

6. CONCLUSION

In this paper, we reconstructed the task of guitar effect recognition to make it closer to the actual use-case of guitar effect pedals. We created a workflow that renders arbitrary guitar recordings with all combinations of effects using `SoX`, which increases the variation compared to existing datasets. Secondly, we adapted novel Neural Network models to solve the multi-label classification task under two setups, 'with_clean' and 'no_clean'. The best performing model is modified from the *resnet18* model, achieving a micro F_1 of 0.876 and macro F_1 of 0.906 on an unseen dataset under 'no_clean' setup. Class-wise analysis indicates that the model is able to distinguish most of the effects except a small confusion on *overdrive* and *distortion*. An ablation study shows that the deep structure is necessary to achieve the performance, but a trade-off between complexity and performance is optional. Possible improvements include randomizing the effect order and parameters, or using better effect plugins, even real effect units.

7. REFERENCES

[1] Michael Stein, Jakob Abeßer, Christian Dittmar, and Gerald Schuller, "Automatic detection of audio effects in guitar and bass eecordings," *Journal of the Audio Engineering Society*, May 2010.

[2] Michael Stein, Jakob Abeßer, Christian Dittmar, and Gerald Schuller, "Automatic detection of multiple, cascaded audio effects in guitar recordings," in *Proc. Digital Audio Effects (DAFx-10)*. Graz, Austria, Sep. 2010, pp. 4–7.

[3] Maximilian Schmitt and Björn Schuller, "Recognising guitar effects - which acoustic features really matter?," in *INFORMATIK 2017*. Chemnitz, Germany, 2017, pp. 177–190.

[4] Henrik Jürgens, Reemt Hinrichs, and Jörn Ostermann, "Recognizing guitar effects and their parameter settings," in *Proc. Digital Audio Effects (DAFx-20)*. Vienna, Austria, Sep. 2020.

[5] Marco Comunità, Dan Stowell, and Joshua D. Reiss, "Guitar effects recognition and parameter estimation with convolutional neural networks," *Journal of the Audio Engineering Society*, vol. 69, no. 7/8, pp. 594–604, Jul. 2021.

[6] Christopher M. Bishop, *Neural networks for pattern recognition*, pp. 116–161, Oxford university press, 1995.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, Nevada, USA, Jun. 2016, pp. 770–778.

[8] Dipjyoti Bisharad and Rabul Hussain Laskar, "Music genre recognition using residual neural networks," in *2019 IEEE Region 10 Conf. (TENCON)*. Kochi, India, 2019, pp. 2063–2068.

[9] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho, "Convolutional recurrent neural networks for music classification," in *IEEE Intl. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 2392–2396.

[10] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," in *Proc. 14th Sound and Music Computing Conf. (SMC)*. Espoo, Finland, Jul. 2017.

[11] Taejun Kim, Jongpil Lee, and Juhan Nam, "Sample-level cnn architectures for music auto-tagging using raw waveforms," in *2018 IEEE Intl. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, Alberta, Canada, 2018, pp. 366–370.

[12] Qingyang Xi, Rachel M. Bittner, Johan Pauwels, Xuzhou Ye, and Juan Pablo Bello, "Guitarset: A dataset for guitar transcription.," in *Proc. Intl. Society for Music Information Retrieval Conf. (ISMIR)*. Suzhou, China, 2018.

[13] Christian Kehling, Jakob Abeßer, Christian Dittmar, and Gerald Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters," in *Proc. Digital Audio Effects (DAFx-14)*, 2014.

[14] Chris Bagwell and SoX contributers, "Sox: Sound exchange, the swiss army knife of sound processing," 1991.

[15] Rachel Bittner, Eric Humphrey, and Juan Bello, "Pysox: Leveraging the audio signal processing power of sox in python," in *Proc. Intl. Society for Music Information Retrieval Conf. (ISMIR) Late Breaking and Demo Papers*. New York City, USA, Aug. 2016.

[16] Brian Whitman, Gary Flake, and Steve Lawrence, "Artist detection in music with minnowmatch," in *Neural Networks for Signal Processing XI: Proc. 2001 IEEE Signal Processing Society Workshop*, 2001, pp. 559–568.

[17] Beth Logan, "Mel frequency cepstral coefficients for music modeling.," in *Proc. Intl. Symposium on Music Information Retrieval (ISMIR)*. Plymouth, Massachusetts, USA, 2000, vol. 270, p. 11.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*. Vancouver, Canada, Dec. 2019, vol. 32.

[19] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] Corinna Cortes and Vladimir Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

NEURAL MODELING OF MAGNETIC TAPE RECORDERS

*Otto Mikkonen, Alec Wright, Eloi Moliner and Vesa Välimäki**

Acoustics Lab, Department of Information and Communications Engineering
Aalto University, Espoo, Finland
firstname.lastname@aalto.fi

ABSTRACT

The sound of magnetic recording media, such as open-reel and cassette tape recorders, is still sought after by today's sound practitioners due to the imperfections embedded in the physics of the magnetic recording process. This paper proposes a method for digitally emulating this character using neural networks. The signal chain of the proposed system consists of three main components: the hysteretic nonlinearity and filtering jointly produced by the magnetic recording process as well as the record and playback amplifiers, the fluctuating delay originating from the tape transport, and the combined additive noise component from various electromagnetic origins. In our approach, the hysteretic nonlinear block is modeled using a recurrent neural network, while the delay trajectories and the noise component are generated using separate diffusion models, which employ U-net deep convolutional neural networks. According to the conducted objective evaluation, the proposed architecture faithfully captures the character of the magnetic tape recorder. The results of this study can be used to construct virtual replicas of vintage sound recording devices with applications in music production and audio antiquing tasks.

1. INTRODUCTION

Magnetic recording has had a profound impact on the history of recorded music, providing a dramatic leap in the quality of the stored audio in comparison to the earlier direct-to-disk techniques. The advances in magnetic recording grounded practices such as multitrack and sound-on-sound recording within the industry, and as the technology matured and became cheaper, allowed for entire generations of professional and amateur musicians alike to experiment with these powerful production techniques. Reel-to-reel tape recorders, an example of which is shown in Fig. 1, have been largely replaced by digital recording techniques for sound capture and reproduction. However, the idiosyncrasies of the magnetic recording process are now used as a creative effect. This paper studies the imperfections of the magnetic recording process and emulates them digitally and with neural networks.

Virtual analog (VA) modeling is an area of digital signal processing with a rich lineage in the past decades, aiming at modeling analog audio devices and making the emulations available as software [1]. The techniques used for the modeling are traditionally divided into white-box, grey-box, and black-box methods, depending on the type of information used as the basis for the task.

* This work was supported by the Nordic Sound and Music Computing Network (NordForsk project number 86892).

Figure 1: *Akai 4000D reel-to-reel tape recorder.*

White-box techniques use exact knowledge of the underlying circuits to reconstruct the physics in the digital domain in order to match the observed behavior. Black-box techniques use observations collected from the target as the basis and optimize a general-purpose method to replicate the behavior. In grey-box modeling, a combination of white- and black-box methods is used. Over the last decade, advances in deep learning have given rise to their increased application also for VA modeling, with the techniques capable of exhibiting state-of-the-art performance in a number of different tasks [2, 3, 4].

While the physics underlying magnetic recording has been studied thoroughly in the past [5, 6], it has been applied for the purpose of digitally emulating the recording process only recently [7]. Pertinent to this topic is the lineage of work related to the modeling of tape delay devices, which has been covered more extensively [8, 1, 9]. In our work, we build up from earlier literature on emulating the sound of the magnetic tape, and propose a grey-box system for the emulation task. We consider the sound of the tape recorder to be build up of the nonlinear hysteretic magnetization of the tape medium, the filtering produced by the recording and playback heads, a fluctuating delay induced by the imperfections in the tape transport mechanism, the tape hiss, as well as the subtle nonlinearities and filtering of the input/output amplifiers. The proposed system uses a hybrid array of modern deep-learning techniques as the backbone for modeling these different aspects of the character.

The rest of this paper is organized as follows. Sec. 2 reviews the background theory regarding the effects of magnetic recording. Sec. 3 gives an overview of the system proposed for the modeling. Sec. 4 provides details concerning the data used to evaluate the proposed method, while Sec. 5 describes the training process for the different network architectures. The experimental procedure is divided into two sections depending on the type of data used for the evaluation: in Sec. 6, toy data collected from a virtual tape machine is used, while Sec. 7 uses data from a real machine. Sec. 8 concludes the work.

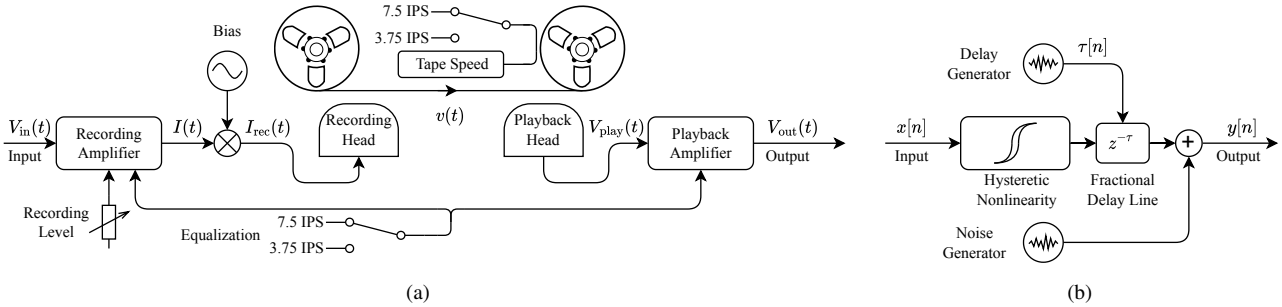


Figure 2: *(a) Real and (b) VA system block diagrams.*

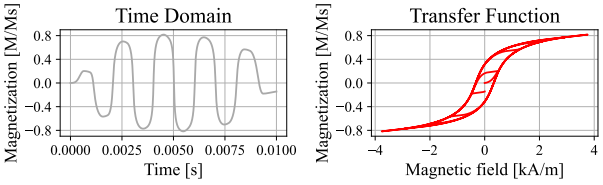


Figure 3: *Tape magnetization nonlinearity exhibits hysteresis.*

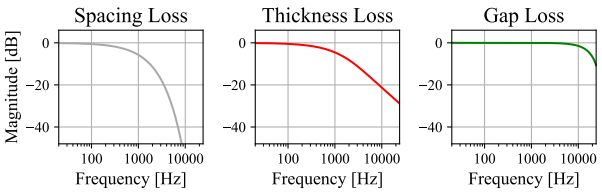


Figure 4: *Three components of playback losses.*

2. MAGNETIC RECORDING

The block diagram of a typical magnetic recorder is shown in Fig. 2a. The system consists of a recording amplifier, a recording head, the moving tape medium, a playback head, and a playback amplifier. The following paragraphs briefly discuss each of these components and their contribution to the overall character, based on earlier literature [7, 5, 6, 1, 8, 9].

The recording head takes the input current from the recording amplifier, and produces a spatial magnetic field determined by both the properties of the recording head and the magnitude of the input current. When the moving magnetic tape is exposed to this field, the magnetic dipoles in the substrate take the form of the field, which is retained as it passes the volume of the field. Since the characteristic magnetization of the tape takes a hysteretic nonlinear shape, shown in Fig. 3, a high-frequency bias is added to the input signal to reduce the nonlinearity produced by the hysteresis.

As the tape moves past the playback head, the changing magnetic field induces a current within its internal coil, restoring part of the stored signal into electrical form. This recovery of the signal is a spatial integration over the magnetized volume of tape, which leads to tape-speed-dependent filtering, shown in Fig. 4. The filtering consists of components related to the spacing between the head and the tape, the tape thickness, and the playback head gap.

To counterweight the spatial filtering induced by the playback head, the recording and playback amplifiers are used as pre- and post-filtering stages. This also serves to maximize the dynamic range of the tape and condition the input and output signals of the system. In our reference design shown in Fig. 1, these amplifiers are implemented with cascaded transistor circuits, producing additional low-order harmonic distortion in the processed signal.

The tape movement speed is not perfectly constant, due to small fluctuations produced by imperfections in the tape transport mechanics. These imperfections include cyclical components produced by the moving parts in the transport mechanism as well as stochastic behavior, shown in Fig. 5a, in the form of a delay trajectory. These inconsistencies in the movement can be heard as small fluctuations in pitch, known as *wow* and *flutter*.

Magnetic tape recorders generate noise artifacts throughout the recording and playback stages, which contribute to the distinctive qualities of the resulting audio signal. The noises could be originated by multiple sources, such as the playback equipment, the magnetic particles in the tape coating, modulation noises during recording, or surface asperities, among others [10].

3. METHODS

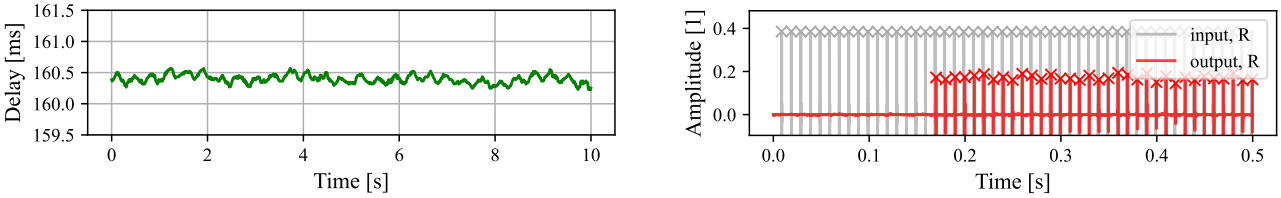
We employ a grey-box model, shown in Fig. 2b, inspired by the block diagram of the target device. The signal path for the proposed system consists of three components: 1) a hysteretic nonlinearity for modeling the magnetic recording process lumped with the record/playback amplifier responses, 2) a time-varying delay line controlled by a delay trajectory generator, and 3) an additive noise component. In the following, details concerning these components as well as the capture of delay trajectories are given.

3.1. Delay Trajectory Retrieval

The fluctuating time delay between the record and playback heads is captured using a pulse train-based measuring technique [8, 11, 7]. The measurement signal consists of a train of unit impulses spaced $T = 1/f$ apart, where f is the repetition frequency. This signal is played through the target device, and the locations of the input and output pulses are compared to determine the time delay between the heads as it fluctuates over time. The frequency f of the pulse train determines the sampling rate for the captured trajectories and in our experiments $f = 100\,\text{Hz}$ was used [8, 9]. An example of a measured delay trajectory, as well as the measurement signal at the input and output of a studied target are illustrated in Fig. 5.

3.2. Lumped Nonlinearities

The lumped effects include the hysteretic nonlinearity of the magnetization process, the spatial filtering of the playback head, as



(a) (b)

Figure 5: *(a) Delay trajectories and (b) measurement signal.*

well as the low-order distortion components and filtering originating from the record and playback amplifiers. We choose a recursive neural network (RNN) architecture for modeling these aspects, consisting of a gated recurrent unit and a linear output layer [2]. The model details can be found in the original research article. The choice of the model stems from the stateful nature of the recurrent unit, which we hypothesize as being helpful in learning the hysteresis shape. Three training schemes for training the RNN are considered: two supervised and one adversarial, explained in the following subsections.

3.2.1. Supervised Approaches

For the supervised approaches, we exploit the stereo nature of most tape recorders. We construct stereo training signals with the audio content on the left channel embedded with the measurement pulse train signal on the right. Using the pulse detection algorithm to construct the delay trajectories for each individual audio segment, the time evolution of the tape medium can be captured and used to restore the alignment of the targets and model predictions.

We implement and compare two approaches for restoring the alignment, shown in Fig. 6. In the first approach (Fig. 6a), inspired by Kaloinen's work [9], we use the captured delay trajectory $\boldsymbol{\tau}$ to demodulate the target segment $\mathbf{y}_L^* = \text{demod}(\mathbf{y}_L, \boldsymbol{\tau})$, where * denotes a signal which has been demodulated or has not had a delay line applied to it. After this procedure, the demodulated signal and the raw output from the nonlinear block become aligned and we compute the loss $\mathcal{L}(\hat{\mathbf{y}}_L^*, \mathbf{y}_L^*)$. In this scenario, the gradients flow only through the nonlinear block, and the time-varying delay line is only added during inference.

In the second approach, shown in Fig. 6b, we use the captured delay trajectory $\boldsymbol{\tau}$ to delay the raw output from the nonlinear block $\hat{\mathbf{y}}_L[n] = \hat{\mathbf{y}}_L^*[n - \boldsymbol{\tau}[n]]$ in order to compute the loss directly as $\mathcal{L}(\hat{\mathbf{y}}_L, \mathbf{y}_L)$. In this approach, the gradients flow also through the delay line, which needs to be differentiable. Linear interpolation was used to implement the continuously variable delay line, and the implementation was originally included in the Magenta DDSP codebase [12]. Care must be taken to avoid non-differentiable rounding operations such as the floor operator.

The delay line is initially filled from left to right with integers representing delay-line length, starting from N and decreasing with a step size of 1, with the final value being 0. The desired delay-line length is subtracted from each element of this vector and the abs operator is applied. Each element of this vector is subtracted from 1, and finally, the rectified linear unit (ReLU) function is applied, resulting in a vector where all indices are filled with zeros, except for the indices with indexes immediately above and below the desired delay-line length. This vector can then be multiplied element-wise with a buffer of previous input values, and the sum of this operation produces the output of the delay line.

3.2.2. Adversarial Approach

An alternative approach using an adversarial training method is also proposed. This allows for the modeling of monophonic tape recorders. In this method, measured or synthesized delay lines are applied to the RNN model output. Instead of training the model using a supervised loss function, which requires the delay line to be known, an adversarial loss can be used such that the delay line applied does not have to match the actual delay line applied in the training data. In this case, a discriminator model is trained to distinguish between real examples of processed audio from the target dataset, and synthetic examples which are produced using our modeling approach. The discriminator model and training procedure used are identical to those used earlier by Wright et al. [13]. The discriminator receives a time-frequency representation of the audio as input, and consists of a stack of 1D convolutional layers, with the first layer treating the frequency bins as an input channel. The discriminator and the tape model are trained adversarially using the hinge loss function.

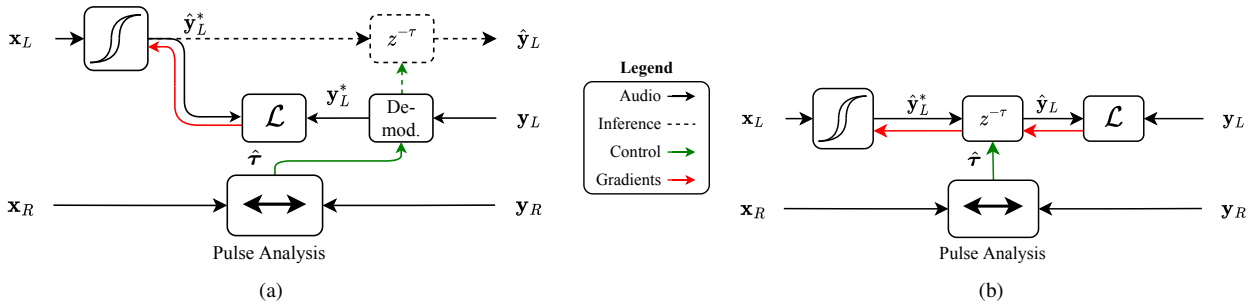
3.3. Noise Generator

The background noise component is modeled using a diffusion probabilistic model [14], in a similar fashion to previous work from Moliner and Välimäki [3]. A diffusion model is used as a data-driven universal approximator of the probability distribution of all the additive disturbances that are introduced during the recording, magnetization, and playback processes. The model can be trained with recorded silent passages containing only the background textures produced by the reel-to-reel machine. By reversing a diffusion process, white Gaussian noise segments are progressively morphed into noises from the training data distribution. Based on the assumption that the noises are additive, the generated noise samples are added to the output signal as a final step.

Although we have adopted the main concept from a previous work [3] as a basis for our research, there are significant deviations in the technical details of our approach due to our use of more recent developments on diffusion models. We adopt some of the design choices from Karras et al. [15], including the ordinary differential equation (ODE) formulation, the neural network preconditioning, the training objective, and the noise schedule parameterization, the latter being a Variance Exploding noise schedule [16].

3.4. Trajectory Generator

Given the stochastic nature of the delay trajectories underlined in Sec. 2, in this work, they are modeled using a probabilistic generative model. Similar to Sec. 3.3, a diffusion model is also used to for this task. In this case, the model is trained to emulate the distribution of the measured delay trajectories.



(a) (b)

Figure 6: *Supervised approaches for training the nonlinearity: (a) Demodulation and (b) differentiable delay line.*

4. DATA COLLECTION

This section provides details concerning the data used in the experimental procedure, including the compiled datasets. The compiled datasets are made available in the accompanying webpage [1].

As input data, we use a fraction of the inputs from SignalTrain [17] for training the nonlinear block. The dataset consists of short musical passages representing varying genres played using various instruments, together with synthetic measurement signals, sampled at 44.1 kHz. A total of 60 min, 20 min, and 15 min of audio is used for training, validation, and testing, respectively.

4.1. Toy Data

We test the modeling architecture using synthetic data, generated via wrapping the VST instance of CHOWTape [7], a white-box modeled tape machine, using Pedalboard [2]. During the generation, the VST instance is set to 16× oversampling using an 8-iteration Newton-Rhapson solver for the tape hysteresis ODE, which are the highest quality settings available. To make sure the virtual tape is sufficiently saturated, the tape drive, tape saturation, and tape bias are set to $(0.75, 0.75, 0.0) \in [0, 1]$, respectively. The timing parameters—the flutter depth, wow depth, and wow variance—are set to $(0.75, 0.75, 1.0) \in [0, 1]$, respectively. We turn off any additional processing from the VST which did not appear in the original research article. Two datasets are collected for the experiments with the toy data: one with only the tape effects enabled and the timing effects disabled, and one with both of the effects enabled.

4.2. Real Data

The real data for evaluating the modeling architecture was collected using an Akai 4000D open-reel tape recorder (Fig. 1). The device is a $\frac{1}{4}$ inch, four-track, three-head, stereo recorder from the 1970s, capable of running at $3\frac{3}{4}$ and $7\frac{1}{2}$ inches per second (IPS) and using discrete transistor circuitry for the input/output amplifiers. The data was collected using two types of magnetic tape: a Maxell low-noise/high-output tape and a Scotch low-noise tape from the 1970s. An RME Fireface UCX audio interface was used for recording and playback.

Using a three-head recorder allows for simultaneous recording and playback to and from the tape, allowing the fluctuating time delay between the record and playback heads to be captured. In practice, we connect a stereo line feed from the audio interface into the line inputs of the tape recorder, set monitoring to *TAPE*, and record both the stereo line feed from the tape recorder, as well as a loopback signal from the interface outputs back to its inputs. The recording level of the device was set such that when monitoring the signal entering the tape (monitoring set to *INPUT*), a 0 dBFS signal from the interface is just below the clipping threshold of the record and playback amplifiers.

The collected data is divided into datasets used for training the lumped nonlinearities, the delay trajectory generator, and the noise generator. We collected two versions of each dataset using the (tape branch, tape speed) configurations (MAXELL, $7\frac{1}{2}$ IPS) and (SCOTCH, $3\frac{3}{4}$ IPS). While initially the same datasets were intended to be used for training both the nonlinear block and the delay trajectory generator, studying the extracted trajectories from the lumped nonlinearity datasets revealed that the accuracy of the trajectory generator would be severely limited by the considered sampling rate of 44.1 kHz, and thus we collected separate high-resolution datasets at 192 kHz for the trajectory generator, consisting of the same upsampled audio. Finally, to train the noise generator, two datasets consisting of only the hiss captured from the line feed of the tape recorder were collected using the original sampling rate of 44.1 kHz.

5. IMPLEMENTATION DETAILS

This section provides details concerning implementing the different components in the modeling architecture, including modeling training and the loss functions used.

5.1. Supervised Approaches

The first two approaches use supervised training to optimize the weights of the nonlinear block, aligning the target and output segments using the proposed methods shown in Fig. 6. The RNN is trained using truncated back-propagation through time (TBPTT) [18], allowing the RNN state to initialize before tracking the gradients. For the first approach, the RNN state is initialized for 1024 steps, and for the second, the initialization length is determined by taking the next power of two of the maximum delay length in samples encountered in the training dataset. We use a hidden size of 64 as preliminary experiments indicated that increasing the hidden size beyond this did not bring an improvement in model performance. We use Adam with the default hyperparameters as implemented in PyTorch as the optimizer. We use a learning rate

[1] `http://research.spa.aalto.fi/publications/papers/dafx23-neural-tape/`

[2] `https://github.com/spotify/pedalboard`

of 1×10^{-3} and reduce it with a factor of 0.75 every time the validation loss has not improved for 10 epochs. A batch size of 32 is used for all the experiments and the models are trained using a graphical processing unit for 4 hours.

To compute the prediction discrepancy against the target output for the supervised approaches, the error-to-signal ratio (ESR) loss is used [19]. Details concerning the loss function can be found in the paper by Damskägg et al. [19].

5.2. Adversarial Approach

For the adversarial training method, TBPTT was also used. As the time-varying delay line is initially filled with zeros, the initialization stage is run for a number of steps equal to the maximum measured delay-line length. This ensures that the delay line is filled with real values during TBPTT. A segment length of 2 seconds was used during training, with parameter updates being carried out every 16384 samples. The longer TBPTT length was used as the discriminator model uses a time-frequency representation of the signal as input. As such, longer input lengths increase the frequency resolution that is seen by the discriminator model.

The discriminator and RNN model are alternately trained using the hinge loss function described by Kumar et al. [20]. A batch size of 16 was used during training. During validation, paired data was used to evaluate the RNN model, with the measured delay line being applied at the output of the RNN. Training was run for 50 epochs, with a multi-resolution log spectral magnitude loss being used to select the best performing model weights.

5.3. Noise Generator

We train our diffusion models following the recommendations by Karras et al. [15]. Considering that the standard deviation of the recorded data is, approximately, $\sigma_{\text{data}} = 8 \times 10^{-4}$, the noise schedule is defined so that, during sampling, the Gaussian noise level decreases logarithmically across the reverse diffusion process from $\sigma_{\text{max}} = 0.1$ (completely masking the data) to $\sigma_{\text{min}} = 5 \times 10^{-5}$ (perceptually insignificant). During training, the model is trained with the L2 preconditioned objective from [15], where the noise level is sampled randomly with a LogUniform distribution. The model is trained with the Adam optimizer with the default momentum hyperparameters and a learning rate of 2×10^{-4}. An exponential moving average of the weights with a decay factor of 0.999 is tracked during training and used as the final inference model.

A standard time-domain convolutional U-Net is used as the backbone deep neural network architecture, which is conditioned on a noise level embedding that allows weights to be shared across different noise levels. The total number of parameters adds to 127k, which is relatively low when compared to standard practice for diffusion models, but has proven qualitatively to be enough for this particular use-case. Inference is performed with a denoising diffusion implicit model (DDIM) [21] sampler with a subtle amount of stochasticity, which allows for a trade-off between sampling speed and quality by adjusting the number of sampling steps. We use 16 steps in our experiments, but we observe that this number can be reduced down to 6 without a significant quality loss.

The noise generator model is trained with samples of 1.5-s at the sampling rate of 44.1 kHz. However, given that the architecture is fully-convolutional, the segment size could be freely adapted during inference. In addition, arbitrarily long sequences can be generated by applying a chunked autoregressive sampling strategy, where separate frames are concatenated and inter-frame coherence can be ensured by applying a zero-shot outpainting technique [22].

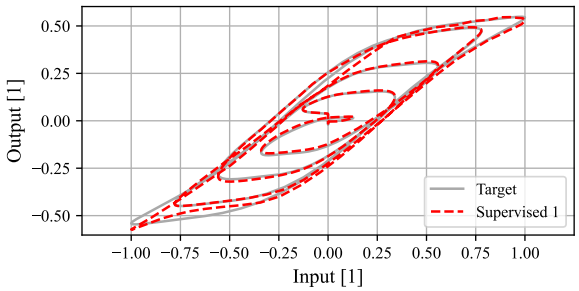


Figure 7: *Model hysteresis using toy data - Lumped nonlinearities only.*

5.4. Trajectory Generator

The delay trajectory generator is trained in a very similar way as the noise generator, specified in Sec. 5.3, while only some implementation details concerning the characteristics of the data differ. The model is trained with 5.2-s segments at a sampling frequency of 100 Hz, defined by the measuring pulse frequency (see Sec. 3.1), resulting on segments of 512 samples. During training, every delay trajectory segment is mean-normalized, leaving only the local fluctuations from the average delay as the distribution to be modeled. The backbone neural network architecture is also a standard convolutional U-Net with 77k trainable parameters; we refer to the source code for further details. The noise schedule design is motivated analogous to Sec. 5.3 and depends on the statistics of the dataset. For the toy data experiment (see Sec. 6), the approximated data standard deviation is $\sigma_{\text{data}} = 6.8 \times 10^{-3}$, and the noise schedule is designed between $\sigma_{\text{max}} = 0.5$ and $\sigma_{\text{min}} = 1 \times 10^{-5}$. For the real data experiment (see Sec. 7), the data standard deviation is approximately $\sigma_{\text{data}} = 1 \times 10^{-4}$, and we found that $\sigma_{\text{max}} = 0.01$ and $\sigma_{\text{min}} = 1 \times 10^{-5}$ were a suitable design choice. The sampling is performed with a 10-step DDIM sampler [21], but we observed that the number of discretization steps could be reduced down to only 4 with minimal qualitative differences.

6. EXPERIMENT 1: TOY DATA

This section presents experiments using synthetic data in two subsections. Sec. 6.1 studies the capability of the modeling method for the hysteretic magnetization of the tape without the fluctuating timing effects. Later, the timing effects are also enabled, allowing the evaluation of the three training schemes for the nonlinearity in Sec. 6.2 and the trajectory generator in Sec. 6.3. The toy data does not contain a noise component, however. Audio examples for these experiments are available on the accompanying web page [1].

6.1. Lumped Nonlinearities Only

To evaluate model performance, we use the ramped sine technique for the hysteresis [23]. Additionally, we encourage readers to listen to the example predictions on the accompanying web page[1]. The ramped sine technique is especially useful here, since it evaluates both the deadzone effect resulting from an under-biased tape and the saturation at higher amplitudes.

The model hysteresis versus the target is shown in Fig. 7. While the match is not perfect, the model clearly learns the shape

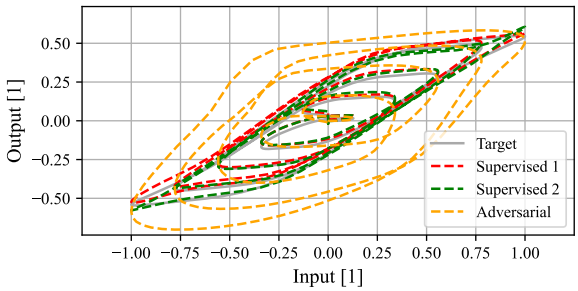


Figure 8: *Model hysteresis using toy data - Nonlinearities and timing effects.*

of the hysteresis loop, including the deadzone effect, the saturation of the tape, as well as the loop width. Listening and comparing the model predictions against the target further validates this finding, confirming the suitability of the RNN architecture for modeling the type of nonlinearity.

6.2. Lumped Nonlinearities and Timing Effects

The various training schemes are evaluated as in Sec. 6.1, on top of which the model losses over the test set are compared. Since the adversarial models are not trained with a time-domain loss, we include a multi-resolution short-time Fourier transform (STFT) loss [24] in the comparison. We use the default hyperparameters for the method as implemented in the Auraloss library [25]. Model predictions can be found in the accompanying webpage[1].

The hysteresis of the models trained using the three approaches versus the target is shown in Fig. 8. While it can be seen that the models trained using the supervised approaches match the shape of the hysteresis well, the hysteresis shape of the adversarially trained model deviates from the target. This can be explained by the adversarial model being trained on a time-frequency domain loss, which does not enforce strict time-domain matching, which is the case for the two supervised models.

The $\mathcal{L}_{\text{ESR}}$ and $\mathcal{L}_{\text{STFT}}$ losses over the test set are listed in Table 1, where the best (smallest) results are highlighted with bold font. The losses for the first supervised approach are computed either via using the real delay trajectories to demodulate the targets (*Demodulated*, similar to training) or applying the trajectories to the predictions (*Delayed*, similar to inference). As can be seen from the results, the second supervised approach produces the best overall performance across the considered metrics. While the time-domain loss $\mathcal{L}_{\text{ESR}}$ is two orders of magnitude higher for the adversarial approach than for the supervised approaches, their time-frequency domain losses $\mathcal{L}_{\text{STFT}}$ are of similar magnitude. For the first supervised approach, it can be seen that for both considered losses, the delayed computational method results in a smaller error. This finding suggests that demodulating the outputs can produce a larger error in comparison to delaying them with an interpolated delay line.

Table 1: *Toy Data: Nonlinearities and timing effects.*

	$\mathcal{L}_{\text{ESR}}$		$\mathcal{L}_{\text{STFT}}$	
Approach	Demod.	Delayed	Demod.	Delayed
Supervised I	0.031	**0.029**	0.645	0.536
Supervised II	–	**0.029**	–	**0.488**
Adversarial	–	1.567	–	1.772

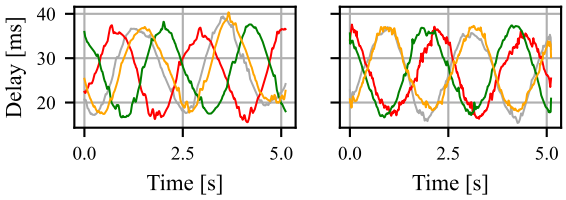


Figure 9: *Measured (left) and generated (right) delay trajectories using toy data.*

6.3. Trajectory Generator

We experiment with the diffusion model approach to generate the delay trajectories from the toy experiment, which have been synthesized as described in Sec. 4.1. Despite our best efforts to devise a methodology for objective evaluation, we found that the stochastic nature of the data and the complexities involved made it difficult to quantify its effectiveness in a reliable and consistent manner. As such, we rely on a merely qualitative assessment, which we believe provides sufficient evidence of the successful model capabilities. In Fig. 9, we show a qualitative comparison between measured trajectories and generated ones. In this case, 10 iterative steps were used to sample from the diffusion model, requiring 10 function evaluations of the neural network. The data contains a prominent sinusoidal component with some spurious artifacts, which seem to be accurately modeled by the diffusion model.

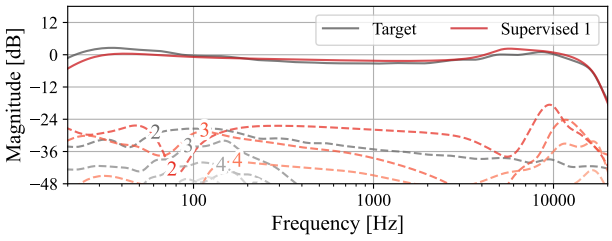
7. EXPERIMENT 2: REAL DATA

Next, the proposed method is evaluated using real data collected from the Akai 4000D tape recorder. Since the toy data did not include a noise component, this section serves as the first validation for the capability of the noise generator to learn the character of the media, as well as further validates the performance of the other two architectural components. Audio examples from the conducted experiments can be found on the web page[1].

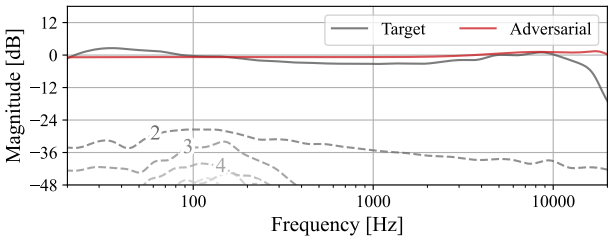
7.1. Lumped Nonlinearities and Timing Effects

We start with the same evaluation strategy as in Sec. 6.2, but find that the magnetic field produced by the recording head is not sufficient to saturate the considered tape formulations, as has been encountered earlier [1]. Thus, instead of comparing the hysteresis of the trained models, we focus on the learned magnitude response and nonlinear distortion components [26]. Only one configuration (MAXELL $7\frac{1}{2}$ IPS) is evaluated here for the sake of brevity. The model predictions can be found on the accompanying web page[1].

We find the responses of the supervised models similar, and only show the learned magnitude response and nonlinear distortion components versus the target for the first supervised approach in Fig. 10a. As can be seen, the model closely matches the shape of the linear response: the attenuated middle frequencies, the subtle emphasis of the highs, as well as the high and low-frequency roll-offs. While the target also portrays the head-bump effect in the low frequencies, this aspect is not matched by the model. We suspect that this might have to do with the dataset used for training the models not having enough low-frequency content to sufficiently learn this frequency band. While the model also learns to produce nonlinear distortion from the target data, the shape of the contours is not matched well, and the model starts to alias above



(a) *Supervised I.*



(b) *Adversarial.*

Figure 10: *Model magnitude responses (solid) and distortion components (dashed), MAXELL* $7\frac{1}{2}$ *IPS.*

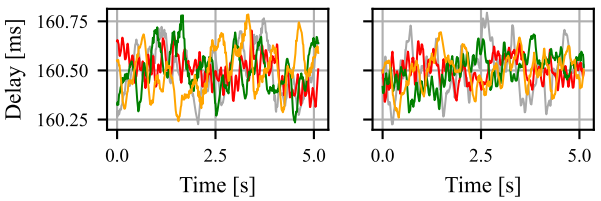


Figure 11: *Measured (left) and generated (right) delay trajectories using real data.*

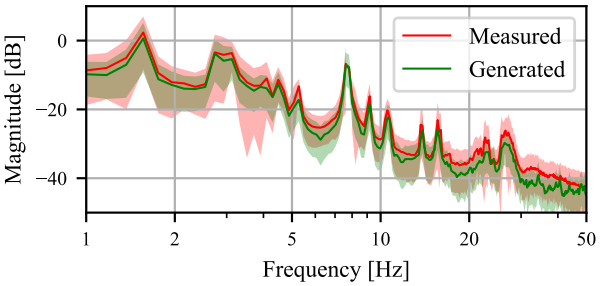


Figure 12: *Average spectra and standard deviations of delay trajectories using real data.*

about 5 kHz, as seen in the sudden increase of nonlinear distortion components in Fig. 10a. The learned magnitude response and nonlinear distortion components for the adversarially trained model are shown in Fig. 10b. Now the match is poor in both the linear response as well as the nonlinear harmonic components.

The $\mathcal{L}_{\text{ESR}}$ and $\mathcal{L}_{\text{STFT}}$ losses over the test set for all of the approaches are listed in Table 2. Similarly as in Sec. 6.2, the two supervised approaches outperform the adversarial approach in terms of both of the considered metrics, with an order of magnitude difference in the time-domain $\mathcal{L}_{\text{ESR}}$ loss. Unlike before, the first supervised approach performs slightly better than the second approach, although the difference is not large. While the two methods for computing the losses for the first supervised approach produce similar evaluation metrics, this time the demodulated computational method brings slight improvements in the $\mathcal{L}_{\text{STFT}}$ loss. This finding suggests that the error produced by the two computational methods also depends on the type of data used.

7.2. Trajectory Generator

Fig. 11 shows delay trajectory samples from the measured data compared to those generated with a 10-step diffusion model. The generated delay trajectory waveforms look realistic at first glance but, in order to provide more insights into the model behavior, we conduct a spectral analysis. Fig. 12 shows the summary statistics (mean and standard deviation) of the long-term spectrum of the measured trajectories compared to that of a batch of 256 generated samples. The data presents some characteristic spectral peaks that the diffusion model is succeeding at replicating. It can also be observed that the average spectral magnitude of the generated trajectories is slightly lower than the target; we attribute this to the over-denoising phenomena that most diffusion models show, as it was observed in [15]. Nevertheless, we believe that these minor dissimilarities will pose no perceptual difference.

Table 2: *Real Data: Nonlinearities and timing effects.*

	$\mathcal{L}_{\text{ESR}}$		$\mathcal{L}_{\text{STFT}}$	
Approach	Demod.	Delayed	Demod.	Delayed
Supervised I	0.066	**0.065**	**1.597**	1.649
Supervised II	–	0.092	–	1.971
Adversarial	–	1.193	–	2.437

7.3. Noise Generator

We assess the diffusion noise generator qualitatively, similar to the delay trajectory generator, since an objective evaluation is not feasible. Fig. 13 displays the long-term spectra of the noise data and compares it with that of the generated noises using the diffusion model with 16 discretization steps. The plot in Fig. 13 has been smoothed using a 1/6th octave band. The majority of the energy in the target data distribution is concentrated in the low-frequency region, with some spectral peaks caused by electrical noise. The diffusion model successfully replicates the spectral distribution, and the over-denoising effect observed in Sec. 7.2 does not occur as a consequence of using a stochastic sampler. Additionally, the noise generator can effectively model some non-stationary local characteristics in the noise data that are not adequately represented in the spectral analysis. We refer the reader to the audio examples on the companion webpage[1].

8. CONCLUSIONS

This work proposed an architecture for modeling the character of magnetic tape recorders, consisting of three components for reproducing the different aspects of the target: 1) a nonlinear block for the joint effects of the magnetic recording process as well as the record and playback amplifiers, 2) a time-varying delay line controlled by a delay trajectory generator for imperfections in the tape transport, and 3) a noise generator for the tape hiss. The different blocks were implemented using separate neural network archi-

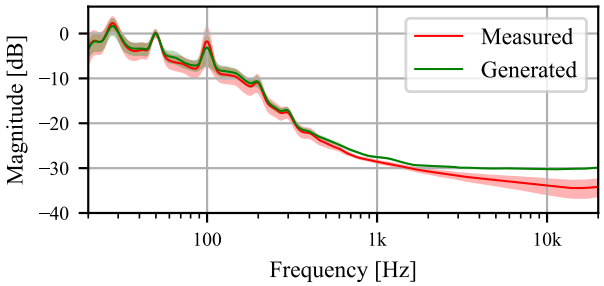


Figure 13: *Average spectra and standard deviations of tape hiss.*

tectures: an RNN for the nonlinear block and separate diffusion models for the delay and noise generators. Three training schemes were considered for the nonlinear block: two supervised and one adversarial.

Our results indicate that the RNN architecture is suitable for learning the characteristic hysteretic nonlinear behavior of the tape magnetization. This was also found recently elsewhere for the related case of audio transformers [4]. Based on objective and qualitative evaluation and informal listening, the two supervised approaches for the nonlinear block together with the generative models for the delay trajectories and tape hiss capture the perceptual character of the tape recorder as a whole. While the proposed supervised training schemes require the target machine to operate in stereo, an aspect of which was circumvented by the adversarial approach, this latter approach did not prove mature yet in learning the nonlinear character of the tape.

9. ACKNOWLEDGMENTS

We acknowledge Aalto Science IT for the computational resources.

10. REFERENCES

[1] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, *DAFX: Digital Audio Effects*, chapter "Virtual analog effects", pp. 473–522, John Wiley & Sons, Ltd, Chichester, UK, Mar. 2011.

[2] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. Int. Conf. Digital Audio Effects (DAFX)*, Birmingham, UK, Sept. 2019, pp. 173–180.

[3] E. Moliner and V. Välimäki, "Realistic gramophone noise synthesis using a diffusion model," in *Proc. Int. Conf. Digital Audio Effects (DAFX)*, Vienna, Austria, Sept. 2022, pp. 240–247.

[4] O. Massi, A. I. Mezza, R. Giampiccolo, and A. Bernardini, "Deep learning-based wave digital modeling of rate-dependent hysteretic nonlinearities for virtual analog applications," *EURASIP J. Audio Speech Music Process.*, vol. 2023, no. 1, Mar. 2023.

[5] H. N. Bertram, *Theory of Magnetic Recording*, Cambridge University Press, Cambridge, New York, 1994.

[6] M. Camras, *Magnetic Recording Handbook*, Springer Netherlands, Dordrecht, Netherlands, 1998.

[7] J. Chowdhury, "Real-time physical modelling for analog tape machines," in *Proc. Int. Conf. Digital Audio Effects (DAFX)*, Birmingham, UK, Sept. 2019, pp. 292–298.

[8] S. Arnardottir, J. S. Abel, and J. O. Smith, "A digital model of the Echoplex tape delay," in *Proc. Audio Eng. Soc. 125th Conv.*, San Francisco, CA, Oct. 2008.

[9] J. Kaloinen, "Neural modeling of the audio tape echo effect," M.S. thesis, Aalto University, Espoo, Finland, Aug. 2022.

[10] E. D. Daniel, "Tape noise in audio recording," *J. Audio Eng. Soc.*, vol. 20, no. 2, pp. 92–99, Mar. 1972.

[11] U. Zölzer, *DAFX: Digital Audio Effects*, John Wiley & Sons Ltd, Chichester, UK, second edition, 2011.

[12] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Proc. ICLR*, Addis Ababa, Ethiopia, Aug. 2020.

[13] A. Wright, V. Välimäki, and L. Juvela, "Adversarial guitar amplifier modelling with unpaired data," in *Proc. IEEE ICASSP*, Rhodes Island, Greece, June 2023.

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. NeurIPS*, Online, Dec. 2020, vol. 33, pp. 6840–6851.

[15] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," in *Proc. NeurIPS*, New Orleans, LA, Oct. 2022, vol. 35, pp. 26565–26577.

[16] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *Proc. ICLR*, Vienna, Austria, May 2021.

[17] S. Hawley, B. Colburn, and S. I. Mimilakis, "Profiling audio compressors with deep neural networks," in *Proc. Audio Eng. Soc. 147th Conv.*, New York, NY, Oct. 2019.

[18] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[19] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Brighton, UK, May 2019, pp. 471–475.

[20] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, *et al.*, "MelGAN," in *Proc. NeurIPS*, Vancouver, Canada, Dec. 2019, vol. 32, pp. 14843–14854.

[21] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Proc. ICLR*, Vienna, Austria, May 2021.

[22] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," in *Proc. NeurIPS*, New Orleans, LA, Dec. 2022, vol. 35, pp. 8633–8646.

[23] M. Holters and U. Zölzer, "Circuit simulation with inductors and transformers based on the Jiles-Atherton model of magnetization," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, Brno, Czech Republic, Sept. 2016, pp. 55–60.

[24] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel WaveGAN," in *Proc. IEEE ICASSP*, Barcelona, Spain, May 2020, pp. 6199–6203.

[25] C. J. Steinmetz and J. D. Reiss, "Auraloss: Audio-focused loss functions in PyTorch," in *Proc. DMRN+15*, London, UK, Dec. 2020.

[26] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," in *Proc. Audio Eng. Soc. 108th Conv.*, Paris, France, Feb. 2000.

PERCEPTUAL EVALUATION AND GENRE-SPECIFIC TRAINING OF DEEP NEURAL NETWORK MODELS OF A HIGH-GAIN GUITAR AMPLIFIER

Will J. Cassidy and Enzo De Sena,

Institute of Sound Recording
University of Surrey
Guildford, United Kingdom
willcassidy7@yahoo.com | e.desena@surrey.ac.uk

ABSTRACT

Modelling of analogue devices via deep neural networks (DNNs) has gained popularity recently, but their performance is usually measured using accuracy measures alone. This paper aims to assess the performance of DNN models of a high-gain vacuum-tube guitar amplifier using additional subjective measures, including preference and realism. Furthermore, the paper explores how the performance changes when genre-specific training data is used. In five listening tests, subjects rated models of a popular high-gain guitar amplifier, the Peavey 6505, in terms of preference, realism and perceptual accuracy. Two DNN models were used: a long short-term memory recurrent neural network (LSTM-RNN) and a WaveNet-based convolutional neural network (CNN). The LSTM-RNN model was shown to be more accurate when trained with genre-specific data, to the extent that it could not be distinguished from the real amplifier in ABX tests. Despite minor perceptual inaccuracies, subjects found all models to be as realistic as the target in MUSHRA-like experiments, and there was no evidence to suggest that the real amplifier was preferred to any of the models in a mix. Finally, it was observed that a low-gain excerpt was more difficult to emulate, and was therefore useful to reveal differences between the models.

1. INTRODUCTION

Analogue vacuum-tube guitar amplifiers are still valued in the audio community, despite being heavy, expensive, and high-maintenance. Historically, several methods have been proposed to emulate vacuum-tube amplifiers [1], including white-box models such as transient modified nodal analysis and wave digital filters [2], and block-oriented grey-box methods such as the Wiener-Hammerstein topology [3]. More recently, advances in deep neural networks (DNNs) have seen promising results compared to traditional approaches [4, 5, 6]. Neural networks are particularly well suited to the black-box modelling of the complex and non-linear internal operations of a guitar amplifier, where training can be performed based on input data (a direct-injected guitar signal) and output data (the distorted, amplified signal) alone. In the literature, DNN models are normally evaluated using objective accuracy measures such as error-to-signal ratio [7, 4] and mean square error based metrics [8, 9, 10]. Subjective accuracy measures have also been seen in the works of [4] and [5], where subjects were asked to rate models in terms of how accurately they approximated the timbre of the reference, and in terms of perceived similarity to the reference, respectively. However, the 'realism' and 'preference' of DNN amplifier models has not been studied to the same extent - is the accuracy of a guitar amplifier model as important as its realism? Also, could a model be preferred to the real amplifier? This paper aims to assess the subjective performance of two popular DNN topologies, namely a long short-term memory recurrent neural network (LSTM-RNN) and a WaveNet-based convolutional neural network (CNN), modelling a popular high-gain vacuum-tube guitar amplifier, the Peavey 6505.

Guitar amplifiers are often specific to certain genres of music. This is especially so for high-gain amplifiers, which are commonly used in heavy rock and metal. Furthermore, certain types of guitar hardware are used more than others, as well as certain playing techniques. Several guitar recordings datasets are publicly available, including the Fraunhofer Institute for Digital Media Technology (IDMT) guitar and bass datasets [11, 12]. These recordings include a range of general techniques, notes and guitar types, and have been used in [10], [8] and [13] to train DNN amplifier models. As Parker *et al.* point out [14], the state-space of an audio system may require certain inputs in order for the target characteristics to be learned effectively, such as for nonlinearities that only occur above a magnitude threshold. On this basis, it is hypothesised in this paper that DNN models of high-gain amplifiers should be trained using data tailored to the target device. In this paper, the IDMT dataset is compared to two genre-specific training files, focused on rock and metal styles, respectively.

The paper is organised as follows. Section 2 describes the target system, i.e. the amplifier and loudspeaker cabinet chain. The DNN models used in this work are then detailed in Section 3, the training of which is outlined in Section 4. The methodology and results of the listening experiments are presented in Section 5, and the results are discussed in Section 6. The main conclusions are summarised in Section 7 with suggestions for further work.

2. TARGET SYSTEM

The target system consists of a guitar amplifier and a loudspeaker cabinet. Only the guitar amplifier was modelled as part of the DNNs, while the loudspeaker cabinet was modelled separately [13], as a linear time-invariant (LTI) system.

The selected target amplifier is the high-gain vacuum tube Peavey 6505, a popular choice in metal recording. High-gain amplifiers usually consist of a preamplifier stage with around 3-7 small vacuum-tubes, commonly 12AX7 dual-triodes, which provide the majority of the non-linear signal distortion [2, 15]. For high-gain amplifiers, a 'drive' parameter applies gain to the input signal before this stage to drive the preamplifier tubes, thus in-

creasing the total harmonic distortion (THD) of the system. The preamplified signal passes through a linear tone stack circuit before power amplifier vacuum-tubes provide signal gain to sufficiently drive the loudspeaker cabinet [2]. These power tubes contribute to the linear tonal characteristics of the amplifier, referred to as the 'British' or 'American' tone depending on their model [15].

The selected target loudspeaker cabinet was a Marshall 1960-AV, consisting of four 12-inch Celestion Vintage 30 loudspeakers which were also used in [16]. The impulse response (IR) of the loudspeaker cabinet was measured using a 30-second long exponential sine sweep (ESS). The sweep was generated at -6 dBFS which was routed to the line output of a Universal Audio Apollo Twin X audio interface. A QSC RMX850 power amplifier was used to apply clean gain to the ESS signal, the output of which was connected to the matched-impedance input of the loudspeaker cabinet. The loudspeaker response was recorded using a Royer R-121, a professional-grade figure-8 ribbon microphone with a 30–15,000 Hz $\pm$3dB response and very high overload characteristics (135 dB SPL). The microphone was positioned at approximately 20mm off-centre from the dust cap. The output SPL of the loudspeaker cabinet was set high enough to provide sufficient SNR, yet not to the extent where significant cone breakup was introduced.

3. DNN MODELS

Two DNN topologies that have been previously used for guitar amplifier modelling are a feedforward variant of WaveNet and an LSTM-based RNN, both of which are compared in [10], [13], and [17]. The implementation used in this paper for the WaveNet-based CNN is the *PedalNetRT* repository, while the one used for the LSTM-RNN is the *Proteus* repository [18]. *PedalNetRT* modifies the original *pedalnet* repository [19], which was a recreation of the WaveNet-based model from the paper by [7]. The modification uses custom causal padding and reorganises conv1d layers to allow trained models to be saved as .json files, which can be loaded using the audio plugin from the *WaveNetVA* repository [20].

The *Proteus* project consists of an audio plugin built using *RTNeural*, a realtime C++ inferencing engine [21]. The plugin can load models trained using the *Automated GuitarAmpModelling* respository [18], forked from Wright's repository [22], which is an implementation of the LSTM-RNN network used by Wright *et al.* [13] in their modelling of the Blackstar HT-1 amplifier and the Big Muff Pi pedal.

Despite both of these models being capable of conditioned training, where the effects of varying a parameter such as drive can be learned, the models in this paper were designed to be a 'snapshot', i.e. a model of the amplifier with fixed parameters, since this was sufficient for the scope of the experimentation.

The LSTM-RNN models used the hyperparameters recommended by [18] for medium to high-gain amplifier emulation. This used an LSTM hidden size of 40 as required by the *Proteus* audio plugin, no pre-emphasis filtering, one recurrent block, and a skip-connection. The WaveNet-based models in this experiment used the default hyperparameters from *PedalNetRT*, i.e. 9 layers, 4 convolution channels, a kernel size of 3 and a batch size of 64. While this is lower than what was suggested by Wright *et al.* [7], these hyperparameters result in a running complexity closer to LSTM-RNN. It is acknowledged that this hyperparameters configuration does not represent the full potential of the WaveNet-based CNN, and therefore the two DNN topologies are not compared directly in this work.

4. TRAINING

This section details the process of producing the direct inject (DI) and amplifier signals for three training sets: a general dataset, an existing genre-specific dataset, and a proposed genre-specific dataset. Both DNNs introduced in Section 3 were trained on each training set, using back-propagation with a loss function based on the error-to-signal ratio (ESR). Google Colab was used to train the LSTM-RNN models, and the WaveNet-based models were trained remotely using the University of Surrey High Performance Cluster, utilising the Python preparation and training files provided in the aforementioned repositories by Bloemer [18].

4.1. Existing Training Datasets

The training file used by Wright *et al.* [13] in their emulation of the Blackstar HT-1 amplifier, accessible from [18] is used here. This training set was constructed using excerpts from the Fraunhofer IDMT databases, forming a 5 minute, 40 second file of half bass and half electric guitar. A range of pickup selections and string gauges were used, and the main playing techniques are described in Table 1. This training is henceforth referred to as the *general* training.

Bloemer [18] recorded a set of genre-specific training samples featuring a wider range of techniques and notes than the *general* file, lasting 3 minutes and 31 seconds. The excerpts in this training are more rock-oriented than those of the IDMT database, and the duration was weighted more towards electric guitar than bass. This training dataset is henceforth referred to as the *rock-specific* training, and serves as a middle ground between the general training and the training made specifically for the Peavey 6505.

4.2. Proposed Metal-specific Training

Rock and metal genres of music share many electric guitar techniques, with some aspects being more exclusive to metal such as pinch-harmonics and low tuning. In this paper, a training file is proposed which was created by recording popular metal guitar excerpts, with a focus on more specific metal techniques highlighted by [23], which were not present in the other datasets. These techniques are outlined in Table 1.

The guitars used for the proposed dataset were the Schecter KM-7 MKIII Artist, the Ibanez RG421 with Bareknuckle Aftermath pickups, and the Dingwall NG-2 5-string. When recording each guitar, the output was connected in series to a true-bypass Peterson tuner pedal, a Radial J48 active DI box and the microphone input of a Universal Audio Apollo Twin X interface. Engaging the -15dB PAD (passive attenuation device) on the DI box was necessary for the active guitars as the input transformer was being overloaded, and so it was engaged for all guitars for consistency. The guitar volume/tone potentiometers were first set to 100% (most transparent), and the preamplifier gain was set such that 10dB of headroom was present when palm-muting heavily. The UAD Diezel Herbert amplifier simulator [24] was used for monitoring purposes.This proposed training dataset lasts a total of 5 minutes and 27 seconds and is henceforth referred to as the *metal-specific* training.

4.3. Training Comparison

Table 1 presents musical aspects of the three training sets, where the genre-specific training files can be seen to have a more ex-

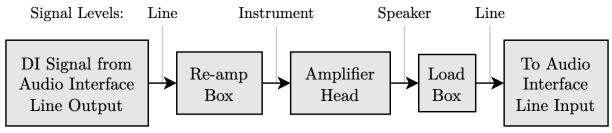


Figure 1: *Signal chain for the amplifier recording process. Signal levels are annotated to highlight the importance of each block.*

Figure 2: *Equipment setup for the recording of the amplifier output.*

tended range of techniques and notes compared to the *general* set. The *rock-specific* training has the widest frequency range when considering the exponential sine sweeps and noise samples at the start of the file.

4.4. Recording the Amplifier Output

The direct output of the Peavey 6505 *LEAD* channel was recorded for each of the three DI training files. The recording chain was set up as per the block diagram in Figure 1, the equipment of which is shown in Figure 2. A Radial X-Amp active re-amp box was used to attenuate the line-level audio interface output to instrument level, with an output impedance of 10kΩ. Typical electric guitar output impedances are in the range of 5-12kΩ [25] - the values of which are expected to be seen by the input of a guitar amplifier. Presenting the correct output impedance is important to ensure the voltage drop across the amplifier is within nominal levels, in order for the amplifier to behave as expected.

The Rivera RockCrusher load box was used to attenuate the high-power output of the amplifier to line level. This is required in replacement of a loudspeaker cabinet, since powering a vacuum-tube amplifier without sufficient load can be damaging [26]. As [13] points out, the type of load may influence the behaviour of the amplifier differently to a loudspeaker cabinet. Therefore, care was taken to select a high-quality reactive load box to act as transparently as possible. The load box output impedance of 560Ω [26] allows for optimal voltage transfer to the line-level input of the Apollo Twin X (rated at 10kΩ [27]). After a preliminary recording, the preamplifier gain was increased to compensate for the voltage loss resulting from headroom provided at earlier stages.

5. SUBJECTIVE LISTENING EXPERIMENTS

Each of the three training sets introduced in Section 4 were used to train the LSTM-RNN and WaveNet-based CNN, resulting in 6 models of the Peavey 6505. Listening tests were conducted to investigate the perceptual preference, realism and accuracy of these models, compared to the real amplifier. The test samples are made

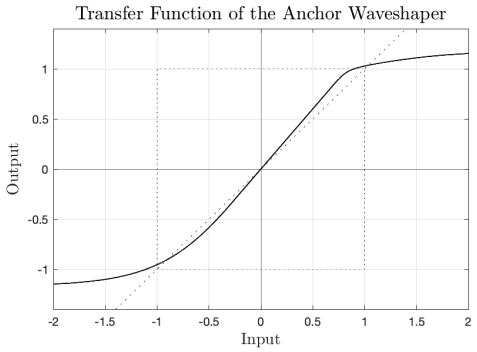


Figure 3: *The transfer function used as a static waveshaper to produce the anchor test samples.*

available on the Institute of Sound Recording's GitHub page[1].

5.1. Test Subjects

A total of 21 participants took part in the first three listening tests. In an anonymous survey, 86% of subjects said they had critically listened to rock or metal music before (i.e. in studio monitoring conditions), 76% had previous experience with vacuum-tube guitar amplifiers, and 81% had experience with hardware or software amplifier simulators.

In the final two listening tests, 16 people took part, all of which had used a vacuum-tube guitar amplifier before, 94% had used an amplifier simulator before and 88% had critically listened to rock or metal music.

All subjects were students of the BSc in Music and Sound Recording course (Tonmeister) at the University of Surrey, all of whom received critical listening training as part of their curriculum.

5.2. Test Excerpts

A range of pickups were used to record 8 guitar excerpts as detailed in Table 2. For each pickup, two excerpts from existing rock and metal songs were chosen with different pitch registers. Playing techniques were also different between excerpts, which included variations of the guitar volume control. Acting as an attenuation device before the amplifier, the volume control can be used to reduce the guitar output level to the 'edge of breakup' (also known as breakup point [9]). These test samples were recorded as 44.1kHz, 16-bit integer linear PCM waveform files, and were each 5-8 seconds in length.

An anchor was produced using a static waveshaper (as in [7]), created using a piecewise transfer function shown in Figure 3. This transfer function is based on the vacuum-tube-like waveshaper designed by [28], and was intended to be distinguishable from the real amplifier due to its simplicity.

The 8 DI guitar excerpts were sent through each of the 6 models, the real amplifier and the anchor waveshaper, resulting in 64 test samples. The models were captured via the *WaveNetVA* and *Proteus* audio plugins in *REAPER*, and the real amplifier output was recorded as part of the process in Section 4.4. The output signals were then convolved with the loudspeaker cabinet IR, and

[1] `https://github.com/IoSR-Surrey/DNNAmplifierDemos`

Table 1: *Information about each of the three training data sets from inspection. The pitch ranges consider the lowest and highest notes played, excluding harmonic techniques.*

Training Data Set	Set 1: General [13]	Set 2: Rock-specific [18]	Set 3: Metal-specific (proposed)
Excerpt Durations	Approx. 10-30s	Approx. 1.5s	Approx. 5-10s
Pitch Range	E1-A#4 (3.5 octaves)	E1-C6 (4.67 octaves)	C1-C6 (5 octaves)
Techniques	Picked bass	Picked bass	Picked bass
	Fingerstyle bass	High-velocity strumming	Pinch harmonics
	Slap bass	Palm-muting	Tapped harmonics
	Strummed dead-notes	High-pitched monophony	Tremolo picking
	Fingerstyle arpeggios	Double-stops	Double-stops
	Monophonic notes	Low-velocity arpeggios	Strummed chords
	Staccato chords	Full-tone bends	Vibrato
	Picked arpeggios	Rapid monophonic picking	Intermod. distortion
	Background tone	Strummed chords	Full-tone bends
		Palm-muted scales	Hammer-ons/pull-offs
		Scales high and low	Fast picking runs
		Natural harmonics	Fast legato
		Vibrato	Volume roll-off
			Heavy palm-muting

equalisation was applied (-9.7dB notch filter at 3.8kHz, Q = 19) to reduce the rate of fatigue of each test subject.

5.3. Experimental Methodology and Statistical Analysis

The experimental methodology involved MUSHRA-style tests [29], which have been previously used in this context [4, 5, 6]. Since this work investigated subjective measures beyond model accuracy, a reference was not used in tests where this would bias the subject's opinion. An ABX test was also used to evaluate model accuracy, which is recommended for the evaluation of smaller differences [30]. The listening tests were conducted inside an acoustically treated room in the Institute of Sound Recording at the University of Surrey. The stimuli were presented to subjects via a *Max/MSP* patch on a 2019 MacBook Pro, monitored over Audio-Technica ATH-M40X headphones. Before the listening tests, each participant was guided through familiarisation, training and blind grading phases. Subjects were made to familiarise themselves with all the unlabelled stimuli and the GUI before conducting the test. During this process, listeners were encouraged to set the monitoring volume to a comfortable level.

The statistical analysis of all MUSHRA-style tests was based on (non-parametric) Friedman tests [31] and post-hoc Wilcoxon pairwise signed-ranks tests. Considering that the paper aims to assess the effect of training within each model, and how well each model performed against the real amplifier, only the following pairwise tests were run: (a) differences between the 3 LSTM-RNN models, (b) differences between the 3 WaveNet-based CNN models, and (c) differences between all models and the real amplifier, for a total of 12 comparisons. The Bonferroni correction was applied to adjust for multiple comparisons.

5.4. Experiment 1 - Preference

The aim of the first experiment was to gauge which amplifier the subjects preferred, be it real or artificial. The test samples were presented in a mix of drum kit and bass guitar to simulate the listening conditions the consumer would experience when judging the guitar recording of a song. The test prompt was worded as: "Rate your preference of the electric guitar in samples A-G".

5.4.1. Methodology

The test used a MUSHRA-style methodology, but without a labelled reference of the real amplifier, so as to account for the possibility of the real amplifier not being the preferred stimulus. Also, an anchor was omitted to reduce the compression of results since the samples appeared to sound very similar. Subjects were asked to rate 7 test samples (i.e. the 6 models and the real amplifier) side by side for 4 different excerpts. Each excerpt was presented on a different page, and each page was repeated once. The scale ranged from -50 to 50 with -10 to 10 labelled as "Indifferent", -50 labelled as "This sounds worse than the others" and 50 labelled as "I prefer this to the others".

5.4.2. Results

The results of the preference test are presented in Figure 4. The mean scores and 95% confidence intervals for each model and the real amplifier were each within the -10 to 10 category, labelled "Indifferent". A Friedman test revealed that there was a statistically significant difference between some of the models ($\chi^2(6, N{=}168) = 14.409$, $p = 0.025$). However, post hoc Wilcoxon signed-ranks tests showed that there was no statistically significant difference between the three LSTM-RNN models (i.e. the three different training sets) or between the three WaveNet-based models. Similarly, there was no statistically significant difference between each of the 6 models and the real amplifier.

5.5. Experiment 2 - Realism

This experiment aimed to investigate what subjects believed sounded like a 'real' amplifier given their previous experience of vacuum-tube guitar amplifiers, without a reference. Two MUSHRA-style listening tests were conducted. The first test asked subjects to rate how 'real' the samples sounded, and the second test asked subjects to compare the samples to their previous experience of what a real

Table 2: *Information about each of the 8 excerpts used to form the listening test samples. 'EOB' refers to 'edge of breakup': the lowest of the gain levels. Under 'Song Based On,' the artist is not included for space reasons (full details are provided in the Github repository 1).*

Excerpt	Pickup Model	Pickup Passivity	Song Based On	Tuning	Pitch Register	THD
A	EMG Humbucker	Active	B.Y.O.B. 0:41-0:46	Drob Db	Low	Med/High
B	EMG Humbucker	Active	Tears Don't Fall 0:00-0:06	Drob Db	Mid	Med
C	Fishman Fluence Humbucker	Active	Death Inside 2:21-2:27	Drop Bb	Low	High
D	Fishman Fluence Humbucker	Active	Catalyst 1:31-1:39	Drob Db	Mid	High
E	Fishman Fluence Split-coil	Active	Cry of Achilles 0:32-0:39	Eb Standard	Mid	High
F	Fishman Fluence Split-coil	Active	My Curse 0:00-0:08	Drop C	Mid/High	EOB
G	Bareknuckle Aftermath Humbucker	Passive	My Curse 1:01-1:09	Drop C	Low	High
H	Bareknuckle Aftermath Humbucker	Passive	Buried Alive 4:14-4:20	Standard	High	High

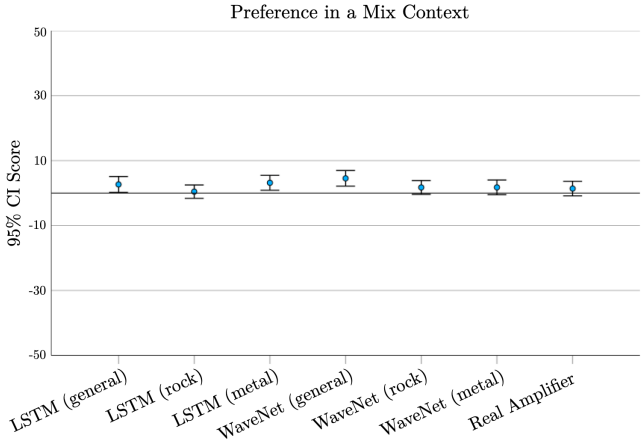


Figure 4: *Means and 95% confidence intervals for the listening test in experiment 1. The results are averaged across all excerpts (C, D, E and G).*

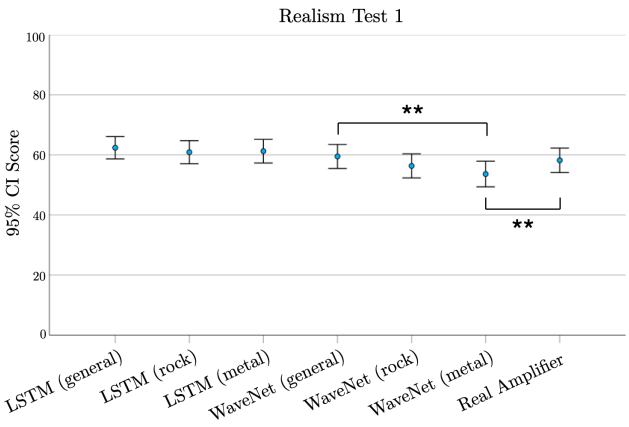


Figure 5: *Means and 95% confidence intervals for the first listening test in experiment 2. Asterisks and bars indicate a significant difference (*: p < .05, **: p < .01 , ***: p < .001 at post-hoc test, Bonferroni corrected).*

vacuum-tube amplifier sounds like. Only subjects that had used a vacuum-tube guitar amplifier before were permitted to participate in the second experiment. A total of 21 subjects participated in the first test, while 16 participated in the second test. These samples were not presented with accompaniment unlike Section 5.4, since realism should not depend on other instruments - using a mix may cause unnecessary masking effects.

5.5.1. Methodology

The first test of this experiment asked subjects to "rate samples regarding how 'real' they sound" on a scale of 0-100 labelled from "This sounds artificial" to "This sounds like a real amplifier". The test consisted of 8 pages, each of which involved comparing the 6 models and the real amplifier using one guitar excerpt as an input. The excerpt was randomly changed for each page, using excerpts A, C, E and G from Table 2 and repeating them once.

The second test asked listeners to "rate each sample based on how similar it sounds to a real vacuum-tube guitar amplifier". On each page of the second test, 8 unlabelled samples were compared (the 6 models, the real amplifier and the anchor). The rating scale was also 0-100, labelled from "Not similar" to "Sounds the same". This was completed for 6 excerpts (A, B, C, F, G and H) and repeated once, resulting in a total of 12 pages.

5.5.2. Results

The results of the first realism test are shown in Figure 5. A Friedman test showed a significant difference between some of the models ($\chi^2(6, N{=}168) = 35.907$, $p < 0.001$), so post hoc Wilcoxon signed-ranks tests were performed. There was no statistically significant difference between the LSTM-RNN models. For the WaveNet-based models, on the other hand, the *general* model was significantly more realistic than the *metal-specific* model (p = 0.0097, adjusted). When comparing each model with the reference, the real amplifier was only significantly more realistic than the *metal-specific* WaveNet-based model (p = 0.0034, adjusted).

Figure 6 shows the mean realism scores of the second test, where a Friedman test also returned statistically significant differences ($\chi^2(6, N{=}192) = 16.412$, $p < 0.012$). Post hoc Wilcoxon signed-ranks tests showed no statistically significant differences between the LSTM-RNN models. Within the WaveNet-based models, the *general* model was significantly more realistic than the *metal-specific* one (p = 0.0269, adjusted), as was seen in the first test. For all of the models there was no statistically difference from the real amplifier.

To examine the effects of the model and excerpt on the mean realism scores of the second test, Friedman tests were run for each of the 6 excerpts used. For the results of excerpt F, a Friedman test

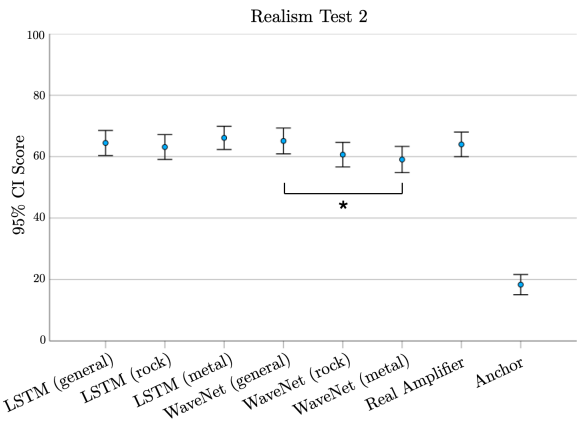


Figure 6: *The mean scores of each model with 95% confidence intervals for the second listening test in experiment 2.*

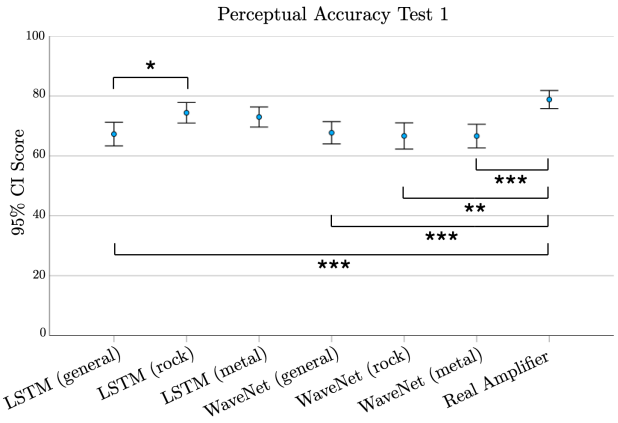


Figure 7: *Means and 95% confidence intervals for the first listening test from experiment 3: similarity to the reference. The results are averaged across all excerpts (B, D, F and H).*

returned a statistically significant difference in realism between the models ($\chi^2(6, N{=}32) = 16.162$, p = 0.013). Performing post hoc Wilcoxon signed-ranks tests showed that the *rock-specific* LSTM-RNN model was rated as significantly *more* realistic than the real amplifier (p = 0.0383, adjusted), and the *general* WaveNet-based model was also significantly *more* realistic than the real amplifier (p = 0.0234, adjusted). For higher-gain excerpts B, C, G and H, Friedman tests revealed there was no significant differences between models. Despite the Friedman test for excerpt A showing significance ($\chi^2(6, N{=}32) = 14.649$, p = 0.023), the post hoc Wilcoxon signed-ranks tests revealed no significant comparisons when considering the LSTM-RNN models alone, the WaveNet-based models alone and the 6 models versus the real amplifier.

5.6. Experiment 3 - Accuracy

The final experiment sought to evaluate the models in terms of perceptual accuracy compared to the (labelled) real amplifier.

5.6.1. Methodology

This experiment first used a MUSHRA-style test which asked subjects to rate the similarity of 7 test samples (the 6 models and a hidden reference) to a labelled reference of the real amplifier on a scale of 0 to 100. Subjects were not asked to rate one of the samples at 100. The excerpts used in this test were B, D, F and H, each on a different page, repeated once, resulting in a total of 8 pages.

An ABX test was also conducted which gave subjects a labelled reference of the real amplifier and two test samples: a hidden reference and one of the 6 models. Subjects were tasked with identifying which of the two samples was the hidden reference for excerpts A, B, C, F, G and H, randomised and repeated once, resulting in 72 trials.

5.6.2. Results

Figure 7 presents the results of the MUSHRA-style test. The hidden reference reached a mean score of just 79% (this motivated running the ABX test later). A Friedman test revealed that there was a statistically significant difference between some of the models ($\chi^2(6, N{=}168) = 49.019$, $p < 0.001$). Post hoc Wilcoxon signed-ranks tests showed that the *rock-specific* LSTM-RNN model was significantly more accurate than the *general* LSTM-RNN (p = 0.0109, adjusted). No significant differences were observed within the WaveNet-based models. The real amplifier was rated as significantly more accurate than all three WaveNet-based models as well as the *general* LSTM-RNN ($p \leq 0.001$ in each case, adjusted).

To investigate differences between excerpts, Friedman tests were run for each excerpt of the MUSHRA-style test which found that the lowest-gain excerpt F had a significant interaction ($\chi^2(6, N{=}42) = 65.812$, $p < 0.001$), while the high-gain and high-pitched excerpt H did not ($\chi^2(6, N{=}42) = 3.848$, p = 0.697). For excerpt F, post hoc Wilcoxon signed-ranks tests were run, which found that both genre-specific LSTM-RNN models were significantly more accurate than the *general* LSTM-RNN ($p < 0.001$ in both cases, adjusted). Significant differences were also found between the real amplifier versus the *general* LSTM-RNN and each genre-specific WaveNet-based model ($p < 0.001$ in each case, adjusted).

Figure 8 shows the ABX results, where the 95% and 99% critical levels are indicated (using the cumulative binomial distribution). At the 95% confidence level, it can be seen that all models could be distinguished from the reference. Using 99% confidence, however, the *rock-specific* LSTM-RNN does not exceed the critical level which suggests it was very similar to the reference. According to the cumulative binomial distribution at 95% confidence, there was no statistically significant difference between the *rock-specific* and *metal-specific* LSTM-RNN results. The *general* LSTM-RNN was identified significantly more often than both genre-specific LSTM-RNN models at $\alpha = 0.05$.

For the lowest-gain excerpts, B and F, the *rock-specific* RNN was the only model not to have been rated as significantly different to the reference at the 95% confidence level. For the high-gain and high-pitched excerpt H, however, none of the models could be distinguished from the reference.

6. DISCUSSION

In terms of preference, the mean scores for all 6 models and the real amplifier were within the -10 to 10 band (labelled “Indifferent"). There were no significant differences between the models

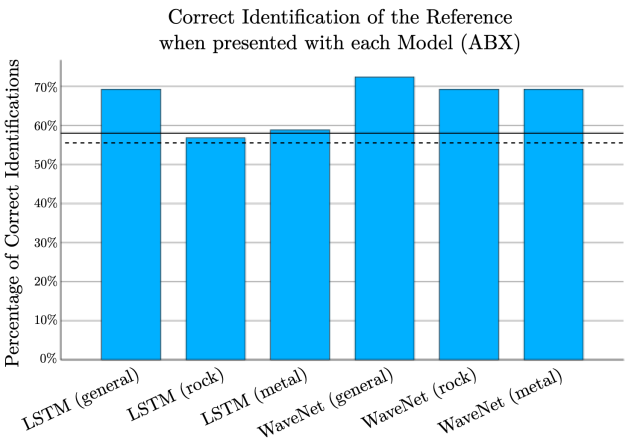


Figure 8: *The percentage of correct identifications of the reference when compared to each model in the ABX test. In this plot, lower values mean better performance. The dashed and solid horizontal lines represent the 95% and 99% critical levels, respectively.*

within either of the DNN topologies, nor were the 6 models rated as significantly different to the real amplifier. This suggests that, despite potential audible differences, the real amplifier was not preferred over any of the DNN models when accompanied with bass guitar and drums, for high-gain excerpts C, D, E and G from Table 2. Therefore, these models seem to be viable as replacements of a real guitar amplifier in a mix.

In one of the realism tests, the *metal-specific* WaveNet-based model was significantly less realistic than the real amplifier. There were no other significant differences in realism between the models and the real amplifier when considering all excerpts cumulatively. This suggests the models were generally realistic-sounding. The mean realism scores for the real amplifier were low in both tests (58% and 64%) - it is possible that this was due to subjects finding the judgement of realism difficult. The training sets did not drastically affect the realism of the models, which is most likely due to the fact that the models are already perceived as very realistic.

In terms of perceptual accuracy, the hidden reference (real amplifier) had a surprisingly low mean score of 79%, despite subjects being asked to rate the degree of similarity to the labelled reference. It is possible that this was due to not forcing subjects to rate at least one sample to 100% and/or to the reference being so close to the other samples and thus difficult to identify. The real amplifier was rated significantly higher only in comparison to the three WaveNet-based models and the *general* LSTM-RNN model in the MUSHRA-style test. This suggests that the two genre-specific LSTM-RNN models were perceived with similar accuracy to the real amplifier, which is supported by the ABX results. The *rock-specific* LSTM-RNN was not significantly distinguished from the reference in the ABX test (at $\alpha = 0.01$), where it was correctly identified only 57% of the time, suggesting it was very similar to the real amplifier. There was no significant difference between the results of the two genre-specific LSTM-RNN models, which indicates that the *metal-specific* LSTM-RNN was also perceptually close to the real amplifier.

It was found that excerpts closer to the 'edge of breakup' revealed more differences between the models. For the high-gain and high-pitched excerpt H, all models were unable to be identified from the real amplifier in the ABX test, and the MUSHRA-style accuracy test showed no significant difference between any of the models. This is supported by the realism results, where none of the models had significantly lower mean scores than the real amplifier for this excerpt, suggesting that they were all sufficiently realistic. For the lowest-gain excerpt F, however, the *rock-specific* LSTM-RNN was the only model indistinguishable from the reference, and it was rated as significantly *more* realistic than the real amplifier. The MUSHRA-style accuracy test for this excerpt revealed that both genre-specific LSTM-RNN models were more accurate than the *general* LSTM-RNN, and that the real amplifier was more accurate than three other models. The *metal-specific* WaveNet-based model was seen to be significantly less realistic than the *general* WaveNet-based model in both realism tests.

7. CONCLUSIONS AND FURTHER WORK

This paper explores the use of two popular DNN topologies for the modelling of the Peavey 6505 amplifier. Perceptual experiments were run to evaluate models trained using a general dataset versus genre-specific datasets, rated in terms of preference, realism and accuracy.

The real amplifier was not preferred to any of the DNN models when presented in a mix. The models also successfully emulated the target amplifier in terms of realism, and one of the models was even rated as *more* realistic than the real amplifier itself. Also considering that the subjects were trained listeners, these results suggest that the models can already replace the real amplifier in most music production workflows.

The genre-specific training resulted in an improvement of the performance of the LSTM-RNN topology both in terms of accuracy, and, to a lesser extent, realism. No significant difference was observed between the three training datasets for the WaveNet-based models. It is possible that this was due to WaveNet-based models being more sensitive to the choice of hyperparameters, which in this experiment were fixed for all training sets.

Results also showed that some excerpts were better than others in highlighting differences between models. More specifically, high-gain and high-pitched excerpts were perceived with the same realism and accuracy as the real amplifier for all models, while a lower-gain excerpt on the 'edge of breakup' was identified as different from the real amplifier for 5 out of 6 models. Furthermore, significant differences in accuracy between the training of the models were highlighted for this excerpt.

Future work will involve investigating whether the results generalise to different guitar amplifiers and different settings, as well as investigating the sensitivity of the individual models to hyperparameter optimisation and pruning. This may determine whether the observed effects of genre-specific training translate to optimised models, especially for a WaveNet-based CNN.

8. ACKNOWLEDGMENTS

Many thanks to all of the subjects who participated in the listening tests.

9. REFERENCES

[1] M. M. Ramírez, E. Benetos, and J. D. Reiss, "Deep learning for black-box modeling of audio effects," *Applied Sciences*, vol. 10(2), p. 638, 2020.

[2] J. Pakarinen and D. T. Yeh, "A Review of Digital Techniques for Modeling Vacuum-Tube Guitar Amplifiers," *Computer Music Journal*, vol. 33(2), pp. 85–100, 2009.

[3] F. Eichas and U. Zölzer, "Gray-Box Modeling of Guitar Amplifiers," *Journal of the Audio Engineering Society*, vol. 66(12), pp. 1006–1015, 2018.

[4] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep Learning for Tube Amplifier Emulation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-19)*, Aalto University, Espoo, Finland, 12-17 May, 2019, pp. 471–475.

[5] A. Wright, V. Välimäki, and L. Juvela, "Adversarial Guitar Amplifier Modelling with Unpaired Data," in *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece, 4-10 June, 2023, pp. 1–5.

[6] C. J. Steinmetz and J. D. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *Audio Engineering Society Convention 151*, London, UK, 2 May, 2022, pp. 1–9.

[7] A. Wright, E. P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Applied Sciences (Switzerland)*, vol. 10(3), pp. 1–18, 2020.

[8] K. Yoshimoto, H. Kuroda, D. Kitahara, and A. Hirabayashi, "Deep Neural Network Modeling of Distortion Stomp Box Using Spectral Features," in *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Auckland, New Zealand, 7-10 December, 2020, pp. 339–345.

[9] P. Bognár, "Audio Effect Modeling with Deep Learning Methods," Ph.D. dissertation, Vienna University of Technology, 2022.

[10] M. A. M. Ramírez, "Deep Learning for Audio Effects Modelling," Ph.D. dissertation, Queen Mary University of London, 2020.

[11] Fraunhofer IDMT, *IDMT-SMT-Guitar*. Available at: https://www.idmt.fraunhofer.de/en/publications/datasets/guitar.html (Accessed: 27 February 2023), 2014.

[12] ——, *IDMT-SMT-Bass-Single-Track*. Available at: https://www.idmt.fraunhofer.de/en/publications/datasets/bass_lines.html (Accessed: 27 February 2023), 2014.

[13] A. Wright, E. P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19)*, Birmingham, UK, 2-6 September, 2019, pp. 1–8.

[14] J. D. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-19)*, Birmingham, UK, 2-6 September, 2019, pp. 165–172.

[15] E. Barbour, "The cool sound of tubes [vacuum tube musical applications]," *IEEE Spectrum*, vol. 35(8), pp. 24–35, 1998.

[16] D. Yeh, B. Bank, and M. Karjalainen, "Nonlinear modeling of a guitar loudspeaker cabinet," in *11th International Conference on Digital Audio Effects*, Espoo, Finland, 1-4 September, 2008, pp. 300–304.

[17] T. Vanhatalo, P. Legrand, M. Desainte-Catherine, P. Hanna, A. Brusco, G. Pille, and Y. Bayle, "A Review of Neural Network-Based Emulation of Guitar Amplifiers," *Applied Sciences (Switzerland)*, vol. 12(12), pp. 1–26, 2022.

[18] K. Bloemer, *GuitarML*. Available at: https://github.com/GuitarML (Accessed: 12 January 2023), 2022.

[19] T. Koker, *pedalnet*. Available at: https://github.com/teddykoker/pedalnet (Accessed: 12 January 2023), 2020.

[20] E.-P. Damskägg, *WaveNetVA*. Available at: https://github.com/damskaggep/WaveNetVA (Accessed: 20 March 2023), 2020.

[21] J. Chowdhury, "RTNeural: Fast Neural Inferencing for Real-Time Systems," *arXiv preprint*, 2021.

[22] A. Wright, *Automated-GuitarAmpModelling*. Available at: https://github.com/Alec-Wright/Automated-GuitarAmpModelling (Accessed: 12 January 2023), 2021.

[23] J. P. Herbst, "Shredding, tapping and sweeping: Effects of guitar distortion on playability and expressiveness in rock and metal solos," *Metal Music Studies*, vol. 3(2), pp. 231–250, 2017.

[24] Universal Audio, Inc., *Diezel Herbert Amplifier*. Available at: https://www.uaudio.com/uad-plugins/guitar-bass/diezel-herbert-amplifier.html (Accessed: 05 April 2023), 2023.

[25] T. D. Sunnerberg, "Analog Musical Distortion Circuits for Electric Guitars," Master's thesis, Rochester Institute of Technology, 2019.

[26] Rivera Research and Development Co., *RockCrusher™ Owner's Manual*. Available at: https://www.manualslib.com/manual/1930406/Rivera-Rockcrusher.html?page=6#manual (Accessed: 25 February 2023), 2010.

[27] Universal Audio, Inc., *Apollo Twin X Hardware Manual*. Available at: https://media.uaudio.com/support/manuals/hardware/Apollo%20Twin%20X%20Hardware%20Manual.pdf (Accessed: 22 February 2023), 2021.

[28] M. Doidic, M. Mecca, M. Ryle, C. Senffner *et al.*, "Tube Modeling Programmable Digital Guitar Amplification System," US Patent 5789689, August 1998.

[29] ITU-R, "Method for the subjective assessment of intermediate quality level of audio systems," *Recommendation ITU-R BS.1534-3*, 2015.

[30] ——, "Methods for the subjective assessment of small impairments in audio systems," *Recommendation ITU-R BS.1116-3*, 2015.

[31] C. Mendonça and S. Delikaris-Manias, "Statistical tests with MUSHRA data," in *Audio Engineering Society Convention 144*, Milan, Italy, May 23-26 2018, pp. 859–868.

A DIFFERENTIABLE DIGITAL MOOG FILTER FOR MACHINE LEARNING APPLICATIONS

Etienne Gerat , Purbaditya Bhattacharya and Udo Zölzer

Department of Signal Processing and Communication
Helmut Schmidt University
Hamburg, Germany
e.gerat@hsu-hh.de | bhattacp@hsu-hh.de

ABSTRACT

In this project, a digital ladder filter has been investigated and expanded. This structure is a simplified digital analog model of the well known analog Moog ladder filter. The goal of this paper is to derive the differentiation expressions of this filter with respect to its control parameters in order to integrate it in machine learning systems. The derivation of the backpropagation method is described in this work, it can be generalized to a Moog filter or a similar filter having any number of stages. Subsequently, the example of an adaptive Moog filter is provided. Finally, a machine learning application example is shown where the filter is integrated in a deep learning framework.

1. INTRODUCTION

The Moog ladder filter is a well known analog filter present in numerous synthesizers. It has been introduced in 1965 by Robert Moog [1]. Since then, it has been considered as a central piece of some subtractive synthesizers that gives a very recognizable character to the sound and offers intuitive control parameters. Nowadays, many of the iconic analog synthesizers are digitally modeled to be included in digital hardware or software synthesizers. Several digital models of the Moog filter are already studied using different approaches [2, 3].

In recent years, *machine learning* (ML) has been actively applied to the field of audio signal processing. And it has been stated that classic ML and *deep learning* (DL) structures are not always well adapted to solve audio related problems. Integrating audio systems directly in a DL architecture has been proven to be successful and to achieve good results with smaller architectures [4]. Differentiable function blocks are required to allow backpropagation and thus the integration into DL systems. This paper shows the differentiation process of the chosen Moog filter structure with respect to (w.r.t.) its control parameters, as it can be applied to other *Infinite Impulse Response* (IIR) filters and digital signal processing algorithms [5, 6]. As a proof of concept for the backpropagation capabilities, an adaptive version of the filter has been programmed.

This paper is part of a research project, where a subtractive synthesizer that includes a Moog filter would be differentiated to apply timbre matching using ML. This has already been studied using non gradient-based methods [7, 8] and genetic algorithm [9]. And more recently using Variational Auto-Encoders and Normalizing Flows[10]. Hence, the current work initially investigates the capability of learning the control parameters of a simple Moog filter with the help of ML algorithms.

2. DIGITAL MOOG FILTER

In this paper, a digital Moog filter structure presented by Stilson and Smith [11, 3] is used which aims to simulate the analog Moog ladder filter proposed by Robert Moog in 1965 [1]. In the next sections the Moog filter refers to the Stilson and Smith structure.

2.1. Parameters

The filter has two control parameters, the cutoff frequency f_c and the resonance factor K. The cutoff frequency is present at different places in the filter structure. Stilson and Smith [3] have developed a useful compromise first-order filter that has mostly independent control of the resonance value with the cutoff frequency. The filter coefficients h_0, h_1 and h_2 are parameterized to set the filter behavior close to the expected cutoff frequency. They are defined as

$$h_0 = \frac{\omega_c}{1.3} \qquad h_1 = \frac{0.3\,\omega_c}{1.3} \qquad h_2 = 1 - \omega_c, \tag{1}$$

where ω_c is the angular cutoff frequency related to f_c and the sampling frequency f_s by

$$\omega_c = \frac{2\pi f_c}{L f_s} \qquad f_c = \frac{\omega_c L f_s}{2\pi}. \tag{2}$$

Here, L denotes the oversampling factor and is set to 2. The oversampling helps to achieve stability for high values of ω_c and K.

The resonance parameter K is located in the feedback loop, as visible in Fig. 1. It controls the prominence of the resonance overshoot. It ranges from 0 to 1. Values above 1 lead to self-oscillation and instability.

2.2. Structure

The filter is composed of four cascaded first-order low-pass filters with a feedback loop over the whole cascade. A non-linearity expressed as an hyperbolic tangent is present in the loop to provide a ceiling of the feedback signal. A unit delay is present in the feedback loop to ensure the feasibility of the filter as shown in Fig. 1. Figure 2 illustrates the detail of a stage of the Moog ladder filter.

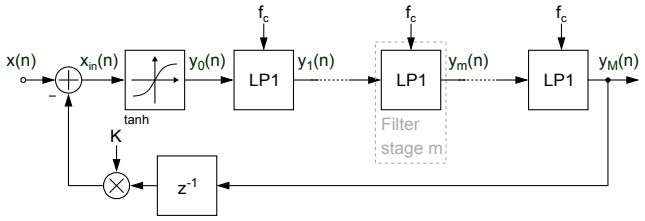


Figure 1: *Block diagram of a M[th] order Moog filter structure.*

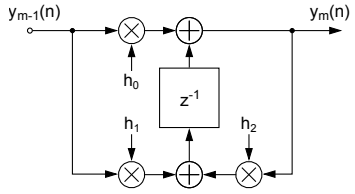


Figure 2: *Block diagram of the m[th] filter stage.*

The following difference equations describe the signals visible in Fig. 1:

$$x_{\text{in}}(n) = x(n) - Ky_4(n-1), \tag{3}$$

$$y_0(n) = \tanh\left(x_{\text{in}}(n)\right), \tag{4}$$

$$y_m(n) = h_0 y_{m-1}(n) + h_1 y_{m-1}(n-1) + h_2 y_m(n-1), \tag{5}$$

where $x(n)$ denotes the input signal and $y_m(n)|_{m=1\cdots4}$ the outputs of the filter stages from 1 to 4.

3. BACKPROPAGATION

The backpropagation of the error calculated by a loss function through the system allows the adaption of the control parameters to match a target sound. For this purpose, the partial derivatives of the filter output w.r.t. its control parameters must be calculated.

3.1. Partial Derivatives

In this section, the expressions of the partial derivatives w.r.t. the control parameters ω_c and K are derived . It is decided for simplicity to calculate the partial derivative against the angular frequency ω_c defined in Eq. (2).

At first, the derivatives of the filter coefficients h_0, h_1 and h_2 w.r.t. ω_c need to be calculated as follows:

$$\frac{\partial h_0}{\partial \omega_c} = \frac{1}{1.3}, \qquad \frac{\partial h_1}{\partial \omega_c} = \frac{0.3}{1.3}, \qquad \frac{\partial h_2}{\partial \omega_c} = -1. \tag{6}$$

Based on Eq. (5), the partial derivative of an output of a filter stage w.r.t. ω_c

$$\begin{aligned}\frac{\partial y_m(n)}{\partial \omega_c} =& \frac{\partial h_0}{\partial \omega_c} y_{m-1}(n) + \frac{\partial h_1}{\partial \omega_c} y_{m-1}(n-1) + \\ & \frac{\partial h_2}{\partial \omega_c} y_m(n-1) + h_0 \frac{\partial y_{m-1}(n)}{\partial \omega_c} + \\ & h_1 \frac{\partial y_{m-1}(n-1)}{\partial \omega_c} + h_2 \frac{\partial y_m(n-1)}{\partial \omega_c},\end{aligned} \tag{7}$$

where m denotes the index of the filter stages from 1 to 4 and the expression $\frac{\partial y_m(n)}{\partial \omega_c}\big|_{m=0}$ is given by

$$\frac{\partial y_0(n)}{\partial \omega_c} = \left[1 - \tanh(x_{in}(n))^2\right] \frac{\partial x_{in}(n)}{\partial \omega_c}, \tag{8}$$

where

$$\frac{\partial x_{in}(n)}{\partial \omega_c} = -K \frac{\partial y_4(n-1)}{\partial \omega_c}. \tag{9}$$

For the initialization, as $n = 1$, the partial derivatives are initialized as follows:

$$\begin{aligned}\left.\frac{\partial y_m(n)}{\partial \omega_c}\right|_{n=1} =& \frac{\partial h_0}{\partial \omega_c}\, y_{m-1}(1) + \left.h_0\, \frac{\partial y_{m-1}(n)}{\partial \omega_c}\right|_{n=1} \\ =& \frac{1}{3} \sum_{k=0}^{m-1} h_0^k y_{m-1-k}(1).\end{aligned} \tag{10}$$

In the next step, the partial derivatives w.r.t. K are calculated and given by

$$\begin{aligned}\frac{\partial y_m(n)}{\partial K} =& h_0 \frac{\partial y_{m-1}(n)}{\partial K} + h_1 \frac{\partial y_{m-1}(n-1)}{\partial K} + \\ & h_2 \frac{\partial y_m(n-1)}{\partial K},\end{aligned} \tag{11}$$

where m denotes the index of the filter stages from 1 to 4 and the expression $\left.\frac{\partial y_m(n)}{\partial K}\right|_{m=0}$ is given by

$$\begin{aligned}\frac{\partial y_0(n)}{\partial K} &= \frac{\partial}{\partial K}\left[\tanh\left(x_{in}(n)\right)\right] \\ &= \left[1 - y_0(n)^2\right] \left[-y_4(n) - K \frac{\partial y_4(n-1)}{\partial K}\right],\end{aligned} \tag{12}$$

where

$$\begin{aligned}\frac{\partial x_{in}(n)}{\partial K} &= \frac{\partial}{\partial K}\left[x(n) - K y_4(n-1)\right] \\ &= -K \frac{\partial y_4(n-1)}{\partial K} - y_4(n-1).\end{aligned} \tag{13}$$

As $n = 1$, the partial derivatives of the individual filter stages m ranging from 1 to 4 are initialized as follow:

$$\left.\frac{\partial y_m(n)}{\partial K}\right|_{n=1} = h_0\, \left.\frac{\partial y_{m-1}(n)}{\partial K}\right|_{n=1}, \tag{14}$$

and the expression $\left.\frac{\partial y_m(n)}{\partial K}\right|_{m=0,n=1}$ is given by

$$\left.\frac{\partial y_0(n)}{\partial K}\right|_{n=1} = -y_M(1)\left[1 - y_0(1)^2\right] \tag{15}$$

for a filter of order M.

It is noteworthy to mention that the expressions of partial derivatives are required to manually construct the backward propagation of the gradients in an environment where automatic derivatives of the forward functions are not calculated, for e.g. MatConvNet [12]. ML environments like Keras[13] and PyTorch [14] provide automatic differentiation of modules constructed by the functions available in their respective packages. However, those environments also offer the possibility to create or alter the backward propagation function, if the automatic backpropagation do not show a stable convergent behavior.

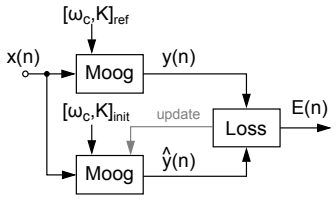


Figure 3: *Simple schematic of a Moog filter adaptation process.*

3.2. Adaptive Moog Filter

In this section, the goal is to verify the functionality of the backpropagation method with the help of an adaptive Moog filter and determine if any gradient tweaking or conditioning is necessary for a simple parameter learning problem . The control parameters of the Moog filter are adapted via backpropagation and a simple parameter update algorithm based on the derivations shown in the previous section is performed.

To adapt a Moog filter, a set of ground truth control parameters $[\omega_c, K]_{\text{ref}}$ are defined initially as shown in Fig. 3. Using a random mixture of various basic signals as the input signal $x(n)$ and the ground truth parameters, the output of Moog filter $y(n)$ is generated and used as the ground truth signal. It is noteworthy to mention that the control signal is oversampled by a factor of $L = 2$ and the corresponding parameters are adjusted. The Moog filter parameters are then initialized with $[\omega_c, K]_{\text{init}}$ and the estimated output signal $\hat{y}(n)$ is compared to the ground truth. The corresponding loss function is given by

$$E = \frac{1}{2}\sum_{n=1}^{N}(|\hat{y}(n)| - |y(n)|)^2, \tag{16}$$

where E denotes the error and $|\cdot|$ denotes the absolute value. The derivative of the error w.r.t. the estimated signal is given by

$$\frac{\partial E}{\partial \hat{y}(n)} = (|\hat{y}(n)| - |y(n)|) \cdot \text{sign}(\hat{y}(n)), \tag{17}$$

where the function $\text{sign}(\hat{y}(n))$ denotes the sign of the estimated signal sample.

The required gradients $\frac{\partial E}{\partial \omega_c}$ and $\frac{\partial E}{\partial K}$ w.r.t. the cutoff frequency f_c and feedback coefficient K are calculated with the help of the chain rule of derivatives and can be given by

$$\frac{\partial E}{\partial \omega_c} = \sum_{n=1}^{N}\frac{\partial E}{\partial \hat{y}(n)}\frac{\partial \hat{y}(n)}{\partial \omega_c}, \tag{18}$$

$$\frac{\partial E}{\partial K} = \sum_{n=1}^{N}\frac{\partial E}{\partial \hat{y}(n)}\frac{\partial \hat{y}(n)}{\partial K}. \tag{19}$$

It is noteworthy to mention that the gradient $\frac{\partial E}{\partial \omega_c}$ is heavily weighted during parameter update such that the initial samples are given more importance. The altered expression can be given by

$$\frac{\partial E}{\partial \omega_c} = \sum_{n=1}^{N}\frac{\partial E}{\partial \hat{y}(n)}\frac{\partial \hat{y}(n)}{\partial \omega_c}\frac{1}{n^k}, \tag{20}$$

where n denotes the sample index and k denotes a positive integer. This change avoids a possible gradient explosion and ensures a stable parameter update and smooth convergence. The expressions in Eq. (20) and Eq. (19) can be derived further with the help of Eq. (17), Eq. (7), and Eq. (11). Finally, gradient clipping is also performed on both the expressions from Eq. (20) and Eq. (19) in order avoid any large gradient jumps and ensure a stable convergence. This can be expressed as

$$\frac{\partial E}{\partial \omega_c} = \min\left(\alpha_{\omega_c}, \left|\frac{\partial E}{\partial \omega_c}\right|\right) \cdot \text{sign}\left(\frac{\partial E}{\partial \omega_c}\right), \tag{21}$$

$$\frac{\partial E}{\partial K} = \min\left(\alpha_K, \left|\frac{\partial E}{\partial K}\right|\right) \cdot \text{sign}\left(\frac{\partial E}{\partial K}\right), \tag{22}$$

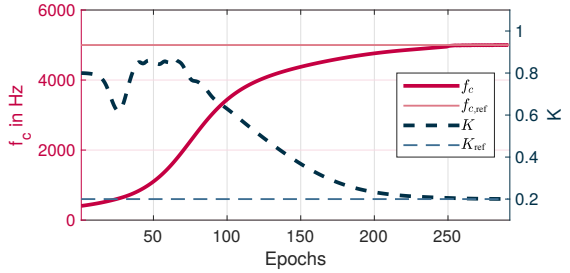
where α_{ω_c} and α_K are two positive small fractional scalars.

The parameters are finally updated using the gradient descent method given by

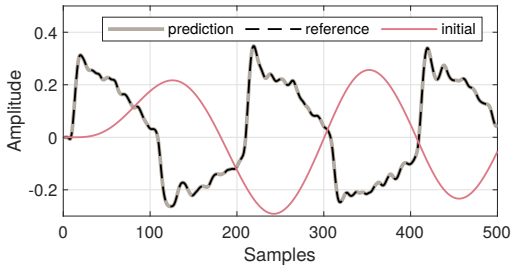
$$\omega_c := \omega_c - \eta_{\omega_c} \cdot \frac{\partial E}{\partial \omega_c}, \tag{23}$$

$$K := K - \eta_K \cdot \frac{\partial E}{\partial K}, \tag{24}$$

where η_{ω_c} and η_K are the corresponding learning rates.



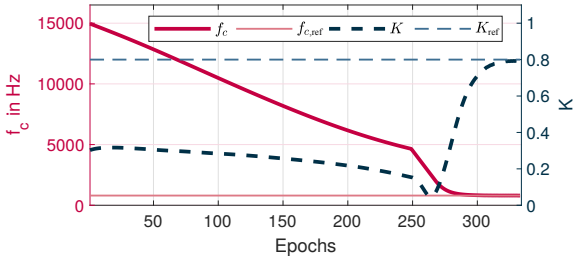
(a) *Parameter adaption curves per epoch.*



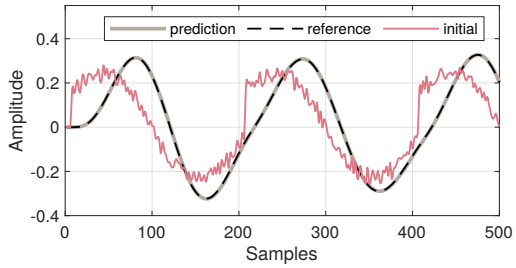
(b) *Predicted, reference and initial signals.*

Figure 4: *Adaption example for:* $f_{c,init} = 400$ *Hz,* $K_{init} = 0.8$*,* $f_{c,ref} = 5$ *kHz,* $K_{ref} = 0.2$

Figures 4 (a) and 4 (b) show two examples of the evolution of parameters during the adaption process per epoch. In the first example, the target cutoff frequency is set quite above the initial cutoff frequency while the target resonance coefficient is set much below the initial value. In the second example, the target cutoff frequency is set quite below the initial cutoff frequency while the target resonance coefficient is set much above the initial value. In both examples, one can see that the resonance parameter do not converge to a solution until the cutoff frequency gets close enough to its target. This is primarily because of a larger overall error gradient when the cutoff frequencies are far apart. The corresponding initial, reference, and predicted signals are depicted in Fig. 5 (a) and Fig 5 (b).



(a) *Parameter adaption curves per epoch.*



(b) *Predicted, reference and initial signals.*

Figure 5: *Adaption example for:* $f_{c,init} = 15\,kHz$, $K_{init} = 0.3$, $f_{c,ref} = 800\,Hz$, $K_{ref} = 0.8$

4. INTEGRATION IN MACHINE LEARNING

Figure 6 shows a block diagram to illustrate an example of integration of a differentiable subtractive synthesizer with a DL model. During training, the DL model learns to map a reference audio signal y to a set of estimated parameters $\hat{p}$. When applied to the synthesizer, this set of parameter produces an estimated audio output $\hat{y}$ that matches a reference audio input. This process, called tone matching or timbre matching, uses a loss function to evaluate the similarity between reference and estimated outputs. To achieve an update of the DL model weights based on this loss, the error needs to be backpropagated through the synthesizer with regard to its parameters. This should help the timbre matching process to perform better than a system that uses only parameter based loss [15].

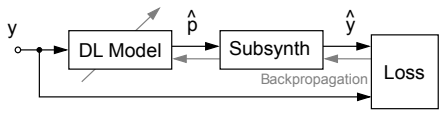


Figure 6: *Simple block diagram of a DL model for subtractive synthesizer parameter estimation.*

The audio signals are not directly fed into the DL model but are subject to pre-processing. A spectro-temporal representation is key to separate the timbre information and the slower temporal changes of the sounds.

Tone Matching or Timbre Matching, has been performed using classical optimization methods but recent approaches tend to use DL methods. While a DL model requires a lot of time for training because of the inclusion of the synthesizer in its end-to-end training process, its performance during testing or evaluation can be quite accurate and fast, particularly if the trained model is small. A DL model can lead to a qualitatively better performance than stochastic optimization based methods [16]. In this work though, instead of the entire synthesizer, only the Moog filter is experimented with and a simple sample based loss function is used to drive an end-to-end learning process.

5. CNN BASED PARAMETER ESTIMATION

In this section, an example application is described where a convolutional neural network (CNN) is used to estimate the control parameters of a Moog filter and Fig. 7 shows the corresponding block diagram. Initially, a dataset is created with a set of predefined input signals x and ground truth control parameters $[\omega_c, K]_{\text{ref}}$ for the Moog filter. The set of output signals y from the filter are collected and their spectral representations Y are used as the input to a CNN. The estimated parameters $[\hat{\omega}_c, \hat{K}]$ from the CNN and the corresponding set of input signals x are used with the Moog filter to generate the estimated output signal $\hat{y}$, The loss computed between y and $\hat{y}$ is used to train the CNN. It is important to mention that a loss between $[\omega_c, K]_{\text{ref}}$ and $[\hat{\omega}_c, \hat{K}]$ can drive the training process, but the goal of this work is to illustrate a successful training via backpropagation through the Moog filter. Additionally, if a CNN has to estimate more control parameters for a complex synthesizer in any later application, the problem might become ill-posed due to a possibility of many parametric solutions. A direct loss between audio signals or their spectral representations should be better for the semantics of sound perception. Both of the loss functions can be used together as well.

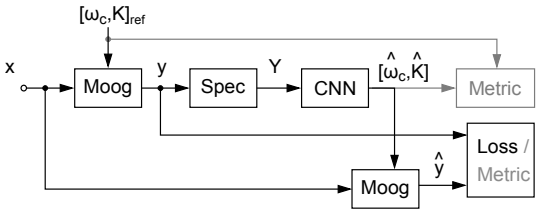


Figure 7: *Block diagram of a CNN integration for Moog filter parameter estimation.*

5.1. Dataset

The dataset used here is composed of 280 audio files at a sample rate of 44.1 Hz. These files are generated using MATLAB. An oscillator produces randomly weighted combinations of five different waveforms (sine, triangle, sawtooth, rectangle and white noise) at 440 Hz that are used as input for the Moog filter. These weights are set to add up to 1. The control parameters of the filter are selected uniformly in the range $[100, 15000]$ Hz for f_c and $[0.1, 0.9]$ for K. They are paired randomly to generate the reference or ground truth audio signals.

As pre-processing, magnitude spectrograms of the output signals from the filter are computed. A window size of 1024 samples, an overlap of 512 samples, and a 256 points DFT are used. The resulting matrix is then resized to match the input of the CNN (128×64). Finally, 256 samples of the input and output audio signals are selected during training to compute the loss. More samples result in longer training times without adding significant improvements.

5.2. CNN Architecture

The architecture of the CNN is illustrated in Fig. 8. The input to the CNN is a resized Spectrogram of the Moog filter output signal. The network has a simple feedforward architecture composed of four blocks. In each of the first three blocks, a convolution layer (Conv) is used followed by a rectified linear unit (ReLU) and a pooling layer (Pool). The Conv layers use filters of size $3 \times 3 \times d$, where d denotes the input feature map depth, and a stride of 2. 96 such filters are used in each Conv layer. The Pool layers perform maximum pooling and use a kernel size of 2×2 with a stride of 2. The ReLU layer uses a leakage factor of 0. The stride in Conv and Pool layers ensures the reduction of spatial dimensions of the feature maps. The final block is composed of a *fully connected layer* (FC) followed by a ReLU layer. The FC layer vectorizes the input feature map and delivers an output vector of the required size. A sigmoid layer can be used instead of a ReLU layer for faster convergence but boundary values of the estimated parameters suffer due to the saturation regions of the function.

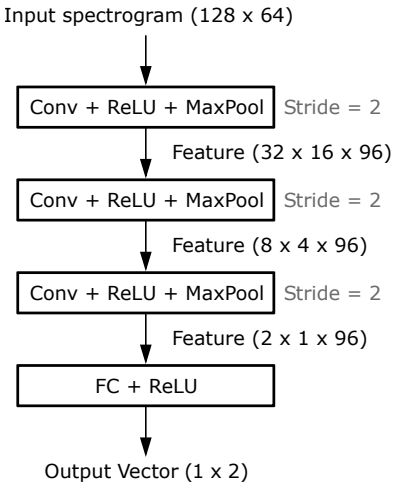
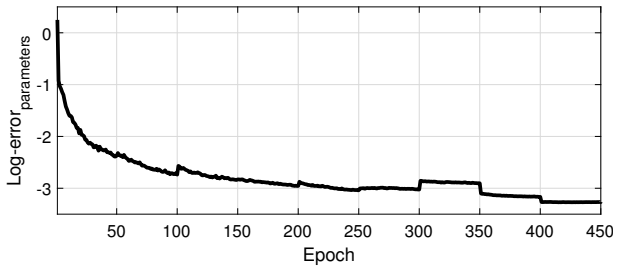


Figure 8: *Block diagram of the CNN architecture.*

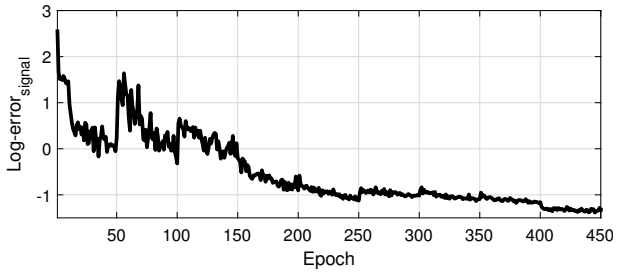
5.3. Model Performance

Initially, CNN models were constructed to be trained with raw audio snippets of multiple sample lengths. For longer audio snippets (> 1024 samples), the models converged initially but stagnated quickly. The models were then trained with the magnitude responses of the raw audio snippets. While the networks were able to reduce the loss and show convergent behavior, particularly for longer audio, the final results were not quite good. Finally, the representation described in Section 5.2 is selected as the CNN input. In order to train the model, the loss function given by Eq. (16) in section 3.2 is used. The gradients w.r.t. the parameters are also constrained by gradient clipping for a stable convergence. Similar to the adaptive Moog filter, the gradient w.r.t. the cut-off frequency is exponentially suppressed in order to assure a more stable convergence. The given model is then trained for about 1000 epochs where the learning rate is reduced linearly from 10^{-4} to 10^{-8}. A batch size of 8 is selected which results in 35 iterations per epoch and the *adam* optimizer [17] is used as the update method. The model is built in the last version of MatConvNet [12] deep learning toolbox for MATLAB. The training is performed on a machine with a Nvidia QUADRO RTX8000 graphical processing unit.

To measure the model performance, sum of squared error (SSE) function between the estimated outputs and the expected ground truths is used. Figure 9 (a) shows the logarithm of the SSE between the estimated $[\hat{\omega}_c, \hat{K}]$ and expected $[\omega_c, K]_{\text{ref}}$ parameters while Fig. 9 (b) shows the logarithm of the SSE between the estimated $\hat{y}(n)$ and expected $y(n)$ signals, measured over the epochs.



(a) *Parameters error.*



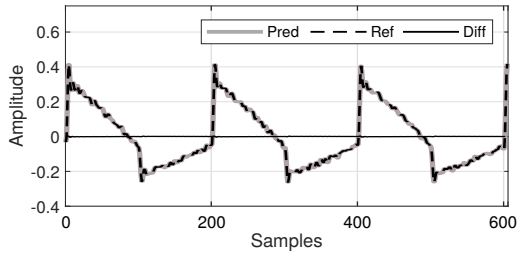
(b) *Signal Error.*

Figure 9: *Training error Curves over epochs.*

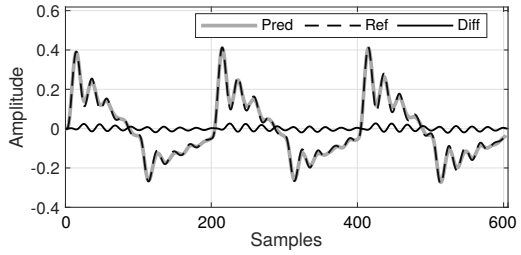
Both of the errors decrease relatively faster within the first 200 epochs and then converges slowly. The training can be performed until 600 epochs beyond which no improvement is achieved. As mentioned previously, the signal error for backpropagation is computed with 256 audio samples. Calculation of the loss with more samples might improve the results but will drastically increase the training time.

Figure 10 shows three audio examples with different types of signals and different sets of parameters. The first 1024 samples of the signals are shown. Figure 10 (a) shows an example ground truth signal generated by a Moog filter for a cutoff frequency of 11549 Hz and resonance coefficient of value 0.309. Based on the corresponding estimated cutoff frequency of 11564 Hz and resonance coefficient of value 0.315, the predicted signal is constructed, denoted by Pred in the plot. The difference between the reference and estimated signal, denoted by Diff, shows a near perfect reconstruction. Figure 10(b) shows another audio example where the predicted cutoff frequency of 3948 Hz is close to the ground truth cutoff frequency of 3867 Hz but the predicted resonance coefficient of value 0.624 is not as close to the ground truth value of 0.712. The difference or error signal is more prominent than the previous examples. The third audio example shown in Fig. 10 (c) has additive noise but the network prediction is still quite good as it closely matches the ground truth values and the difference signal has a low amplitude.

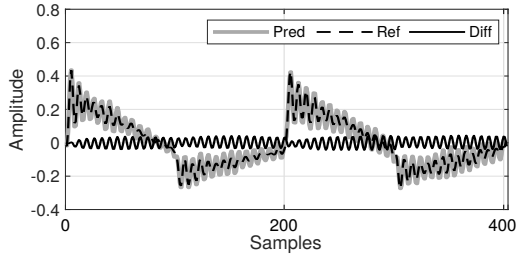
In general, it can be concluded that the network performs ad-



(a) $\hat{f}_c = 11564$, $\hat{K} = 0.315$, $f_{c,ref} = 11549$, $K_{ref} = 0.309$



(b) $\hat{f}_c = 3948$, $\hat{K} = 0.624$, $f_{c,ref} = 3867$, $K_{ref} = 0.712$



(c) $\hat{f}_c = 8858$, $\hat{K} = 0.963$, $f_{c,ref} = 9001$, $K_{ref} = 0.856$

Figure 10: *Direct comparison of signal examples.*

mirably for most examples across all parameter values in experiment. However, some uncertain results are also observed particularly around the lower values of the cutoff frequency. Training with a higher number of samples or including more examples might resolve such uncertainties and can be experimented as part of the future work. Additionally, the network training should also be performed with other forms of loss functions in time or frequency domain to find the best model in terms of error reduction and convergence. Further experiments should be conducted with multiple input representations for the CNN.

6. CONCLUSION

This work is a step toward the modeling of differentiable subtractive synthesizers, which would allow to perform tone matching based on psychoacoustic loss functions. Hence, the presented work should be extended towards the parameter estimation for an entire synthesizer and inclusion of loss functions considering audio semantics. As further work, other blocks of the synthesizer like oscillators, envelope generators and audio effects should be differentiated with respect to their parameters. Experiments should be conducted towards computation of the appropriate input representation for a given CNN model. Since synthesizers can have a large number of time or frequency dependent parameters, multiple time-frequency representations as the model input should be studied. Multiple loss functions in time and frequency domain should also be studied in order to find the most appropriate combination for training a model. Finally the CNN model and its modules should be studied in order to improve the estimation performance. Finally, the synthesizer could be implemented in a PyTorch environment and the performance could be observed and improved.

7. REFERENCES

[1] Robert A. Moog, "A voltage-controlled low-pass high-pass filter for audio signal processing," *Journal of the Audio Engineering Society*, october 1965.

[2] Effrosyni Paschou, Fabian Esqueda, Vesa Välimäki, and John Mourjopoulos, "Modeling and measuring a moog voltage-controlled filter," december 2017, pp. 1641–1647.

[3] Tim Stilson and Julius Smith, "Analyzing the moog VCF with considerations for digital implementation," 1996.

[4] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts, "DDSP: differentiable digital signal processing," in *International Conference on Learning Representations*, 2020.

[5] Purbaditya Bhattacharya, Patrick Nowak, and Udo Zölzer, "Optimization of cascaded parametric peak and shelving filters with backpropagation algorithm," in *23rd International Conference on Digital Audio Effects (DAFx)*, september 2020.

[6] Boris Kuznetsov, Julian Parker, and Fabian Esqueda, "Differentiable IIR filters for machine learning applications," in *23rd International Conference on Digital Audio Effects (DAFx)*, 2020.

[7] Cheng-Zhi Anna Huang, David Duvenaud, Kenneth C. Arnold, Brenton Partridge, Josiah W. Oberholtzer, and Krzysztof Z. Gajos, "Active learning of intuitive control knobs for synthesizers using gaussian processes," New York, NY, USA, 2014, Association for Computing Machinery.

[8] Matthew D. Hoffman and Perry R. Cook, "Feature-based synthesis: Mapping acoustic and perceptual features onto synthesis parameters," in *International Conference on Mathematics and Computing*, 2006.

[9] Matthew Yee-King and Martin S. Roth, "Synthbot: An unsupervised software synthesizer programmer," 2009.

[10] Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos, "Universal audio synthesizer control with normalizing flows," in *International Conference on Digital Audio Effects (DaFX 2019)*, Birmingham, United Kingdom, september 2019.

[11] Vesa Välimäki, Stefan Bilbao, Julius O. Smith, Jonathan S. Abel, Jyri Pakarinen, and David Berners, *DAFX: Digital Audio Effects*, chapter 12: Virtual analog effects, pp. 279–320, John Wiley & Sons, Ltd, 2011.

[12] Andrea Vedaldi and Karel Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proceedings of the 23rd ACM International Conference on Multimedia*, New York, NY, USA, 2015, MM '15, p. 689–692.

[13] François Chollet et al., "Keras," `https://keras.io`, 2015.

[14] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[15] Matthew John Yee-King, Leon Fedden, and Mark d'Inverno, "Automatic programming of vst sound synthesizers using deep networks and other techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, 2018.

[16] Matthew Yee-King and Martin Roth, "A comparison of parametric optimisation techniques for musical instrument tone matching," in *Audio Engineering Society Convention 130*, january 2011.

[17] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

DIFFERENTIABLE GREY-BOX MODELLING OF PHASER EFFECTS USING FRAME-BASED SPECTRAL PROCESSING

Alistair Carson *

Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
alistair.carson@ed.ac.uk

Cassia Valentini-Botinhao

Centre for Speech Technology Research
University of Edinburgh
Edinburgh, UK
cvbotinh@inf.ed.ac.uk

Simon King

Centre for Speech Technology Research
University of Edinburgh
Edinburgh, UK
simon.king@ed.ac.uk

Stefan Bilbao

Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
sbilbao@ed.ac.uk

ABSTRACT

Machine learning approaches to modelling analog audio effects have seen intensive investigation in recent years, particularly in the context of non-linear time-invariant effects such as guitar amplifiers. For modulation effects such as phasers, however, new challenges emerge due to the presence of the low-frequency oscillator which controls the slowly time-varying nature of the effect. Existing approaches have either required foreknowledge of this control signal, or have been non-causal in implementation. This work presents a differentiable digital signal processing approach to modelling phaser effects in which the underlying control signal and time-varying spectral response of the effect are jointly learned. The proposed model processes audio in short frames to implement a time-varying filter in the frequency domain, with a transfer function based on typical analog phaser circuit topology. We show that the model can be trained to emulate an analog reference device, while retaining interpretable and adjustable parameters. The frame duration is an important hyper-parameter of the proposed model, so an investigation was carried out into its effect on model accuracy. The optimal frame length depends on both the rate and transient decay-time of the target effect, but the frame length can be altered at inference time without a significant change in accuracy.

1. INTRODUCTION

A broad class of audio effects found in almost all genres of popular music is that of time-varying modulation effects, and includes phasing, flanging, chorus, and tremolo. Model-based digital implementations of these effects are straightforward [1], but many musicians prefer the timbre and character of the original analog or electro-mechanical devices used to create these effects, and these may be considerably more difficult to model. Circuit-based simulations [2, 3] can produce physically accurate results but are highly optimized to a specific device, and require complete knowledge of the circuit and its component values.

In general, modelling of audio effects using machine learning has become an active area of research in recent years, with a particular focus on modelling non-linear time-invariant effects such as guitar amplifiers and distortion pedals. Approaches in modelling these systems include fully black-box methods using recurrent or convolutional neural networks (RNNs / CNNs) [4, 5], as well as grey-box models which use some prior knowledge of the reference system, such as differentiable state-space models [6, 7] and differentiable DSP-based models [8, 9]. Modelling of effects with time-varying input-dependent behaviour such as dynamic range compression has also been explored through the use of CNNs with long receptive fields [10] and temporal feature-wise linear modulation [11]; as well as a differentiable DSP model proposed in [12]. Modelling of time-varying modulation effects presents a unique challenge due to the modulation of system behaviour by a low frequency oscillator (LFO). The deep-learning approach proposed in [13] can emulate a wide range of effects, however the use of bi-directional long-term short-memory networks (LSTMs) makes the model non-causal and therefore not suited for real-time use. Wright et al. [14] proposed a real-time model for phasing and flanging effects using RNNs, but required manual measurement and estimation of the LFO prior to training as this was an input to the model. Earlier work by Kiiski et al. [15] proposed a grey-box model of phasing, but in which no machine learning was used.

This work presents a differentiable DSP model of a phaser effect that can be trained through gradient descent to jointly learn the underlying LFO signal and the time-varying spectral response of an analog reference pedal. The model utilises frequency domain approximations of IIR filters to accelerate training times, as has been employed in [9, 12]. Through experiment, we investigate the conflicting demands of time and frequency resolution when using this method in the context of time-varying effects.

The paper is structured as follows: Section 2 provides background on analog and digital phasing effects; Section 3 outlines the proposed model; Section 4 describes the target systems and data; Section 5 describes the experimental procedure and results; and Section 6 concludes the paper with an outlook on areas of future work. Source code and audio examples are provided at the accompanying web-page [16].

* A Carson is funded by the Scottish Graduate School of Arts and Humanities (SGSAH).

2. ANALOG AND DIGITAL PHASE SHIFTERS

The phaser is a time-varying filter effect in which the phase of an input signal is subject to a periodic modulation and combined together with the 'dry' (uneffected) signal to create audible notches in the frequency spectrum [17]. The movement of these notches with time gives the characteristic perceived sweeping effect. The phasing effect is often confused with flanging, and indeed both effects are caused by modulating notches in the spectra. The key difference is that a flanger generates an infinite series of harmonically spaced notches, whereas the phaser response has finite non-uniformly spaced notches [18]. Furthermore, implementations of phasers and flangers differ both historically and in present day commercial products. Originally, the flanging effect was created by playing two tape machines in near unison with a small variable time delay between the two reels [19]. Nowadays, flangers are typically implemented using time-varying digital comb filters, or 'bucket-brigade' delay lines in analog pedals. In contrast, phasers create phase shifts through cascaded all-pass filters, in which break frequencies are modulated by a low-frequency oscillator (LFO).

Digital phaser effects are usually implemented as linear time-varying signal-based models, in which the all-pass filters are discrete-time approximations of idealised all-pass filters [1, 18, 20]. The motivation of this work is to explore whether such a model can be embedded in a machine learning framework to emulate the response of an analog phaser pedal. The continuous and discrete time signal processing concepts of phasing, which underpin the proposed model, are outlined in the remainder of this section.

2.1. Continuous-time phasing

We will examine phasers of the topology shown in Figure 1 composed of a cascade of K identical first-order all-pass filters. (In many practical implementations of the phaser, $K = 4$ [18], but other choices of K have been employed [15].) The continuous-time transfer function of each such first-order section is [18]:

$$A(s) = \frac{s - \omega_b}{s + \omega_b} \tag{1}$$

where $s = \sigma + j\omega$ is the complex frequency and $\omega_b \geq 0$ is known as the *break-frequency* of the all-pass filter. Because A is all-pass, for real frequencies $s = j\omega$ it may be written as $A = e^{j\Theta}$, where:

$$\Theta(\omega) = \pi - 2\arctan\left(\omega/\omega_b\right) . \tag{2}$$

From (2) it can be observed that ω_b is the frequency at which $\Theta(\omega_b) = \pi/2$. Furthermore, the all-pass filter inverts DC, and the phase response tends to zero as $\omega \to \infty$.

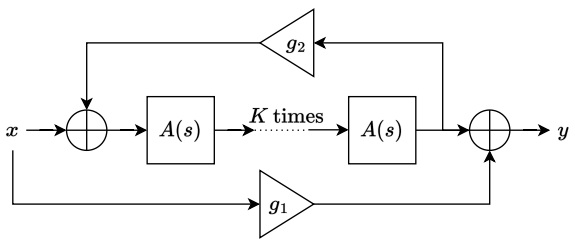


Figure 1: *A typical phaser structure in continuous time.*

The cascade arrangement in Figure 1 includes a through path with gain g_1, and a feedback path with gain g_2, with $0 \leq g_2 < 1$.

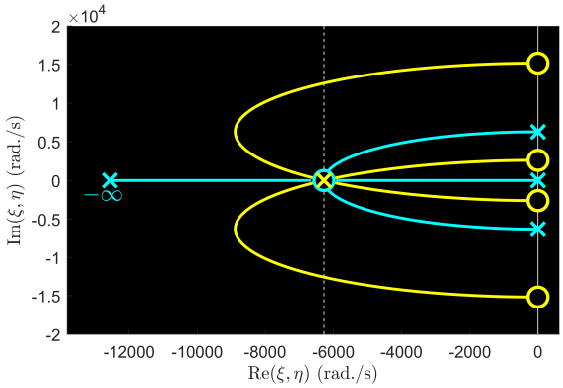


Figure 2: *Root locus plot, for $g_1 = 1$, and for for g_2 between 0 (open loop, indicated by circles) and 1 (indicated by crosses). Pole locations are indicated by blue lines, and zero locations by yellow lines. The real frequency $-\omega_b$ (where here, $\omega_b = 2\pi \cdot 1000$ rad/s) is indicated by a dashed line.*

The following transfer function results:

$$H(s) = g_1 + \frac{A^K}{1 - g_2 A^K} . \tag{3}$$

The poles $s = \xi_k$, and zeros $s = \eta_k$, $k = 0 \ldots, K-1$, are:

$$\xi_k = \omega_b \frac{1+\lambda_k}{1-\lambda_k} \quad \text{where } \lambda_k = \frac{e^{j2\pi k/K}}{\sqrt[K]{g_2}} \tag{4a}$$

$$\eta_k = \omega_b \frac{1+\beta_k}{1-\beta_k} \quad \text{where } \beta_k = \sqrt[K]{\frac{g_1}{1-g_1 g_2}} e^{j\pi(2k+1)/K} \tag{4b}$$

Figure 2 shows a root locus plot of the pole and zero trajectories for $0 \leq g_2 \leq 1$, and when $g_1 = 1$. Under open loop ($g_2 = 0$) conditions, two notches are located on the non-negative imaginary axis. As $g_1 \to 0$ the zeros move away from the imaginary axis and as such, parameter g_1 controls the perceived depth of the phaser effect. Under fully closed loop conditions, two poles are located on the non-negative imaginary axis (one at DC).Variations in the transfer function magnitude $|H|$ with both g_1 and g_2 are shown in Figure 3.

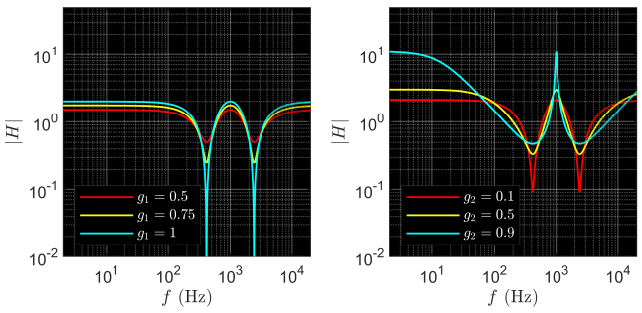


Figure 3: *Magnitude response for H, under open-loop conditions at left, for different values of g_1, as indicated, and under closed-loop conditions at right, for $g_1 = 1$, and different values of g_2, as indicated. As before, $\omega_b = 2\pi \cdot 1000$ rad/s.*

2.2. Discrete-time phasing

The continuous-time transfer function (1) can be approximated in discrete-time via the bilinear transform to give the discrete-time all-pass section, $A_d(z)$, defined as:

$$A_d(z) = \frac{p - z^{-1}}{1 - pz^{-1}} \qquad \text{where} \qquad p = \frac{1 - \tan(\omega_b T/2)}{1 + \tan(\omega_b T/2)} \,. \tag{5}$$

Here, $T = 1/F_s$ is the sampling period, for sample rate F_s in Hz. The corresponding discrete-time phaser is shown in Figure 4. The discrete-time transfer function $H_d(z)$ of K cascaded all-pass sections, including the residual connection and feedback loop is:

$$H_d(z) = g_1 + \frac{A_d^K}{1 - g_2 z^{-\phi} A_d^K} \,. \tag{6}$$

Note that a digital delay $z^{-\phi}$, $\phi \in \mathcal{Z}^+$ has been included in the feedback loop. Without this, a delay-free loop would be present in the resulting difference equation. Efficient modelling of delay-free loops found in phaser pedals has proven challenging: the state-space model of a phaser pedal presented in [3] required a Newton-Raphson solver at run-time; and Kiiski et al. reported that the fictitious delay line in their DSP model ($\phi = 1$) resulted in perceptual differences in the feedback effect when compared with the analog reference device [15]. The effect of the delay line on the magnitude response of the system can be seen in Figure 5. It is clear that the fictitious delay line in the feedback loop drastically changes the magnitude response for high-frequencies, and should ideally be avoided in virtual analog models of phasers.

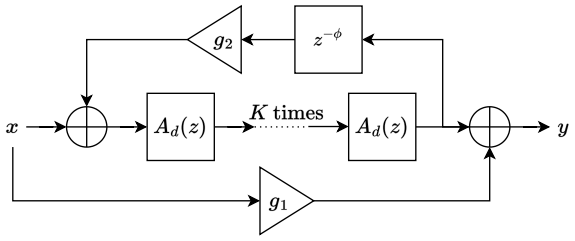


Figure 4: *A typical DSP phaser structure.*

2.3. Time-varying behaviour

In analog phase shifters, the break frequency of the all-pass sections, ω_b, is periodically modulated by a sub-audio rate LFO, typically with frequency in the range 0.05 Hz to 5 Hz. The mapping between LFO voltage and break frequency is often nonlinear and asymmetrical: for example, in the MXR Phase 90 the LFO varies the voltage across JFETs which in turn alters the break frequency of the all-pass [3]. Certain devices, such as the Uni-Vibe and its replicas, are optical: the LFO varies the voltage across a light source (an incandescent bulb in original units [21]) surrounded by light dependent resistors (LDRs). The resulting fluctuation in current through the LDRs controls the all-pass break frequencies. Accurate prediction of the LFO characteristics (including fundamental frequency and waveshape) are therefore critical to the quality of discrete-time emulations of such effects, including phasers.

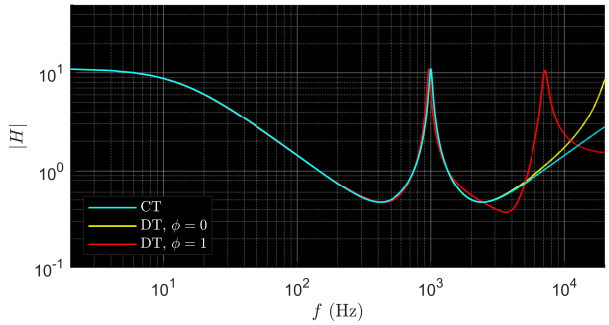


Figure 5: *Comparison of magnitude responses of continuous-time (CT) and discrete-time (DT) phaser models, with constant parameters* $\omega_b = 2\pi \cdot 1000$ *rad/s,* $g_1 = 1.0$*,* $g_2 = 0.9$*, and* $T = 1/44100$ *s. Distinct cases of the discrete-time model without* ($\phi = 0$) *and with* ($\phi = 1$) *a fictitious delay are illustrated.*

3. MODELLING METHOD

This section outlines the proposed model of a differentiable phase shifting algorithm which, given input-output audio recordings of a reference device, can be trained to emulate the time-varying behaviour. Consider an arbitrary phase-shifting device that has been sampled in discrete time as:

$$y[n] = f(x[n], \theta[n]) \tag{7}$$

where x is the input signal, y is the output signal, θ are the time-varying parameters of the system, n is the sample index and f is a linear function of x. We seek to develop a model of the form:

$$\hat{y}[n] = g(x[n], \hat{\theta}[n]) \tag{8}$$

whose output $\hat{y}[n]$ is perceptually indistinguishable from $y[n]$. The function g is assumed differentiable so that the model can be trained using gradient descent to find model parameters $\hat{\theta}$ that minimise an objective loss function $\mathcal{L}(y, \hat{y})$. The proposed model architecture is shown in Figure 6.

3.1. Frame-based processing

Utilising frame-based spectral processing [22, 23], the proposed model assumes that the target phaser can be treated as a linear time-invariant (LTI) system over the duration of a short frame of length W seconds. Suppose that an input audio signal $x[l]$ of length L samples is segmented into $N_f = \lceil L/H \rceil$ frames of length $N = \lfloor WF_s \rfloor$ with hop-size H samples. (The resulting frame rate is $F_f = F_s/H$ in Hz.) The mth frame, $m = 0, \ldots, N_f - 1$, is defined as the $N \times 1$ column vector $\mathbf{x}_m = [x[mH], \ldots, x[mH + N - 1]]^T$. The short-time Fourier transform vector $\mathbf{X}_m$, $m = 0 \ldots, N_f - 1$ is derived from $\mathbf{x}_m$ through windowing and Fourier transformation as follows:

$$\mathbf{X}_m = \mathbf{U}\mathbf{Q}\mathbf{x}_m \,. \tag{9}$$

Here, $\mathbf{Q} = [\mathbf{W}\ \mathbf{Z}]^T$ is an $N' \times N$ windowing matrix, where where N' is the DFT length. It includes the diagonal $N \times N$ matrix $\mathbf{W}$, containing samples of the Hann window on the diagonal, and an $N \times (N' - N)$ all-zero matrix $\mathbf{Z}$ implementing zero-padding. $\mathbf{U}$ is an $N' \times N'$ DFT matrix. At each frame, spectral

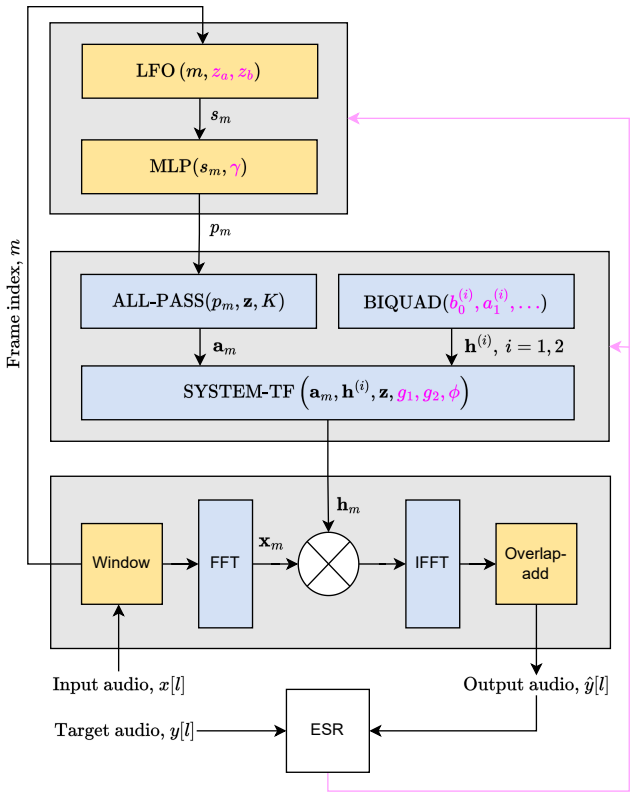


Figure 6: *Structure of the proposed model. Black arrows indicate signal flow, with magenta indicating the flow of gradients to the learnable parameters. The blue and yellow boxes indicate operations in the frequency domain and time domain respectively. The subscript m denotes parameters varying at the frame rate, with all other parameters held constant for the duration of a training epoch. Parameters in magenta are updated once per epoch by the optimizer.*

processing is applied via element-wise complex multiplication in the frequency domain, followed by an inverse Fourier transform, truncation and windowing to yield the $N \times 1$ output vectors $\mathbf{y}_m$, $m = 0, \ldots, N_f - 1$:

$$\mathbf{y}_m = \frac{1}{N'} \mathbf{Q}^T \mathbf{U}^* \mathbf{H}_m \mathbf{X}_m \,. \tag{10}$$

Here, $\mathbf{H}_m$ is an $N' \times N'$ diagonal matrix containing values of a transfer function (incorporating Hermitian symmetry) at frame m on its diagonal. $\mathbf{U}^*$ is the conjugate transpose of the DFT matrix $\mathbf{U}$. Finally, the output time series $\hat{y}[l]$ is obtained from the frames $\mathbf{y}_m$ through an overlap-add procedure, with hop size H. It can be shown that for $\mathbf{H}_m = \mathbf{I}$, exact reconstruction of the input signal can be obtained if N/H is an integer greater than two—this property is known as constant-overlap-add and is enforced in the proposed model. Considering the transfer function in (10) to represent a digital filter with $M \in \mathbb{Z}$ non-zero taps, then the DFT length must be $N' \geq N + M - 1$ to avoid temporal aliasing in the output frames. For IIR filtering (where $M \to \infty$), some degree of temporal aliasing is inevitable but can be practically suppressed by choosing M as the 60dB decay time (in samples) of the filter. In this work the decay time of the target system was not known, but we found a DFT length of $N' = 2^{\lceil \log_2(N) \rceil}$ to be sufficient to train the models.

3.2. LFO generator

The proposed model has an LFO module which governs the time-varying behaviour of the spectral processing. The LFO produces samples at the frame rate, F_f, and is defined as the real part of a damped complex exponential:

$$s_m(z_a, z_b) = \mathrm{Re}(z_b z_a^m) = |z_b||z_a|^m \cos(m\angle z_a + \angle z_b) \tag{11}$$

where $z_a, z_b \in \mathbb{C}$ are the complex frequency and complex amplitude respectively. Hayes et al. showed that Wirtinger's calculus can be used to compute the partial derivatives of real-valued, complex-variable functions such as (11) to enable sinusoidal frequency estimation by gradient descent [24]. In this work we extend this method to include a learnable starting phase and amplitude (defined by z_b). The Wirtinger derivatives of (11) are:

$$\frac{\partial s_m}{\partial z_a} \triangleq \frac{1}{2}\left(\frac{\partial s_m}{\partial \mathrm{Re}(z_a)} - j\frac{\partial s_m}{\partial \mathrm{Im}(z_a)}\right) = \frac{m z_b z_a^{m-1}}{2} \tag{12a}$$

$$\frac{\partial s_m}{\partial z_b} \triangleq \frac{1}{2}\left(\frac{\partial s_m}{\partial \mathrm{Re}(z_b)} - j\frac{\partial s_m}{\partial \mathrm{Im}(z_b)}\right) = \frac{z_a^m}{2} \,. \tag{12b}$$

In the model implementation, the LFO parameters were initialised to:

$$z_a = 0.7 \exp(j\zeta / F_f), \quad z_b = 1.0 \tag{13}$$

where ζ is a random number sampled from a standard normal distribution. The damped amplitude envelope was only applied during training; during inference z_a was normalised to the unit circle to give a lossless LFO.

3.3. Multi-layer perceptron waveshaper

The output of the LFO generator is passed through a multi-layer perceptron (MLP) to allow the model to learn non-sinusoidal control signals like those described in Section 2.3. The conceptual motivation for this module was to emulate the linear and non-linear mapping of the LFO signal (in volts or amperes) to the break-frequencies of the all-pass filters (in $\mathrm{rad\,s^{-1}}$) in an analog phaser device. Therefore we treat the output of the MLP as the model's prediction of the normalised time-varying break-frequency signal, such that the all-pass parameter is given by:

$$p_m = \frac{1 - \tan(d_m)}{1 + \tan(d_m)} \quad \text{where} \quad d_m = \mathrm{MLP}(s_m, \gamma) \,, \tag{14}$$

with γ being the MLP parameters. The MLP consisted of three hidden layers; with 8 neurons per layer; hyperbolic tangent activation functions in the hidden layers; and linear activation in the output layer. The input and output features were scalar, based on the assumption that all K all-pass filters are identical in the reference device.

3.4. Model transfer function

The frame-dependent transfer function of the model has the form:

$$\mathbf{h}_m \triangleq \mathrm{diag}(\mathbf{H}_m) = \mathbf{h}^{(1)} \cdot \left(g_1 + \frac{\mathbf{h}^{(2)} \cdot \mathbf{a}_m}{1 - |g_2| \mathbf{z}^{-I(\phi)\phi} \cdot \mathbf{h}^{(2)} \cdot \mathbf{a}_m}\right) \tag{15}$$

where $\mathbf{a}_m$ is the frame-dependent all-pass kernel:

$$\mathbf{a}_m = \left(\frac{p_m - \mathbf{z}^{-1}}{1 - p_m \mathbf{z}^{-1}} \right)^K \tag{16}$$

and $\mathbf{z}$ is the N'- element vector $\mathbf{z} = [e^{\frac{2\pi j(0)}{N'}}, \ldots, e^{\frac{2\pi j(N'-1)}{N'}}]^T$. (Here and elsewhere in this section, operations on vectors are assumed to be applied element-wise.) The parameters $g_1, g_2, \phi \in \mathbb{R}$ retain their physical meanings from Section 2 but are now initialised as learnable parameters of the model. $I(\cdot)$ is the Heaviside step function and was included to prevent the model learning a non-casual transfer function. The number of all-pass filters is a hyper-parameter and was fixed for all experiments to $K = 4$. The model includes two frequency domain representations of bi-quad filters, given by:

$$\mathbf{h}^{(i)} = \frac{b_0^{(i)} + b_1^{(i)} \mathbf{z}^{-1} + b_2^{(i)} \mathbf{z}^{-2}}{1 + a_1^{(i)} \mathbf{z}^{-1} + a_2^{(i)} \mathbf{z}^{-2}} \tag{17}$$

where $b_0^{(i)}, b_1^{(i)}, b_2^{(i)}, a_1^{(i)}, a_2^{(i)} \in \mathbb{R}$ are the learnable filter parameters for the ith biquad, $i = 1, 2$. These kernels were included to account for any further LTI filtering that an analog phaser might impart in addition to the core phasing effect described in Section 2. For example, low-pass filtering, DC blocking or gain adjustment.

3.5. Loss function

The proposed model uses the error-to-signal ratio (ESR) as the objective loss function during training:

$$\mathcal{L}(y, \hat{y}) = \frac{\sum_{l=0}^{L-1} (y[l] - \hat{y}[l])^2}{\sum_{l=0}^{L-1} y[l]^2} \tag{18}$$

where L is the length of the training data in samples. This loss function has been widely used in black-box and grey-box modelling of other audio effects [5, 12, 14], but the strict time-alignment required by (18) introduced some interesting challenges when it came to training the proposed model. Because the frequency and phase of the LFO are unknown parameters, the training data cannot be arbitrarily split into short segments (as is common when using long audio sequences as training data [5, 11]). For example, even if the initial random frequency guess was precisely correct, the model would 'see' a different starting phase for each segment (unless the segment length happened to be an integer multiple of the LFO period). In initial experiments, this phase discrepancy was accounted for using the current estimate of LFO frequency, but this caused noisy optimizer updates and convergence issues when the segment length was shorter than one LFO period in the target data.

This presents a dilemma in training this model: we need a sufficient duration of training data to capture the slowly time-varying features in the target system, but are constrained to training with single-batch gradient descent, meaning the time taken for one optimizer step increases linearly with the length of training data. The proposed solution to this problem was to use a short, spectrally-flat training signal. Details of this signal are outlined in Section 4.

3.6. Training details

All models were trained on audio with a sample rate of $44.1\,\mathrm{kHz}$ using an Adam optimizer [25] with an initial learning rate of 10^{-3}. Models were trained for a maximum of 5000 epochs on a NVIDIA Titan-X GPU. The training times varied depending on the length of training data and window size. For example, for an audio sequence of $10\,\mathrm{s}$ the training times were $\sim$3 hours and $\sim$16 hours for window lengths of $160\,\mathrm{ms}$ and $10\,\mathrm{ms}$ respectively. It is important to note that early designs of the proposed model were implemented in the time-domain and trained via back-propagation through time, but training times were deemed too slow to pursue this approach further ($\sim$24 hours for 1000 epochs on 1s of training audio). The prohibitive training times of IIR filters has been reported in previous work [9] [12].

3.7. Model inference

In this work, at model inference we use the same algorithm as in training (i.e. using frame-based spectral processing). This has the limitation of introducing a minimum latency of W seconds into the system, which may be unsuitable for real-time use. A time-domain implementation using IIR filters could conceivably be derived via inverse z-transform of the system transfer function (15), but this is left as a task for future work. The handling of the possibly non-integer delay-line length ϕ would require some consideration, but could be implemented with an all-pass filter in the feedback loop shown in Figure 4.

4. TARGET SYSTEMS AND DATASETS

A custom dataset was collected consisting of 60s of a synthetic chirp-train signal followed by 60s of direct-input (DI) guitar recordings. The chirp-train signal was used as the input signal for model training; whereas the guitar recordings were reserved for testing.

4.1. Synthetic training signal

The chirp-train signal was synthesised as an impulse train with period $30\,\mathrm{ms}$ passed through a cascade of 64 all-pass filters (5) with $p = 0.9$. This type of signal has been used previously for estimating the LFO frequency and shape of LTV audio effects [14, 15]. Due to its spectral flatness, it was hypothesised that even a few seconds of this signal would be sufficient to train the model.

4.2. Digital phaser

As a simplified test problem, the proposed model was initially trained on data generated through a digital phaser with transfer function (6) ($K = 4$) implemented through a time-domain recursion in MATLAB. This can be viewed as a specific instance of the model itself but without frame-based spectral processing and under known parameters, shown in Table 2. The LFO was set to a triangular wave, sweeping through break-frequencies from $4000\,\mathrm{rad\,s^{-1}}$ to a maximum of $16\,000\,\mathrm{rad\,s^{-1}}$. The maximum $60\,\mathrm{dB}$ decay time of this system (occurring at the minimum of the LFO cycle) was measured to be $t_{60} = 38\,\mathrm{ms}$.

4.3. EHX Small Stone

The Electro-Harmonix (EHX) Small Stone is a commonly encountered analog phaser pedal. The pedal has a single knob to control the rate of the effect, and a binary "colour" switch. High-level analysis of a circuit schematic [26] showed that it consisted of an LFO module, a series of four first order all-pass filter sections and a feedback loop (engaged when the "colour" switch is on). This circuit therefore shares the same topology as the proposed model.

The Small Stone data was collected by processing both the training and testing audio through the pedal in one continuous recording. The audio was sent to the pedal via the output of a PreSonus Audiobox i2 interface, and the output of the pedal re-connected to the input of the audio interface. A calibration recording was obtained with the pedal in bypass-mode and used as the input to the models to negate the effect of the recording equipment on model training. Six unique parameter configurations were captured: three different positions of the 'rate' knob with colour switch ON (circuit with feedback) and colour switch OFF (no feedback). The LFO rates were estimated through manual inspection of the spectrogram, providing a pseudo-ground-truth f_0 which could later be compared to the learned LFO signals — see Table 1.

Table 1: *Manually estimated LFO rates of the Small Stone under different parameter configurations. * denotes approximate values.*

Label	Colour	Rate knob position	T_0* [s]	f_0* [Hz]
SS-A		3 o'clock	0.44	2.28
SS-B	OFF	12 o'clock	1.60	0.625
SS-C		9 o'clock	11.6	0.086
SS-D		3 o'clock	0.70	1.4
SS-E	ON	12 o'clock	2.56	0.38
SS-F		9 o'clock	18	0.056

5. EXPERIMENTS AND RESULTS

The focus of the experiments presented here is on the effect of the window length W (in seconds) on model accuracy in the context of both model training and inference. In all experiments, the accuracy metric was the resultant ESR (18) on the test dataset. It was noted that the training convergence was sensitive to the initialisation of the MLP parameters, γ. To address this issue, each training procedure was re-initialised and repeated three times. The iteration with the lowest ESR was retained. A frame overlap of 75% was used across all experiments, with the frame length in samples N truncated to a multiple of four to ensure constant-overlap-add [18].

5.1. Experiment 1: training frame size sweep

As an initial experiment, instances of the model were trained with frame lengths ranging from $10\,\mathrm{ms}$ to $160\,\mathrm{ms}$ on the following data:

(a) Digital phaser with LFO rate $T_0 = 2\,\mathrm{s}$ (DP-2).

(b) Small Stone with parameter configuration A (SS-A).

(c) Small Stone with parameter configuration D (SS-D).

The training data was truncated to $2.67\,\mathrm{s}$ in duration to accelerate training, and was deemed sufficient given that it contained at least one LFO cycle for all case studies.

The ESR obtained in experiment 1 can be seen in Figure 7. In the case of the digital phaser, frame lengths of $40\,\mathrm{ms}$ to $160\,\mathrm{ms}$ all resulted in a error-to-signal of less than 1% on the testing data – implying an accurate match between the model output and target waveform with minimal artefacts introduced by windowing. The frame lengths of $10\,\mathrm{ms}$ and $20\,\mathrm{ms}$ produced worse results, suggesting an insufficient number of bins in the transfer function shape the spectrum and/or severe time aliasing in the output. The learned parameters during this experiment can be found in Table 2 and show a good estimation of the parameters in the target model. Figure 8 shows the synthesised LFO signals, compared to the target triangular wave.

In the case of the Small Stone, the resulting model accuracy depended on the presence of feedback in the circuit. The minimum test loss was approximately 1.5% without feedback and 10% with feedback. This result is expected due to the increase in circuit complexity and longer decay time associated with the feedback case. In both cases, window lengths of $40\,\mathrm{ms}$ and $80\,\mathrm{ms}$ provided the best performance—suggesting a good trade-off between time and frequency resolution. Despite the discrepancies in numerical results, the perceptual differences are difficult to distinguish, informally, from the target system—however, some differences in the low-frequencies are noted for short window lengths. The reader is referred to the accompanying web-page for audio examples. It is interesting to observe the learned LFO signals of the Small Stone, as shown in Figure 9. In both cases, the MLP module has consistently predicted a similar wave-shape across the frame-rates. When engaged, the colour switch appears to increase both the depth and the period of break-frequency modulation.

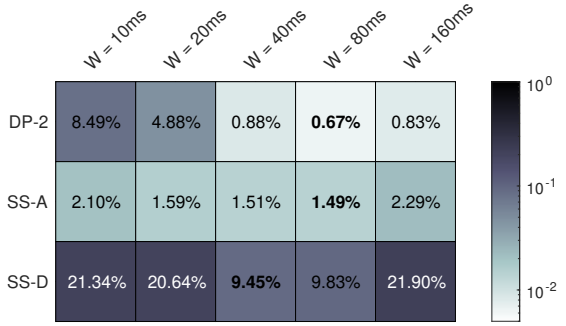


Figure 7: *ESR for different training window lengths W on the test audio of three case studies: digital phaser with $T_0 = 2$s (DP-2), Small Stone with parameter configurations A and D (SS-A, SS-D).*

5.2. Experiment 2: training frame size vs LFO rate

Experiment 2 investigated the effect of frame-length on model accuracy in more detail, considering as a case study the digital phaser with LFO periods $T_0 = 0.5\,\mathrm{s}$, $2\,\mathrm{s}$ and $8\,\mathrm{s}$ The length of training data was held constant at $10\,\mathrm{s}$, and the prior knowledge of target LFO periods informed the choice of frame-lengths:

$$W_b = T_0 2^{b/2}/100 \quad \text{where} \quad b = 0, \ldots, 10 \tag{19}$$

Figure 10 (top) shows the results of the experiment, with minimum test loss against training frame size for different rates of phaser effect. Firstly, we see that the model accuracy increases for longer LFO periods. This is intuitive, as in the limit $T_0 \to \infty$ the target system becomes linear and time-invariant (LTI) so we expect the artefacts of frame-based processing to diminish. Also intuitively, the results suggest the optimum window length depends on the target LFO period. In the bottom figure, the window length has been normalised to target LFO period. In this case, the optimum W/T_0 ratio shows consistency across the phaser rates, with $W/T_0 \approx 5\%$ giving, on average, the lowest loss.

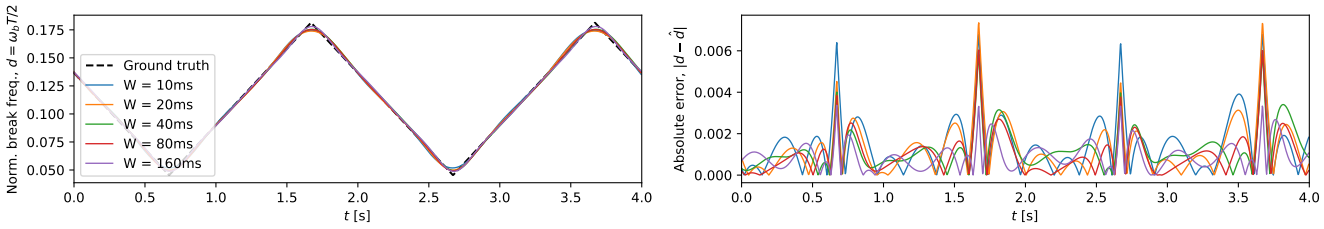


Figure 8: *Outputs of the MLP module (left) and the absolute error (right) compared to the triangular LFO in the target digital phaser, with $T_0 = 2s$ (plotted left as ground truth).*

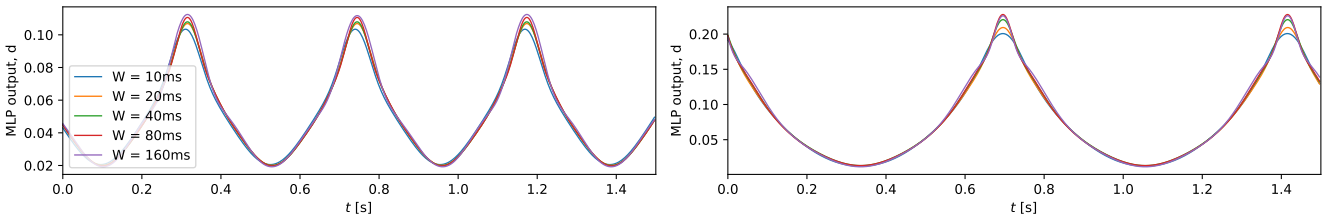


Figure 9: *Outputs of the MLP module in the Small Stone modelling task for different training window lengths with feedback off (SS-A, left) and feedback on (SS-D, right). In both cases, the training audio was recorded with the pedal's rate knob at 3 o'clock. NB the ground-truth signal is unknown so not plotted.*

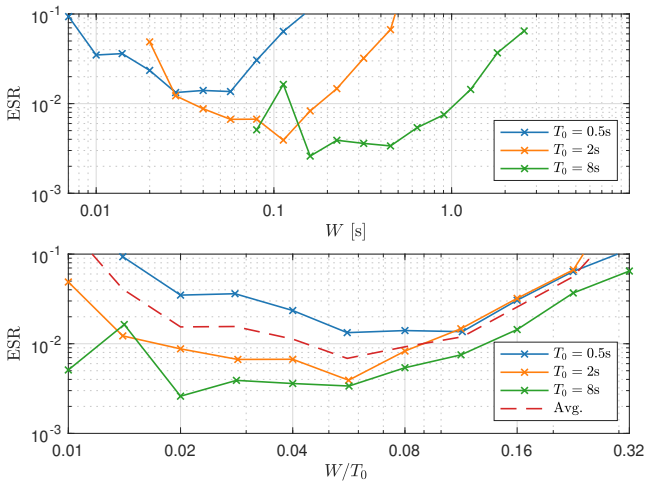


Figure 10: *ESR against training window length W (top) and the ratio of training window length to target LFO period W/T_0 (bottom) for the digital phaser.*

5.3. Experiment 3: inference frame size

The aim of the final experiment was two-fold: to train instances of the model on all six parameter configurations of the Small Stone, and to investigate the effect of window length on model accuracy during inference. For each configuration, the training window length was informed by the results of Experiment 2 and set within 5-10% of the estimated LFO period (see Table 1). The training data was truncated to contain approximately three cycles of the LFO. After training, the models were tested using various window lengths at inference, with the results shown in Figure 11

In both feedback configurations, the models trained on higher LFO rates (SS-A, SS-D) were most sensitive to changes in window size at inference time. This implies a fine balance between the window size being long enough to simulate the transient behaviour of the device, but short enough to not smear the LFO behaviour. In contrast, the accuracy of models trained on longer LFO periods (SS-C, SS-F) was mostly unchanged across the range of window sizes tested. This is a promising result, as it implies one can use a long window size for accelerated training; but a short window size for lower-latency playback at inference. However, the results suggest that there will always be a lower bound on the frame size that is determined by the decay time of the system modelled.

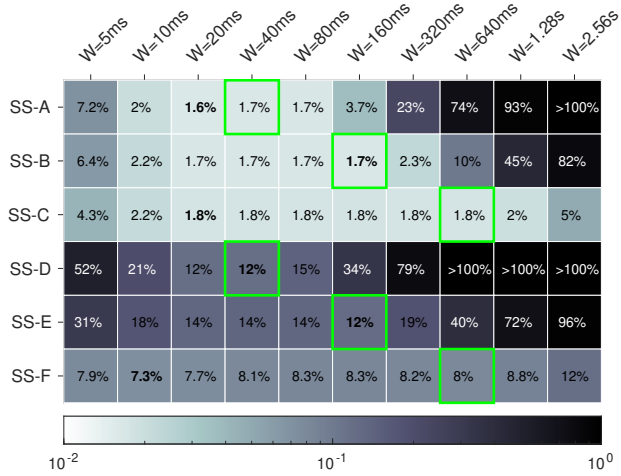


Figure 11: *ESR using various window sizes at inference time across the Small Stone parameter configurations. The green squares indicate the training window lengths.*

Table 2: *Example learned parameters of digital phaser model with $T_0 = 2\,\mathrm{s}$ (DP-2). The parameters are from the best performing model in Experiment 1, obtained using a window size of 80ms during training. Note that the biquad feedforward coefficients have been normalised.*

Parameter	Description	Target value	Initial Value	Learned Value (to 3.s.f)
f_0	LFO rate [Hz]	0.5	0.0627	0.500
g_1	Wet mix	1.0	1.0	0.999
g_2	Feedback gain	0.7	0.01	0.700
ϕ	Feedback delay-line length (samples)	1	0.5	0.995
$[b_{0_1}, b_{1_1}, b_{2_1}]$	Biquad 1 feedforward coeffs.	[1, 0, 0]	[1, 0, 0]	[1.00, -0.0641, 0.0336]
$[a_{1_1}, a_{2_1}]$	Biquad 1 feedback coeffs.	[0, 0]	[0, 0]	[-0.0629, 0.0336]
$[b_{0_2}, b_{1_2}, b_{2_2}]$	Biquad 2 feedforward coeffs.	[1, 0, 0]	[1, 0, 0]	[1.00, -0.0214, 0.0140]
$[a_{1_2}, a_{2_2}]$	Biquad 2 feedback coeffs.	[0, 0]	[0, 0]	[-0.0238, 0.0136]

6. CONCLUSIONS AND FURTHER WORK

This work has presented a differentiable DSP model of a phaser that uses frame-based spectral processing to implement a time-varying filter in the frequency domain. The model was based on a generalised continuous-time model of a phaser effect with several free parameters learnable via gradient descent. It was shown that the model can recover the parameters of a reference digital phaser and learn the correct frequency, starting phase and waveform of the underlying low-frequency oscillator (LFO), without seeing the ground-truth LFO during training. Furthermore, the model was trained to emulate an analog reference device. Informal listening found the model perceptually convincing in this task for a range of parameter configurations. Formal listening tests are important, but left for future work. It was found that the objective model accuracy depended on the training window length and required a manual estimation of the target LFO frequency for the best results. Future work will aim to remove the need for this initial estimation, perhaps through a multi-resolution training process. A key limitation of the proposed model is the inherent latency introduced by the frame-based approach, which could be problematic for real-time model inference. Future work will aim to remove this by implementing an equivalent audio-rate time-domain recursion. Finally, further work may involve extending the general approach proposed in this paper to grey-box modelling of other time-varying filters and delay-based audio effects such as auto-wah, flangers and chorus.

7. ACKNOWLEDGMENTS

The authors would like to thank Ben Hayes, Lauri Juvela and Alec Wright for helpful discussions on some underlying concepts of this work. Thank you to the anonymous reviewers for their comments.

8. REFERENCES

[1] P. Dutilleux, M. Holters, S. Disch, and U. Zölzer, "Filters and delays," in *DAFX: Digital Audio Effects*, U. Zölzer, Ed., chapter 2, pp. 47–81. John Wiley & Sons, Ltd, 2011.

[2] M. Holters and U. Zölzer, "Physical modelling of a wah-wah effect pedal as a case study for application of the Nodal DK Method to circuits with variable parts," in *Proc. 14th Int. Conf. Digital Audio Effects*, Paris, France, Sept. 2011.

[3] F. Eichas, M. Fink, M. Holters, and U. Zölzer, "Physical modeling of the MXR Phase 90 guitar effect pedal," in *Proc. 17th Int. Conf. Digital Audio Effects*, Erlangen, Germany, Sept. 2014.

[4] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *Proc. IEEE Int. Conf. on Acoust. Speech Sig. Proces.*, Brighton, UK, May 2019.

[5] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Appl. Sci.*, vol. 10, no. 2, 2020.

[6] J. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. 22nd Int. Conf. Digital Audio Effects*, Birmingham, UK, Sept. 2019.

[7] F. Esqueda, B. Kuznetsov, and J. Parker, "Differentiable white-box virtual analog modeling," in *Proc. 24th Int. Conf. Digital Audio Effects*, Vienna, Austria, Sept. 2021.

[8] B. Kuznetsov, J. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. 23rd Int. Conf. Digital Audio Effects*, Vienna, Austria, Sept. 2020.

[9] S. Nercessian, A. Sarroff, and K. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *Proc. IEEE Int. Conf. Acoust. Speech Sig. Proces.*, Toronto, Canada, June 2021.

[10] C. Steinmetz and J. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *152nd Audio Eng. Soc. Conv.*, The Hague, Netherlands, May 2022.

[11] M. Comunità, C. Steinmetz, H. Phan, and J. Reiss, "Modelling black-box audio effects with time-varying feature modulation," in *Proc. IEEE Int. Conf. on Acoust. Speech Sig. Proces*, Rhodes, Greece, 2023.

[12] A. Wright and V. Välimäki, "Grey-box modelling of dynamic range compression," in *Proc. 25th Int. Conf. Digital Audio Effects*, Vienna, Austria, Sept. 2022.

[13] M. Martínez-Ramírez, E. Benetos, and J. Reiss, "Deep learning for black-box modeling of audio effects," *Appl. Sci.*, vol. 10, no. 2, 2020.

[14] A. Wright and V. Välimäki, "Neural modeling of phaser and flanging effects," *J. Audio Eng. Soc.*, vol. 69, no. 7, pp. 517–529, 2021.

[15] R. Kiiski, F. Esqueda, and V. Välimäki, "Time-variant gray-box modeling of a phaser pedal," in *Proc. 19th Int. Conf. Digital Audio Effects*, Brno, Czech Republic, Sept. 2016.

[16] A. Carson, "Differentiable Grey-box Modelling of Phaser Effects — accompanying web-page [online]," `https://a-carson.github.io/ddsp-phaser/`, accessed 4/4/23.

[17] B. Bartlett, "A scientific explanation of phasing (flanging)," *J. Audio Eng. Soc.*, vol. 18, no. 6, pp. 674–675, 1970.

[18] J. O. Smith III, *Physical Audio Signal Processing*, `http://ccrma.stanford.edu/~jos/pasp/`, accessed 28/2/23, online book, 2010 edition.

[19] H. Bode, "History of electronic sound modification," *J. Audio Eng. Soc.*, vol. 32, no. 10, pp. 730–739, 1984.

[20] V. Välimäki, S. Bilbao, J. O Smith, J. S Abel, J. Pakarinen, and D. Berners, "Virtual analog effects," in *DAFX: Digital Audio Effects*, U. Zölzer, Ed., chapter 12, pp. 473–522. John Wiley & Sons, Ltd, 2011.

[21] C. Darabundit, R. Wedelich, and P. Bischoff, "Digital grey box bodel of the Uni-Vibe effects pedal," in *Proc. 22nd Int. Conf. Digital Audio Effects (DAFx-19)*, Birmingham, UK, Sept. 2019.

[22] T. Stockham, Jr., "High speed convolution and correlation," in *Proc. Spring Joint Comput. Conf.*, April 1966, p. pp. 229–233.

[23] J. Bonada, X. Serra, X. Amatriain, and A. Loscos, "Spectral processing," in *DAFX: Digital Audio Effects*, U. Zölzer, Ed., chapter 10, pp. 393–445. John Wiley & Sons, Ltd, 2011.

[24] B. Hayes, C. Saitis, and G. Fazekas, "Sinusoidal frequency estimation by gradient descent," in *Proc. IEEE Int. Conf. on Acoust. Speech Sig. Proces*, Rhodes, Greece, 2023.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations*, San Diego, CA, 2015.

[26] J. Sleep, "Small Stone Information [Online]," `http://generalguitargadgets.com/effects-projects/phase-shifters/small-stone-information/`, accessed 26/3/23.

SELF-SUPERVISED DISENTANGLEMENT OF HARMONIC AND RHYTHMIC FEATURES IN MUSIC AUDIO SIGNALS

Yiming Wu

AlphaTheta Corporation
Yokohama, Japan
yiming.wu@alphatheta.com

ABSTRACT

The aim of latent variable disentanglement is to infer the multiple informative latent representations that lie behind a data generation process and is a key factor in controllable data generation. In this paper, we propose a deep neural network-based self-supervised learning method to infer the disentangled rhythmic and harmonic representations behind music audio generation. We train a variational autoencoder that generates an audio mel-spectrogram from two latent features representing the rhythmic and harmonic content. In the training phase, the variational autoencoder is trained to reconstruct the input mel-spectrogram given its pitch-shifted version. At each forward computation in the training phase, a *vector rotation* operation is applied to one of the latent features, assuming that the dimensions of the feature vectors are related to pitch intervals. Therefore, in the trained variational autoencoder, the rotated latent feature represents the pitch-related information of the mel-spectrogram, and the unrotated latent feature represents the pitch-invariant information, *i.e.*, the rhythmic content. The proposed method was evaluated using a predictor-based disentanglement metric on the learned features. Furthermore, we demonstrate its application to the automatic generation of music remixes.

1. INTRODUCTION

Deep neural network (DNN)-based data generation techniques are increasingly used in creative fields. In the audio domain, exciting new methods have been proposed for speech generation, music composition, and sound design. The main advantage of DNNs is their high expressiveness in approximating the real-world data distributions, which can provide consistent generation results that are convincing to human creators. However, because of their highly complicated architecture, the interpretability and controllability of the generative process have become the two main problems with DNN-based data generation. A DNN contains a huge number of stochastically optimized parameters, and hence it is impossible to explain how each parameter or each internal output influences the final output. In addition, a DNN-based generative method often introduces a stochastic process, which improves the diversity of the generation results, but also makes it more difficult for the users to control the output and obtain results that reflect their intentions.

Disentanglement learning is a key approach to solving the problems of interpretability and controllability. Disentanglement learning aims to model the generative process conditioned by multiple *disentangled* latent variables, *i.e.*, a set of independent variables that are sensitive only to certain factors of the observed data. For example, studies in the speech domain focus on the representations of speaker identity, gender, speed of speech, and emotions [1, 2]. Generative models with properly disentangled latent variables make it easier to explicitly reflect the intentions of human users in the generation results.

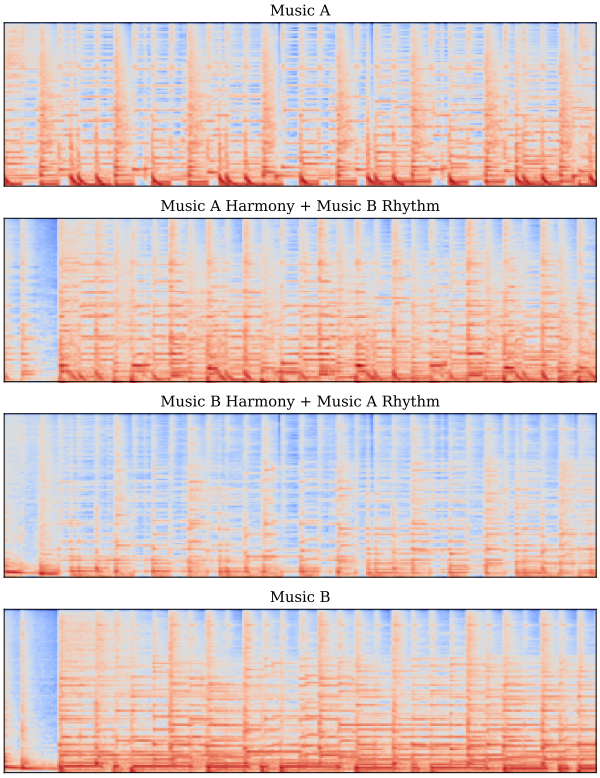


Figure 1: *An example of music remix generation by the proposed harmony-rhythm disentanglement method. The middle two spectrograms were generated by combining the harmony and rhythm contents of different music.*

In this paper, we focus on disentanglement learning for generative models of musical audio. More specifically, our goal is to learn the disentangled latent features of the rhythmic and harmonic content in musical audio. For human listeners, the rhythmic content of a piece of music is derived from the onset timings of the musical audio, and the harmonic content is derived from the different pitches of the musical audio. Therefore, these two types of content are considered to be independent of each other. In the

time-frequency representations (such as spectrograms) of musical audio, the rhythmic and harmonic content can be observed in the temporal progressions along the time and frequency axes, respectively, and this can be used to implement a harmonic-percussive source separation algorithm [3]. We assume that harmonic and rhythmic content can also be separated in latent space.

We propose a simple training method to obtain the disentangled latent features by introducing several constraints during the training process. It involves training a generative model for music audio spectrogram using a variational autoencoder (VAE), in which the encoder network maps the input spectrogram to the latent features while the decoder network maps the latent features back to the audio spectrogram. The key idea behind our approach is to let the VAE not only reconstruct the input spectrogram, but also reverse the transformation applied to the input spectrogram. In the proposed method, the transformation is audio pitch-shifting. We assume that pitch-shifting on the musical audio only changes its harmonic content and not its rhythmic content. By introducing a vector rotation on the harmonic latent feature to reverse the pitch shift operation, the rotated and unrotated latent features can be trained without supervision to represent the pitch-related and pitch-invariant information in musical audio, respectively.

The main contribution of this work is to propose an effective disentanglement learning method that is suitable for DNN-based music audio generation models. In the evaluation section, we show the quality of disentanglement quantitatively using a predictor-based metric. We also explore the application of the proposed method to the automatic generation of music remixes, by replacing the rhythmic (or harmonic) feature of one musical audio clip with that of another musical audio clip. The quantitative evaluations and concrete audio examples demonstrate that the proposed method can generate realistic music remixes that possesses the characteristics of both sources of music.

2. RELATED WORK

This section reviews related work on DNN-based generation and disentanglement learning for musical audio.

2.1. DNN-based Musical Audio Generation

Several different approaches have been proposed for DNN-based music audio generation. One popular approach is based on differential digital signal processing (DDSP) [4], in which the generative model is concatenated with audio DSP modules such as filters and oscillators. DNNs are then trained to estimate the parameters of these DSP modules. Because DDSP-based generative models utilize strong inductive biases, they are generally more interpretable, and require fewer audio examples to achieve reasonable generalized performance. Therefore, DDSP has been applied in several existing synthesizer algorithms, such as wavetable synthesizer [5], waveshaping synthesizer [6], FM synthesizer [7], and the WORLD vocoder [8].

Another approach is the autoencoding approach, which trains a DNN-based generative model and its latent feature space using an autoencoder network. Once the autoencoder has been trained, musical audio can be generated by manipulating the latent feature and reconstructing the audio using the decoder network. More specifically, one can interpolate over the latent feature space like RAVE [9], or train the language model of the latent feature to generate musical audio from scratch, as in Jukebox [10], Musika [11], and MusicLM [12].

2.2. Disentanglement Learning for Audio

The main goal of disentanglement learning for audio is to implement audio transformation systems that change certain aspects of the musical content, such as timbre or musical styles. For example, Noam et al. proposed a music translation method that transforms the domain (musical instruments and styles) of musical audio [13]. The method is based on a multi-domain autoencoder based on WaveNet [14], where the encoder WaveNet transforms the audio waveform into a domain-independent latent representation, and the domain-specific WaveNet decoders reconstruct the audio waveform from the latent representation. To make the encoder extract the domain-independent representation from audio waveforms, the encoder is trained to fool a domain classifier network that tries to correctly recognize the domain type from the latent representation. This approach is not a fully unsupervised method because a domain label should be given for each musical audio clip used to train the neural networks.

Studies on disentanglement learning for audio have proposed several learning schemes to automatically separate the pitch-related and pitch-invariant information in the musical audio in the latent space of an audio generative model. Luo et al. proposed a learning method to encode the pitch and timbre of musical instrument sounds using Gaussian mixture VAE [15] , where the latent representations were learnt in a supervised and semi-supervised manner using pitch and instrument annotations. GANStrument proposed by Narita et al. introduces an adversarial training scheme to extract pitch-invariant features from musical instrument sound [16]. Using the trained feature encoder, GANStrument can generate pitched instrument sounds given a one-shot sound as input. Luo et al. also proposed an unsupervised learning method to encode the pitch and timbre of musical instrument sounds, in which the pitch is represented as a discrete label and the timbre is represented as a continuous feature vector [17] . Similar to our proposed method, they assume that a moderate pitch shift operation does not change the timbre of the original musical instrument sound. Based on this assumption, they treat the original sound and its pitch-shifted version as a pair, and swap the encoded pitch variables before reconstructing the musical sound using the decoder. Because the pitch is represented as a single discrete variable, this method is suitable for monophonic musical sound. Our proposed method formulates a VAE in a similar way; however, we formulate the pitch-related feature as continuous value vectors, so that these vectors can represent the polyphonic pitch information found in any kind of musical recording.

3. PROPOSED METHOD

This section describes the proposed self-supervised disentanglement learning method. An overview of the proposed method is shown in Fig. 2. We formulate a probabilistic generative model representing the generative process of an audio mel-spectrogram from two latent features representing harmony and rhythm in the form of a VAE (Section 3.1). In the training phase, we use an audio pitch-shifting algorithm to enable the model to learn the two latent features that represent the pitch-related and pitch-invariant information of the input audio (Section 3.2). In addition, we train the decoder as a generative adversarial network (GAN) to improve the generation quality (Section 3.3).

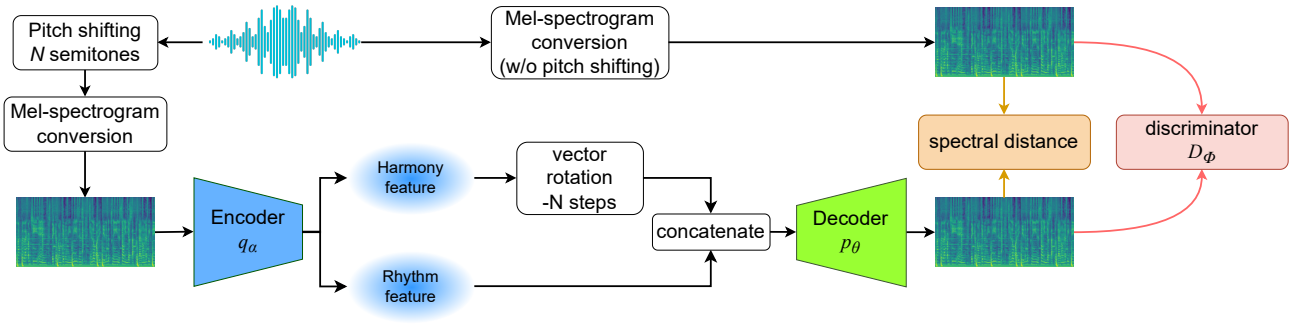


Figure 2: *Proposed VAE architecture and its forward computation procedure.*

3.1. VAE Formulation

Let $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ be a log-scaled mel-spectrogram of a musical audio, represented as a sequence of D-bins spectrum $\mathbf{x}_n \in \mathcal{R}^D$. Let $\mathbf{Z}^h = \{\mathbf{z}_n^h\}_{n=1}^N$ and $\mathbf{Z}^r = \{\mathbf{z}_n^r\}_{n=1}^N$ be sequences of latent features, where $\mathbf{z}_n^h, \mathbf{z}_n^r \in \mathcal{R}^L$ are L-dimensional continuous-valued vectors ($L = 128$) that abstractly represent the harmonic and rhythmic content at the nth audio frame, respectively. We formulate a generative model with $\mathbf{X}$ as the observed variable and $\mathbf{Z}^h, \mathbf{Z}^r$ as the latent features as follows:

$$p(\mathbf{X}) = p_\theta(\mathbf{X}|\mathbf{Z}^h, \mathbf{Z}^r)p(\mathbf{Z}^h)p(\mathbf{Z}^r) \tag{1}$$

where $p_\theta(\mathbf{X}|\mathbf{Z}^h, \mathbf{Z}^r)$ is a conditional generative model with parameters θ. We define p_θ as a decoder neural network parametrized by θ. The decoder network models the generative process of mel-spectrogram from the two latent features $\mathbf{Z}^h$ and $\mathbf{Z}^r$. In our work, we evaluate $p_\theta(\mathbf{X}|\mathbf{Z}^h, \mathbf{Z}^r)$ using the spectral distance between $\mathbf{X}$ and the output of the decoder network $\omega_\theta(\mathbf{Z}^h, \mathbf{Z}^r)$:

$$p_\theta(\mathbf{X}|\mathbf{Z}^h, \mathbf{Z}^r) \sim S_\theta(\mathbf{X}, \mathbf{Z}^h, \mathbf{Z}^r) \stackrel{\text{def}}{=} ||\mathbf{X} - \omega_\theta(\mathbf{Z}^h, \mathbf{Z}^r)||_1 \tag{2}$$

where $||\cdot||_1$ is the *L1* norm.

Since the inference model of the latent features $p(\mathbf{Z}^h, \mathbf{Z}^r|\mathbf{X})$ is intractable, we use a neural encoder network q_α that approximates the distributions of the latent features given an observed mel-spectrogram as follows:

$$q_\alpha(\mathbf{Z}^h|\mathbf{X}) = \prod_{n=1} \mathcal{N}(\mathbf{z}_n^h|\mu_\alpha(\mathbf{X})_n^h, \sigma_\alpha(\mathbf{X})_n^h) \tag{3}$$

$$q_\alpha(\mathbf{Z}^r|\mathbf{X}) = \prod_{n=1} \mathcal{N}(\mathbf{z}_n^r|\mu_\alpha(\mathbf{X})_n^r, \sigma_\alpha(\mathbf{X})_n^r) \tag{4}$$

where $\mu_\alpha(\mathbf{X})^h$, $\sigma_\alpha(\mathbf{X})^h$,$\mu_\alpha(\mathbf{X})^r$, and $\sigma_\alpha(\mathbf{X})^r$ are the four parts of the encoder network output.

The priors $p(\mathbf{Z}^h)$ and $p(\mathbf{Z}^r)$ are set to a standard Gaussian distribution as follows:

$$p(\mathbf{Z}^r) = \prod_{n=1}^N \mathcal{N}(\mathbf{z}_n^h|\mathbf{0}_L, \mathbf{I}_L), \tag{5}$$

$$p(\mathbf{Z}^h) = \prod_{n=1}^N \mathcal{N}(\mathbf{z}_n^r|\mathbf{0}_L, \mathbf{I}_L), \tag{6}$$

As shown in Fig.3, the encoder neural network is composed of stacked residual convolution layers and downsampling layers. Two independent bottleneck modules are appended to the bottom layer of the encoder to compute the parameters of the two latent distributions. Each downsampling layer is implemented with a strided convolution layer that reduces the dimension of the frequency axis of the mel-spectrogram by a factor of four while keeping the dimension of the time axis unchanged. Therefore, the encoder reduces the frequency axis of the input spectrogram by a factor of 64, and outputs the latent features with two dimensions on the frequency axis. Similarly, the decoder neural network is composed of stacked residual convolution layers and upsampling layers that are implemented with strided transposed convolution layers, each of which expands the frequency-axis by a factor of four.

3.2. Self-Supervised Disentanglement Learning

In a normal VAE setting [18], the generative model is trained within the framework of variational inference, which jointly optimizes the encoder and decoder network to maximize the evidence lower bound (ELBO) of the observed data likelihood $p(x)$ as:

$$\begin{aligned}\mathcal{L}_{VAE_{normal}} &= \mathbb{E}_{q_\alpha(\mathbf{Z}^r,\mathbf{Z}^h|\mathbf{X})}[\log p_\theta(\mathbf{X}|\mathbf{Z}^r, \mathbf{Z}^h)] \\ &- \beta\mathcal{D}_{KL}(q_\alpha(\mathbf{Z}^h|\mathbf{X})||p(\mathbf{Z}^h)) - \beta\mathcal{D}_{KL}(q_\alpha(\mathbf{Z}'^r|\mathbf{X})||p(\mathbf{Z}^r))\end{aligned} \tag{7}$$

where $\mathcal{D}_{KL}(q||p)$ is the KL divergence from distribution q to p, and β is a weighting factor that controls the trade-off between the reconstruction accuracy and level of disentanglement within the latent features [19]. The latent variable regularization term in ELBO encourages disentanglement between each dimension of the latent variable[19]. However, without explicit conditioning, there is no guarantee that the latent variables learn to explicitly represent the harmonic (or rhythmic) aspects of the mel-spectrogram.

To distinguish the harmonic and rhythmic content of musical audio, we make the assumption that rhythmic content is invariant to audio pitch shifting, whereas harmonic content is not. Assuming that the musical audio share the same tuning (*e.g.*, tuned to 440Hz), we add a definition of the dimensions of the latent vector z_n^h: the i-th dimension $z_{n_i}^h$ represents the pitch information of a certain pitch height, and the pitch intervals between the pitches corresponding to the i-th and j-th dimension is $j-i$ times of a small pitch interval unit (we use *semitone* in the following statements). In this way, we can relate audio pitch-shifting to a *vector rotation* operation on z_n^h, *i.e.*, when $\mathbf{Z}^h$ is the harmony feature of $\mathbf{X}$, the n-step vector rotation of $\mathbf{Z}^h$ is the harmony feature of the n-semitone pitch shift version of $\mathbf{X}$.

Based on our definition of $\mathbf{Z}^h$, we designed a training procedure to facilitate the harmony-rhythm disentanglement. Con-

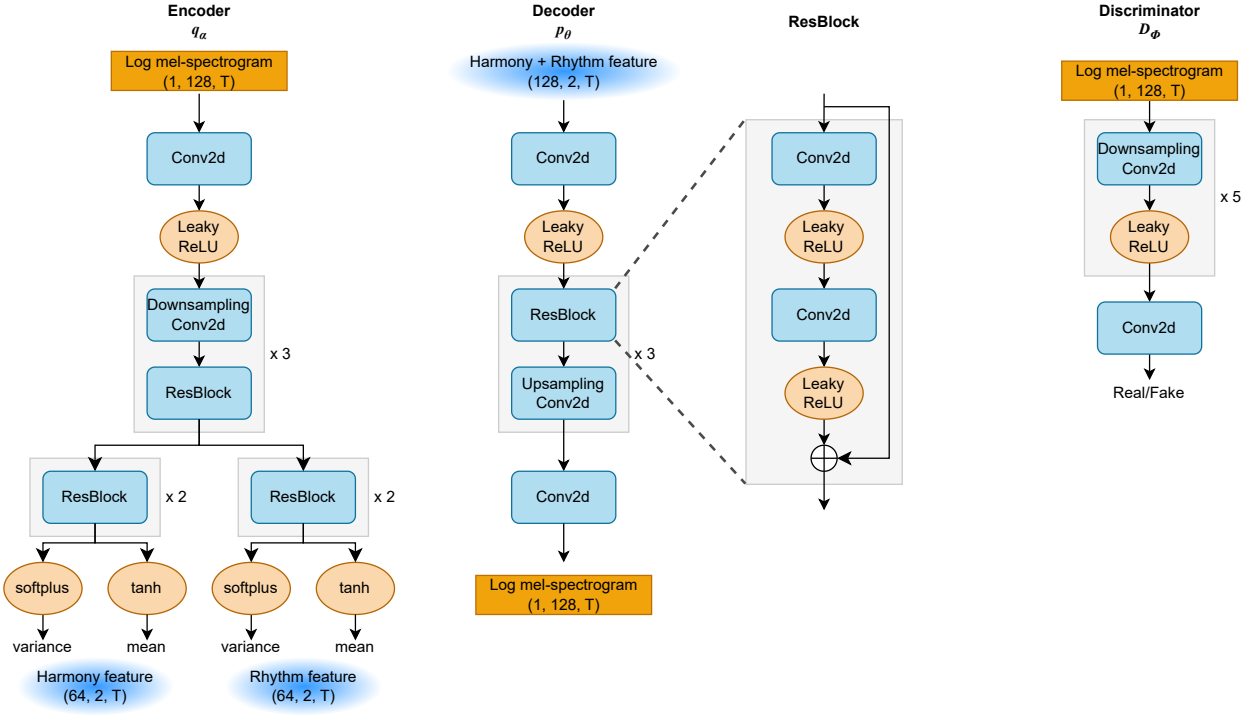


Figure 3: *Proposed VAE architecture and its forward computation procedure.*

cretely, each forward computation in a training iteration proceeds follows:

1. Shift the pitch of the input audio segment by a random number of semitones $n \in [-8, 8]$. Let $\mathbf{X}'$ be the mel-spectrogram of the pitch-shifted audio,
2. Calculate the latent feature distribution $q_\alpha(\mathbf{Z}'^h, \mathbf{Z}'^r|\mathbf{X}')$ using the encoder network,
3. Sample the latent features $\mathbf{Z}'^h, \mathbf{Z}'^r$ from $q_\alpha(\mathbf{Z}'^h, \mathbf{Z}'^r|\mathbf{X}')$ using the reparameterization trick [18],
4. Apply $(-n)$-step vector rotation to the channel dimension of $\mathbf{Z}'^h$. Let $\mathbf{Z}^h$ be the rotated latent feature.
5. Reconstruct the mel-spectrogram from $p_\theta(\mathbf{X}|\mathbf{Z}'^h, \mathbf{Z}^r)$ using the decoder network.

Combining Equation 7 with Equation 2, the training objective of the VAE is:

$$\mathcal{L}_{VAE} = \mathbb{E}_{q_\alpha(\mathbf{Z}'^h, \mathbf{Z}^r|\mathbf{X}')}[S_\theta(\mathbf{X}, \mathbf{Z}'^h, \mathbf{Z}^r)] - \beta\mathcal{D}_{KL}(q_\alpha(\mathbf{Z}'^h|\mathbf{X}')||p(\mathbf{Z}^h)) - \beta\mathcal{D}_{KL}(q_\alpha(\mathbf{Z}'^r|\mathbf{X})||p(\mathbf{Z}^r)) \quad (8)$$

We set $\beta = 0.1$ in our experiment, which places more weight on the reconstruction accuracy. The expectation term is approximated by the Monte Carlo method using the reparameterization trick. Intuitively, the VAE decoder is trained to reconstruct the original mel-spectrogram $\mathbf{X}$ given the latent variables encoded from the pitch-shifted mel-spectrogram $\mathbf{X}'$. Because $\mathbf{Z}'^r$ is not altered during the forward computation, it should represent the pitch-invariant elements in $\mathbf{X}'$ and $\mathbf{X}$. By contrast, because the vector rotation on $\mathbf{Z}'^h$ reverts the pitch shift on $\mathbf{X}$, the rotated variable $\mathbf{Z}^h$ is able to represent the pitch-specific elements of the original $\mathbf{X}$. Therefore, unlike $\mathbf{Z}^r$, $\mathbf{Z}^h$ should represent the pitch-related elements in $\mathbf{X}$ during the optimization.

3.3. GAN Learning

To improve the quality of the generated mel-spectrogram, the VAE networks are also trained as a GAN [20]. We additionally define a discriminator network D_ϕ that learns to distinguish the generated mel-spectrogram from the original mel-spectrogram. The GAN training objective is defined as follows:

$$\mathcal{L}_{dis} = (1 - D_\phi(\mathbf{X}))^2 + D_\phi(\hat{\mathbf{X}})^2 \quad (9)$$

$$\mathcal{L}_{gen} = -D_\phi(\hat{\mathbf{X}})^2 \quad (10)$$

where $\mathbf{X}$ is the original spectrogram and $\hat{\mathbf{X}}$ is the spectrogram reconstructed by the VAE. To stabilize the adversarial training process, a feature matching loss $\mathcal{L}_{FM}$ [21] is further added to the training objective. Altogether, the objective function for the VAE network optimization is

$$\mathcal{L}_{total} = \mathcal{L}_{VAE} + \mathcal{L}_{gen} + \mathcal{L}_{FM}$$

Following the ordinary GAN training procedure, the discriminator network is trained to minimize $\mathcal{L}_{dis}$, and the VAE network is trained to minimize $\mathcal{L}_{total}$. As illustrated in Fig. 3, the discriminator network is composed of five convolutional layers with leaky ReLU activation.

Combination of VAE and GAN objectives is also used to train the RAVE [9] and Musika! [11] audio synthesizer. Unlike RAVE, our method does not optimize the VAE and GAN objectives separately. We also do not fix the parameters of the encoder network.

In our experiments, the objective $\mathcal{L}_{total}$ jointly optimizes the encoder and decoder network.

4. EVALUATION

This section reports the comparative experiment conducted to evaluate the effectiveness of the proposed disentanglement learning method. The experiments were implemented using PyTorch [22], and the source code is available on GitHub. [1]

4.1. Datasets

We use the *fma-large* subset of the Free Music Archive (FMA) dataset [23] to train the VAE. The dataset contains 30-second musical audio snippets from 106,574 Creative Commons-licensed music tracks. To measure the quality of the rhythm–harmony disentanglement of the proposed method, we use the RWC-Popular dataset [24] as the test set. The RWC-Popular dataset contains 100 pieces of popular song audio with chord progression annotations. Following the common automatic chord estimation setting, the annotated chord labels are reduced to the *major* and *minor* triads.

The mel-spectrogram was computed from the audio signal using a sample rate of 22,050Hz. The FFT size, window length, and hop size of the short-time Fourier transform were set to 2048, 2048, and 512 samples, respectively, and the number of mel frequency bins was set to 128 (thus $D = 128$). Hann window was used for FFT computation.

A general-purpose audio pitch-shifting algorithm was used to obtain the pitch-shifted versions of the musical audio. In our experiments, we used the pitch-shifting function implemented in the *Pedalboard* audio processing library, [2] which wraps the *Rubber Band* audio stretching library. [3] The *Rubber Band* audio stretching algorithm is based on the phase-vocoder method that uses phase resets on the percussive transients, an adaptive stretch ratio between phase reset points, and a "lamination" method to improve vertical phase coherence. In contrast to the naive phase-vocoder time stretching algorithm implemented in *librosa* [25] and *torchaudio*, *Rubber Band*'s algorithm can preserve percussive sounds without noticeable distortion.

4.2. Evaluation Metrics

We use a predictor-based evaluation metric similar to that used in [17] to measure the disentanglement between the inferred rhythm and harmony features. Specifically, a sequence classification model based on a two-layer bidirectional gated recurrent unit (GRU) network was trained to predict the chord labels and onset states from the audio features $\mathbf{Z}^h$, $\mathbf{Z}^r$, or the original audio mel-spectrogram $\mathbf{X}$. The accuracy of chord label prediction was measured by the frame-wise label overlap rate, and the accuracy of onset prediction was measured by the binary F-1 score over the onset positions.

The accuracy of chord prediction and onset prediction measures how well the latent features reflect the pitch-related and pitch-invariant information of the audio, respectively. If $\mathbf{Z}^r$ and $\mathbf{Z}^h$ are well disentangled, the classifiers on $\mathbf{Z}^r$ should yield high accuracy for onset prediction and low accuracy for chord label prediction. Similarly, the classifiers on $\mathbf{Z}^h$ should yield high accuracy for chord label prediction and low accuracy for onset prediction.

The RWC-Popular dataset is divided into a training set (90%) and an evaluation set(10%). The data pairs of musical audio and chord label annotations in the RWC-Popular dataset were used to train and evaluate the chord label classifier. Similarly, the musical audio and onset label data pairs were used to train and evaluate the onset label classifier, where the onset label was inferred from the raw music audio using the onset detection algorithm implemented in the *librosa* library.

We further explore the application of the proposed method to the automatic generation of music remixes. To generate music remixes, we used the trained VAE to generate audio spectrograms that simultaneously contain the musical elements of two different music tracks. Given two pieces of beat-synchronized music A and B, a remix was created by the following process:

1. Infer the latent representations $\mathbf{Z}^h_A, \mathbf{Z}^r_A, \mathbf{Z}^h_B$, and $\mathbf{Z}^r_B$ of the mel-spectrograms $\mathbf{X}_A$ and $\mathbf{X}_B$ using the encoder network,
2. Generate the mel-spectrogram from $\mathbf{Z}^h_A, \mathbf{Z}^r_B$ using the decoder network.

We used the Fréchet Inception Distance (FID) [26] to quantitatively measure the quality of the generated spectrograms. The FID measure is given by:

$$F(\mathcal{N}_b, \mathcal{N}_e) = ||\mu_b - \mu_e||^2 + tr(\Sigma_b + \Sigma_e - 2\sqrt{\Sigma_b \Sigma_e}) \quad (11)$$

where $\mathcal{N}_b(\mu_b,\ \Sigma_b)$ is the multivariate normal distribution estimated from the Inception V3 [27] features calculated from a set of spectrograms of the real musical audio, and $\mathcal{N}_e(\mu_e,\ \Sigma_e)$ is the distribution calculated from the generated spectrograms. The generated spectrograms are considered to be more musically realistic if the computed FID is low. The feature extractor is a pre-trained music genre classifier that wass trained using the genre-annotated musical audio in the FMA dataset.

We used the following music remixing methods as the baselines:

- **HPSS**. We apply the harmonic-percussive source separation (HPSS) algorithm [28] in the *librosa* library to music A and music B, and mix the harmonic part of A and percussive part of B to create the remix version. The HPSS algorithm infers the spectral masks for harmonic and percussive parts using median-filtering along the time and frequency axis.
- **ASAP**. We use the *Spectral Morphing* audio effect implemented in the *ASAP* plug-in suite developed by IRCAM. [4] The *Spectral Morphing* plugin combines the spectral characteristics of two audio signals using the source-filter technique where the audio signal of music B is used as a filter of the audio signal of music A. More specifically, the frequency-domain amplitude of the two audio signals are multiplied, while preserving the phase of the source audio. The spectral envelope of music B is further applied to the filtered signal. We set music A as the main input, music B as the sidechain input, and set the *Global Mix* parameter to 100% to generate the remixed version.

We randomly chose 20 songs from the RWC-Popular dataset to create 10 pairs of audio clips. Each audio clip was time-stretched

[1] https://github.com/WuYiming6526/HARD-DAFx2023

[2] https://spotify.github.io/pedalboard/reference/pedalboard.html

[3] https://breakfastquay.com/rubberband/

[4] https://forum.ircam.fr/projects/detail/asap/

Table 1: *Harmony-rhythm disentanglement Metrics*

Feature	chord accuracy	onset F1
harmony feature	**69.61**%	60.09%
rhythm feature	24.65%	**66.04**%
mel-spectrogram	51.95%	65.19%

Table 2: *FIDs of the generated spectrogram*

Model	FID
HPSS	12.84
ASAP	13.18
Disentangled VAE (proposed)	**12.46**

to 120 BPM and was 8s long. Therefore, the FID for each compared method was computed on 10 audio clips generated by the corresponding method. The remixes created by the proposed and the baseline methods can be found on the online project page. [5] Hifi-GAN [29] was used to convert the mel-spectrograms generated by the proposed method into an audio signal.

4.3. Results

Table 1 compares the accuracy of chord classification and onset detection for different audio features. The overall chord classification accuracy for the harmony feature was much higher than for the rhythm feature. The chord labels were almost unpredictable from the rhythm features because these features were trained to be pitch-invariant. By contrast, the beat detection score was higher for the rhythm features than for the harmony features by a much smaller margin. Although the rhythm features were better at representing onset information, the harmony features were not completely onset-invariant. This is somewhat inevitable, since onsets can be inferred in part from pitch transitions. Interestingly, both harmony and rhythm features scored higher than the mel-spectrogram representation in the chord classification and onset detection tasks, respectively. Since the latent features enhance the pitch-related and pitch-invariant elements in the musical audio, it is reasonable that the latent features were found to be more suitable for the pitch-related or rhythm-related music information retrieval tasks. This result indicates that the proposed method can also be used as a self-supervised pre-training method to provide better feature representations for other music information retrieval tasks.

As a qualitative evaluation, we visualized the latent harmony and rhythm representations. Fig. 4 compares the visualized latent features of a song from the RWC-Popular dataset with the ground-truth MIDI pianorolls. It can be seen that the harmony feature had similar pitch progressions to the ground-truth pianoroll. The rhythm features were relatively sparse, and there was no obvious correlation with the pitches or onsets of the ground-truth pianoroll.

As shown in Table 2, the remix generated by the proposed method achieved a better FID score than the baseline methods, suggesting that the proposed method generated spectrograms that are closer to real audio spectrograms than the baseline methods. This is a promising result as it indicates that the proposed method has the potential to generate high-quality remixes. The **HPSS** method simply replaced the percussive part of music A with music B, so the harmonic part does not change. The **ASAP** method

[5] https://wuyiming6526.github.io/HARD-demo/

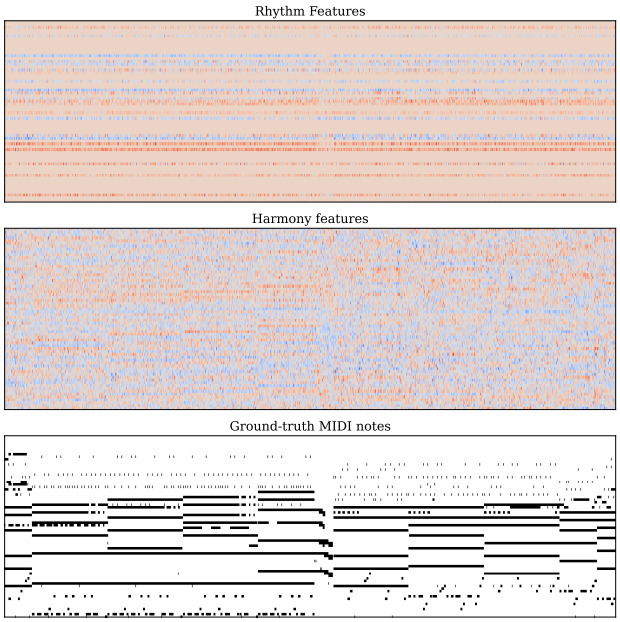


Figure 4: *Visualizations of the rhythm and harmony features of a song from the RWC-Popular dataset. The bottom figure visualizes the MIDI notes from the ground-truth MIDI file.*

added some rhythmic elements of music B to the audio of music A through the dynamic filtering effect, but the rhythmic sounds of music A were still present. In contrast to these baseline methods, the proposed method reflected the rhythmic elements of music B more clearly. Unlike the results of the **HPSS** method, all of the generated audio, including the harmonic part of music A, reflect the rhythm of music B. Unlike the results generated by the **ASAP** method, the rhythm of music A was removed and only the rhythm of music B was present.

5. CONCLUSION

We proposed a simple self-supervised learning method for inferring the disentangled rhythm and harmony features of musical audio. Through quantitative metrics and qualitative observations, we showed that the rhythm and harmony features obtained using the proposed method achieved a high degree of disentanglement. We also demonstrated its potential use for the automatic generation of music remixes.

The generative models that can be used in the proposed method are not limited to spectrogram-based models. In principle, the disentanglement learning strategy can be applied to any kind of autoencoder-based audio generation model, including time domain-based generative models such as RAVE and SoundStream [30]. However, the relationship between the time-domain audio signal and the audio pitch shift is less clear than it is in the time-frequency audio representations. Therefore, disentanglement learning using time domain audio signals may be practically more challenging. In our initial experiments, disentanglement learning on the time-domain generation models did not perform as well as it did with the mel-frequency domain model. The solution to this problem is left for future research.

We also believe that the application of the proposed generative model is not limited to music audio generation. The proposed method could potentially be a pre-training method for downstream music information retrieval tasks. For example, the disentangled acoustic representation of harmony and rhythm may be suitable for musical notes, chords, or beat transcription tasks. Combining the encoder of the proposed VAE with the music transcription model would be worth exploring to push the boundaries of the automatic music transcription research.

6. ACKNOWLEDGMENT

This work has been supported by AlphaTheta Corporation. We thank Kimberly Moravec, PhD, from Edanz (https://jp.edanz.com/ac) for editing a draft of this manuscript .

7. REFERENCES

[1] Wei-Ning Hsu, Yu Zhang, Ron J Weiss, Heiga Zen, Yonghui Wu, Yuxuan Wang, Yuan Cao, Ye Jia, Zhifeng Chen, Jonathan Shen, Patrick Nguyen, and Ruoming Pang, "Hierarchical generative modeling for controllable speech synthesis," in *International Conference on Learning Representations (ICLR)*, 2019, pp. 1–27.

[2] Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ-Skerry Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Ye Jia, Fei Ren, and Rif A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 5180–5189.

[3] Derry FitzGerald, "Harmonic/percussive separation using median filtering," in *Proceedings of the 13th International Conference on Digital Audio Effects (DAFX10)*, 2010, pp. 1–4.

[4] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts, "DDSP: Differentiable digital signal processing," in *International Conference on Learning Representations (ICLR)*, 2020, pp. 1–19.

[5] Siyuan Shan, Lamtharn Hantrakul, Jitong Chen, Matt Avent, and David Trevelyan, "Differentiable wavetable synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 4598–4602, arXiv, version: 2.

[6] Ben Hayes, Charalampos Saitis, and George Fazekas, "Neural waveshaping synthesis," in *Proceedings of the 22rd International Society for Music Information Retrieval Conference (ISMIR)*, 2021, pp. 254–261.

[7] Franco Caspe, Andrew McPherson, and Mark Sandler, "DDX7: Differentiable FM synthesis of musical instrument sounds," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, 2022, pp. 608–616.

[8] Shahan Nercessian, "Differentiable WORLD synthesizer-based neural vocoder with application to end-to-end audio style transfer," *arXiv preprint arXiv:2208.07282*, 2022.

[9] Antoine Caillon and Philippe Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[10] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[11] Marco Pasini and Jan Schlüter, "Musika! fast infinite waveform music generation," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, 2022, pp. 543–550.

[12] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank, "MusicLM: Generating music from text," *arXiv preprint arXiv:2301.11325*, 2023.

[13] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman, "A universal music translation network," in *International Conference on Learning Representations (ICLR)*, 2019, pp. 1–13.

[14] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukchoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[15] Yin-Jyun Luo, Kat Agres, and Dorien Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 746–753.

[16] Gaku Narita, Junichi Shimizu, and Taketo Akama, "GANStrument: Adversarial instrument sound synthesis with pitch-invariant instance conditioning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[17] Yin-Jyun Luo, Kin Wai Cheuk, Tomoyasu Nakano, Masataka Goto, and Dorien Herremans, "Unsupervised disentanglement of pitch and timbre for isolated musical instrument sounds," in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 700–707.

[18] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2014.

[19] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations (ICLR)*, 2017, pp. 1–22.

[20] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, Cambridge, MA, USA, 2014, NIPS'14, p. 2672–2680.

[21] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 1–12.

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in PyTorch," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[23] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson, "FMA: A dataset for music analysis," *arXiv preprint arXiv:1612.01840*, 2017.

[24] Masataka Goto, "RWC music database: Popular, classical, and jazz music databases," in *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, 2002, pp. 287–288.

[25] Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and music signal analysis in python," in *Python in Science Conference*, 2015, pp. 18–24.

[26] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 6629–6640.

[27] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[28] Jonathan Driedger, Meinard Müller, and Sascha Disch, "Extending harmonic-percussive separation of audio signals," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 611–616.

[29] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 17022–17033.

[30] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi, "SoundStream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2022.

DIFFERENTIABLE ALL-PASS FILTERS FOR PHASE RESPONSE ESTIMATION AND AUTOMATIC SIGNAL ALIGNMENT

*Anders R. Bargum**

Multi Sensory Experience Lab
Aalborg University
Copenhagen, Denmark
arba@create.aau.dk

Stefania Serafin

Multi Sensory Experience Lab
Aalborg University
Copenhagen, Denmark
sts@create.aau.dk

Cumhur Erkut

Multi Sensory Experience Lab
Aalborg University
Copenhagen, Denmark
cer@create.aau.dk

*Julian D. Parker**

firstname.lastname@cantab.net

ABSTRACT

Virtual analog (VA) audio effects are increasingly based on neural networks and deep learning frameworks. Due to the underlying black-box methodology, a successful model will learn to approximate the data it is presented, including potential errors such as latency and audio dropouts as well as non-linear characteristics and frequency-dependent phase shifts produced by the hardware. The latter is of particular interest as the learned phase-response might cause unwanted audible artifacts when the effect is used for creative processing techniques such as dry-wet mixing or parallel compression. To overcome these artifacts we propose differentiable signal processing tools and deep optimization structures for automatically tuning all-pass filters to predict the phase response of different VA simulations, and align processed signals that are out of phase. The approaches are assessed using objective metrics while listening tests evaluate their ability to enhance the quality of parallel path processing techniques. Ultimately, an over-parameterized, BiasNet-based, all-pass model is proposed for the optimization problem under consideration, resulting in models that can estimate all-pass filter coefficients to align a dry signal with its affected, wet, equivalent.

1. INTRODUCTION

Digital simulations of analog audio equipment like tape machines, pre-amplifiers and distortion pedals remain in demand due to the hardware's rich history and unique sonic characteristics. With the increase in computational power, the deep learning approach to machine learning has proven useful for simulating virtual analog (VA) black-box models and has in several publications been applied as the main technique for approximating the output response of analog audio systems [1–5]. In [2] and [3] a WaveNet-based model is as an example adapted to predict the current non-linear output sample value, given a certain number of past input samples and the current input. In both works, the number of past input samples, also called the receptive field, is dynamically selected based on the measured impulse response length of the circuits under consideration. In [4] a recurrent neural network (RNN), such as the gated recurrent unit (GRU), is proposed for simulating the non-linear behaviour of distortion circuits due to their stateful nature. Contrary to this, the authors of [5] present the state trajectory network (STN), comprised of a standard multilayer perceptron (MLP) with a skip-layer connection surpassing the densely connected layers of the network. The STN differs from related work as the input data is concatenated with measured values from the states of the circuit in order to model its behaviour. Since all aforementioned models are built upon the black-box paradigm, the results are significantly exposed to errors in the data collection process and any flaws in the hardware. Thus both the sonic characteristics and the phase response of the system are learned, introducing arbitrary and non-linear phase shifts to the incoming signal. This becomes a problem where parallel-path processing is desired, for instance when dry-wet mixing with the given simulations. All-pass filters (APF) that have unitary magnitude response and frequency-dependent phase responses would traditionally be the approach to take account of the phase shifts, however, manual coefficient adjustments would be both time-consuming and for specific problems, impossible. An automatic solution to the problem, therefore, is highly desired. With inspiration from the differentiable digital signal processing (DDSP) methodology [6], we propose a model that tunes the coefficients of a cascaded APF system. The phase response of different black-box effects is thus automatically approximated, and the adjusted APFs are used to align a dry input signal with the processed, phase-shifted output.

The remainder of this paper is structured as follows: the all-pass optimization problem and related work are introduced in section 2. The construction of the differentiable APFs and their formulas are reviewed in section 3. Our approach and different deep optimization architectures are discussed in section 4. Finally, network evaluations, results, allusions and conclusions are presented in sections 5 and 6.

**Collaboration done while interning/employed at Native Instruments*

2. BACKGROUND

DDSP stems from the motivation of generating audio by using a deep learning workflow to predict and extract synthesis parameters for vocoders and subtractive synthesis [6]. However, directly integrating classic signal processing elements into deep learning methods has shown promising results for the control and adjustment of other DSP blocks, including convolution, filters and one-period wave-tables [7]. Specifically, the authors of [8] have demonstrated the use of DDSP in the context of IIR filters, training different filter topologies in a recursive manner to match target frequency responses. Several other projects have investigated deep learning for IIR filter design, but similar to [8], the work has solely been focused on learning coefficients for magnitude rather than the phase responses. In [9], a neural network is applied to carry out parametric equalizer matching using differentiable biquads, whereas they in [10] approximate shelving filter coefficients directly in the difference equation. All of these works carry out an optimization problem in the frequency domain by minimizing the mean squared error (MSE) between the ground truth and the derived magnitude responses.

The approach to frequency response matching is different in [11]. Here a BiasNet is applied to determine the IIR equalizer parameters. The BiasNet is a simple feedforward neural network that takes advantage of the learnable bias terms, denoted $\mathbf{b}_0$, in the input layer. This architecture is called a "deep optimization" algorithm, owing the name to the use of the neural network as a non-convex optimization algorithm used to tune or derive external parameters. An advantage of the BiasNet is its independence of input features, which according to [11] is more likely to provide a solution to many optimization problems. Furthermore, the network does not rely on the input size and content, hence only a target frequency response is required to be given to the loss function. Similar to the IIR system in [11], adjusting cascaded APFs might be a highly non-linear process. In this paper we, therefore, utilise the over-parameterised nature of the BiasNet to overcome the, potentially, non-convex phase response matching problem and extend the work of [8] to be applicable in the domain of APFs and phase response approximation.

2.1. Problem Formulation

We represent the monophonic signals we want to phase compensate as input vectors $\mathbf{x}^T \in \mathbb{R}$, where T is the signal length. The task is to process these signals with an APF function f, such that the signal is phase shifted to match a target signal introducing the least amount of destructive interference. The function f takes as arguments the input and the number of filter coefficients $\mathbf{c}$ matching the filter order N of the given sub-system. This yields the output $y^T = f(\mathbf{c}^n, \mathbf{x}^T)$. For a system of cascaded APFs, we define the function composition of size D, where each function receives the output of the previous one as:

$$y^T = f(\mathbf{c}^n, \mathbf{x}^T)_1 \circ f(\mathbf{c}^n)_2 \circ ... \circ f(\mathbf{c}^n)_{D-1} \circ f(\mathbf{c}^n)_D, \quad (1)$$

where the order $N = nD$, if each sub-system is a 2nd order filter. The system can be more general than that depending on the value of n. Depending on the deep learning techniques used, each function f can be arbitrarily complex and represented either directly as filter coefficients, as done in [8], or as parameterised sub-networks such as the BiasNet applied for the deep filter optimization procedure in this paper.

3. DIFFERENTIABLE ALL-PASS FILTERS

Before outlining the model architecture of the proposed APF filter tuning process, we present the differentiable APF structures used to adjust the coefficients in the deep learning pipeline. Following the transposed direct form-II (TDF-II) structure, a 2nd order IIR APF is given by the traditional biquad transfer function [12]:

$$A_2(z) = \frac{c + dz^{-1} + z^{-2}}{1 + dz^{-1} + cz^{-2}} \quad (2)$$

In practice, this transfer function can be implemented using the following recurrent, and stateful, difference equation:

$$\begin{aligned} y[n] &= cx[n] + v_1[n] \\ v_1[n] &= dx[n] + v_2[n] - dy[n] \\ v_2[n] &= x[n] - cy[n], \end{aligned} \quad (3)$$

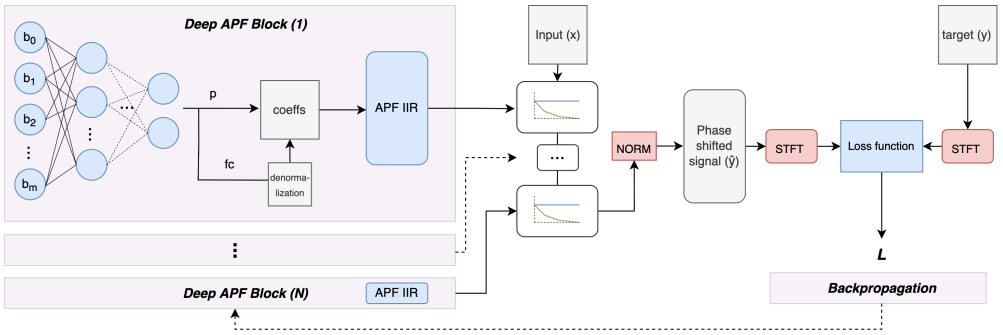
where coefficients c and d are controlling the steepness and the break frequency of the APF's phase response respectively. The coefficients are products of the pole radius R and the cutoff frequency f_c. They have the ranges: $-1 < c < 1$ and $-2 < d < 2$. We introduce a stability constraint to the tuning process and estimate the filter parameters rather than the coefficients themselves. The filter coefficients can for each forward call thereafter be calculated by [13]:

$$c = R^2 \qquad d = -2R\cos(2\pi f_c/f_s), \quad (4)$$

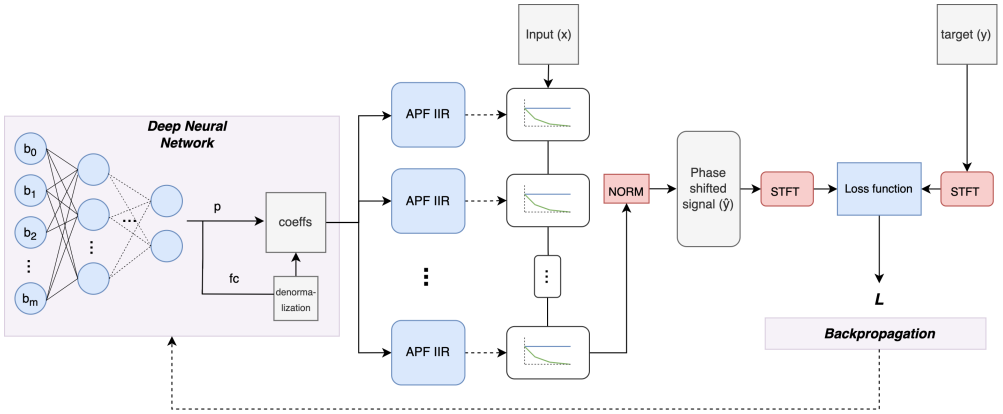
with f_s being the sampling rate of the signal. As the filter coefficients, and thus the steepness of the phase response, are dependent on the pole radius, the phase response might have a significantly narrow resolution at low frequencies. When trying to match frequencies below 100 Hz, the parameters for a system with a high sampling rate (192 kHz) exist in very small ranges with $0.9 < R < 0.999$ and the resulting coefficient d being between $-1.97 < d < -1.999$, depending on the cutoff frequency. It is hypothesised that the prediction of values in such small ranges might introduce numerical overflow and coefficient quantization errors while being difficult to generalise. We, therefore, propose a differentiable warped all-pass structure to increase the frequency resolution in low-frequency ranges, emphasising the importance of low-frequency content in the learning process. A warped APF is designed and realized on a warped frequency scale. It is achieved by replacing the unit delays of a traditional APF with auxiliary 1st order APFs, whose phase response is used to skew the frequency axis [14]. A warped version of the APF in equation (3) is given by the difference equation:

$$\begin{aligned} y[n] &= \frac{x[n](c + a^2 + ad) + v_1[n]}{1 + a^2c + ad} \\ v_1[n] &= x[n](2a + d + ac) + y[n]\big(- 2ac - d - a\big) \\ &\quad -a^3(x[n] - cy[n]) - v_2[n](a^2 + 1) \\ v_2[n] &= x[n] - cy[n] - \big(a^2(x[n] - cy[n]) + av_2[n]\big), \end{aligned} \quad (5)$$

with a being the warping factor i.e. the coefficient of the inserted auxiliary 1st-order APFs. For stability reasons the warping factor for all inserted APFs is identical and thus gathered into a global, but learnable, variable a.



(a) *Sequential BiasNet Structure*



(b) *Fully Connected BiasNet Structure*

Figure 1: *High level overview of the deep-optimization models*

4. PROPOSED METHOD

By utilizing over-parameterisation we propose two BiasNet-based models and a phase alignment procedure to extend the differentiable IIR filter design techniques towards the APFs. We call these models *sequential* and *connected*, as illustrated in figure 1. Both models contain cascaded differentiable warped APFs matching a desired filter order N. The neural network is excited by a learnable bias input layer, whereas its output corresponds to the filter parameters of the closed-form equations used to calculate the final APF coefficients. Three values, R, fc and a are thus fed from the output of the model to every single filter. The primary deep neural network is an MLP with periodic sinusoidal activations for the hidden layers and *tanh* activations for the output layer. The *sine* activation function has been included as it avoids local minima during network optimization, is robust towards vanishing gradients and thus suitable for non-convex problems such as the cascaded APF pipeline [15]. We additionally de-normalise the network outputs taking account of the range in which fc exist. We use a constrained de-normalization technique similar to the one proposed in [11] to de-normalise the *tanh* output layers scaling it between 20 Hz and 20 kHz:

$$fc = \frac{fc_{max} - fc_{min}}{2} p + \frac{fc_{max} + fc_{min}}{2}, \qquad (6)$$

where p denotes the value to de-normalise. The DNN is updated such that its output layer produces filter coefficients that create the needed phase alignment.

We create two different BiasNet models to investigate the importance of over-parameterisation and its impact on the non-convex problem as well as the general learning process. More specifically, the cascade in the *sequential* structure is achieved by chaining several BiasNets together, each representing a respective filter, to create the desired order. Individual DNNs are thus used to derive the coefficients in parallel for each individual APF. The BiasNet is initialized as a densely connected bottleneck with hidden layers of 1024, 512, 256, 128 units respectively. It contains approximately 692.5k learnable parameters, which accumulate to 2.7 million parameters for a cascaded filter of 7th order. Due to its large number of learnable parameters, a benefit of this architecture is the possibility of a complex and detailed parameter estimation process, however, it suffers from longer calculation times and a lack of interaction between the DNNs of each individual block. For the *connected* structure, all filter parameters are coming directly from one large BiasNet. This introduces only 692k learnable parameters in total, independent of the cascaded filter order. The size of the output layer in the *connected* architecture thus equals 11 for a warped APF of 7th order. The *connected* architecture allows for interaction between the cascaded filters since the same network derives all parameters, however, it might suffer from a smaller and therefore less complex parameter space.

4.1. Loss Function

Following the majority of related work, the loss function of the network that is optimized during training happens in the frequency domain. Since the frequency response of an APF by nature is unity gain, a change in magnitude will not be detected and the direct spectrogram comparison used in [6] and [9] is thus not sufficient for the problem at hand. Rather, we calculate the difference between the sum of the target frequency response y and the predicted signal's frequency response $\hat{y}$ individually, as well as the frequency response of the summed signals before the transform. The loss function is given by:

$$\varepsilon_{STFT}(y_i, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}((S(y_i) + S(\hat{y}_i)) - S(y_i + \hat{y}_i))^2, \quad (7)$$

where the function S denotes the spectrogram or the squared magnitude of the STFT, simply given as:

$$S(y_i) = |STFT(y_i)|^2 \quad (8)$$

By doing this, the magnitudes of the target and the prediction are forced to be similar, leaving phase as the only changeable factor. Since the magnitude spectrogram S does not include phase information, it highlights frequency areas where the summation of the input and the target introduce destructive interference. Optimization can thus exclusively be achieved by attaining coefficients whose phase response shifts the input such that the magnitude of the signal summation matches the magnitude summation of each individual STFT. To avoid the frequency-dependent tradeoff of the STFT and to improve the robustness of the loss function, we extend equation (7) by the multi-resolution STFT (M-STFT) loss [16]:

$$\varepsilon_{M-STFT}(y_i, \hat{y}_i) = \frac{1}{M}\sum_{m=1}^{M}\varepsilon_{STFT}(y, \hat{y}), \quad (9)$$

with M being different analysis resolutions. Thus the final loss-function is given by an average over the normal STFT loss in eq (7) at different resolutions. By utilizing multiple FFT-lengths and summing the information across the different resolutions, we capture a more realistic representation of the training signals [16]. The different resolutions are selected according to the STFT parameters presented in [17]:

Table 1: *Details of the parameters for the different STFT resolutions*

FFT-Size	Hop Length	Window Size
512	50	240
1024	120	600
2048	240	1200

4.2. Proof of concept

By a simple proof of concept we show that the over-parameterisation proposed above is crucial for the deep APF optimization problems at hand. To inspect the possibilities of differentiable APFs we first create an example following the work in [8]. We thus start with a naive DDSP approach and derive the coefficient values for the filters directly from the difference equation in order to provide a baseline. To do this we attempt to align the input and output of a simulated 1st-order RC filter, which due to its natural low-pass behaviour creates a phase shift in the higher frequency register. A 1st order RC filter is given by the difference equation [18]:

$$V_{out}^n = \frac{\rho(V_{in}^n + V_{in}^{n-1}) + (1-\rho)V_{out}^{n-1}}{1+\rho}, \quad (10)$$

where V_{in}^n and V_{in}^{n-1} are the current and past input samples, V_{out}^n and V_{out}^{n-1} are the current and past output samples and ρ is given by $fs/(2RC)$, with R and C being the resistance and capacitance of the circuit components. In our case $R = 120\Omega$ and $C = 68nF$ respectively. As the RC filter at maximum will shift incoming frequencies 90°, we train a 1st-order APF the naive way, using the same hyperparameters and loss functions presented in section 5. The results of the trainings are depicted in figure 2.

As seen above, both the M-STFT and the MSE loss converge, with the latter being faster but more noisy. Both cases additionally manage to compensate the phase shifts introduced by the RC filter, with the M-STFT training being more precise. However, when applying the above naive approach to more complex problems such as the VA black box effects presented in section 5, it was quickly realized that the training loss for a system of cascaded APFs diverged and in many cases exploded. When tuning cascaded APFs we are simply handling a highly non-linear problem where the individual minimum of each APF affects the remaining cascade, while the minimum of the loss function most often is based on the frequency areas where alignment gives less destructive interference. The function that can estimate the full phase response thus might be non-convex as it has multiple local minima, which was found to be too complex for the naive and traditional DDSP approach. We argue that over-parameterisation networks and deep optimization frameworks solve this problem.

5. EVALUATION

We examine the proposed models through objective metrics and use the proposals with the best results for final listening tests. The training data consists of a logarithmic sine sweep from 20 Hz to 20 kHz over 10 seconds at a sample rate of 192 kHz. The sweep is fed through three different VA black box simulations shown to introduce significant and complex deviations from linear phase behaviour: 1) Electronic Audio Experiments Surveyor Pre-amp, 2) 15IPS Tape Saturation, and 3) LEM 808R DLX Mixer. We train and evaluate all models in an *in-to-out* fashion, meaning that our models learn the coefficients needed to shift the non-affected input in order to match the VA processed and phase-shifted output. Once training is done, the coefficient values can be exported and inserted into a traditional APF pipeline for the desired real-time adjustments. All models are initialized as a 7th-order APF structure with a cascade of three 2nd-order filters and one 1st-order APF. The output signals of the three systems are sampled and divided into sequences of 2048 samples, which for 20 Hz approximates to a 1/4 of a sinusoidal period at a sample rate of 192 kHz. We heuristically found this sequence length to be a good compromise between phase information and training time. The sub-sequences are additionally organized into batches. We train the models using the earlier mentioned M-STFT loss as well as the traditional MSE loss function, which is used to further validate the phase-compensated simulations/reconstructions. All training sessions are carried out

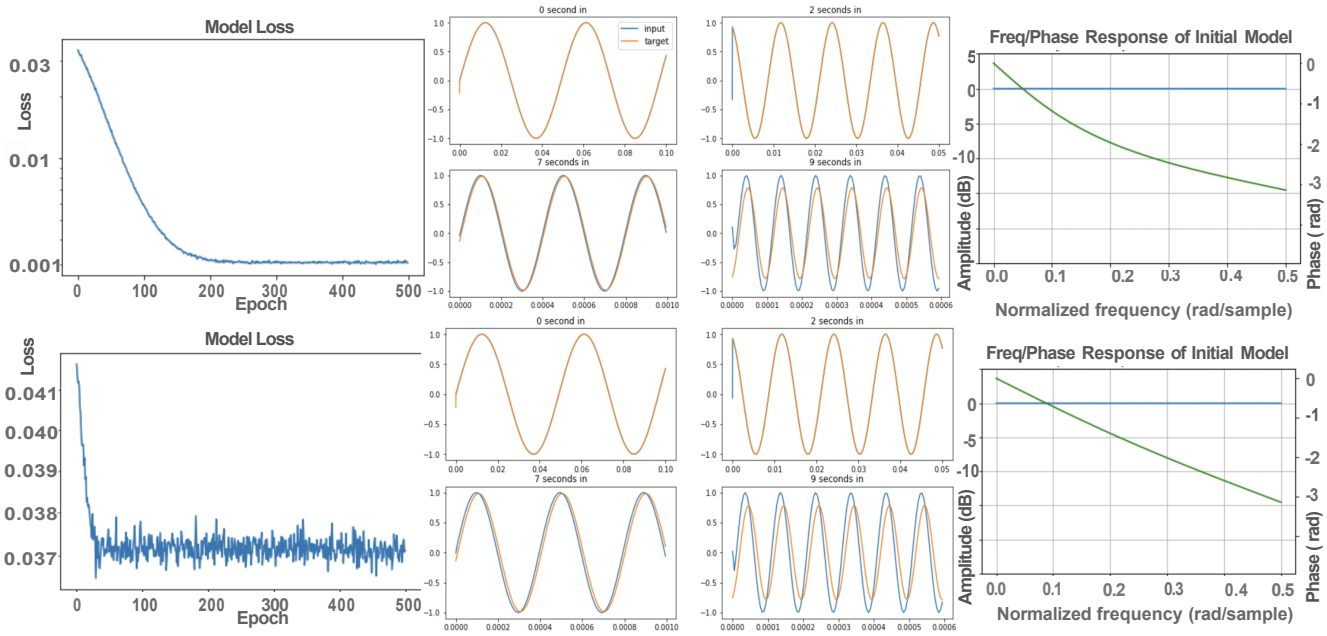


Figure 2: *Loss curve, phase compensation and phase response of RC Filter trained with M-STFT Loss (upper row) and the MSE Loss (lower row). The middle plot shows the alignment at 0, 2, 7 and 9 seconds into the training signal.*

using 1 NVIDIA Tesla T4 GPU for 400 epochs or until training loss plateaus (approx. 5 hours). We train the models with a learning rate of 1e-5, a batch size of 512 and the ADAM optimizer. The final M-STFT and MSE values for the trained models are seen in table 2 below:

Table 2: *Model and training specifications*

Model	Loss Type	Params	Effect	Final Loss
Sequential	MSE	2.7M	Surveyor	1.375e-1
			15 IPS	4.235e-3
			LEM	4.708e-2
Sequential	M-STFT	2.7M	Surveyor	1.857e-2
			15 IPS	8.078e-3
			LEM	9.998e-3
Connected	MSE	692.5K	Surveyor	1.437e-1
			15 IPS	4.271e-3
			LEM	4.707e-2
Connected	M-STFT	692.5K	Surveyor	2.991e-1
			15 IPS	4.871e-1
			LEM	9.887e-3

5.1. Performance Assessment

The performance of the trained models is quantitatively evaluated on unseen test audio. The test audio is chosen such that it exposes the model to signals of various frequencies and timbres. It consists of a concatenation of different loops counting: an acoustic breakbeat drum loop, an electric bass-guitar loop, a guitar loop and a synthesized acid bassline (duration of approx 2 minutes). As signal displacement is given in the time domain, the objective metrics chosen for this study compare the similarity between the automatically shifted input (prediction) and the VA processed output (ground truth) in the sample/phase space. We evaluate the performance by measuring the similarity using the traditional MSE as well as the mean absolute error (MAE) defined as:

$$\epsilon_{MAE}(\hat{y}, y) = \frac{1}{n-1}\sum_{i=0}^{n-1}|\hat{y}_i - y_i| \tag{11}$$

Additionally, we include the error-to-signal ratio (ESR), which can be regarded as an extension of the MSE with the inclusion of target energy normalization to penalise the errors more equally when the input signal is lower in absolute amplitude. The ESR is given by:

$$\epsilon_{ESR}(\hat{y}, y) = \frac{\sum_{i=0}^{N-1}|\hat{y}_i - y_i|^2}{\sum_{i=0}^{N-1}|y_i|^2} \tag{12}$$

The final values for each objective metric are summarized in table 3, with the values for the non-shifted signals included as a static reference. We see that the *sequential* architecture performs best for all objective metrics on both the *surveyor* and the *15IPS* effects (given in bold). The *sequential* model is slightly surpassed by the *connected* architecture in the case of the *LEM* effect, however, only with a combined distance of 0.003 for the MSE trained version and 0.001 for the M-STFT trained version. It can thus be concluded that the *sequential* and over-parameterised BiasNet approach quantitatively provides a closer match to the ground truth phase response of the trained systems. Figure 3 presents a few phase-matching results on different frequency content of the test audio. Examples of all trained VA simulations are shown. It is here clearly seen that the phase-response estimation done by the models compensates for the input to match the saturated and phase-shifted output.

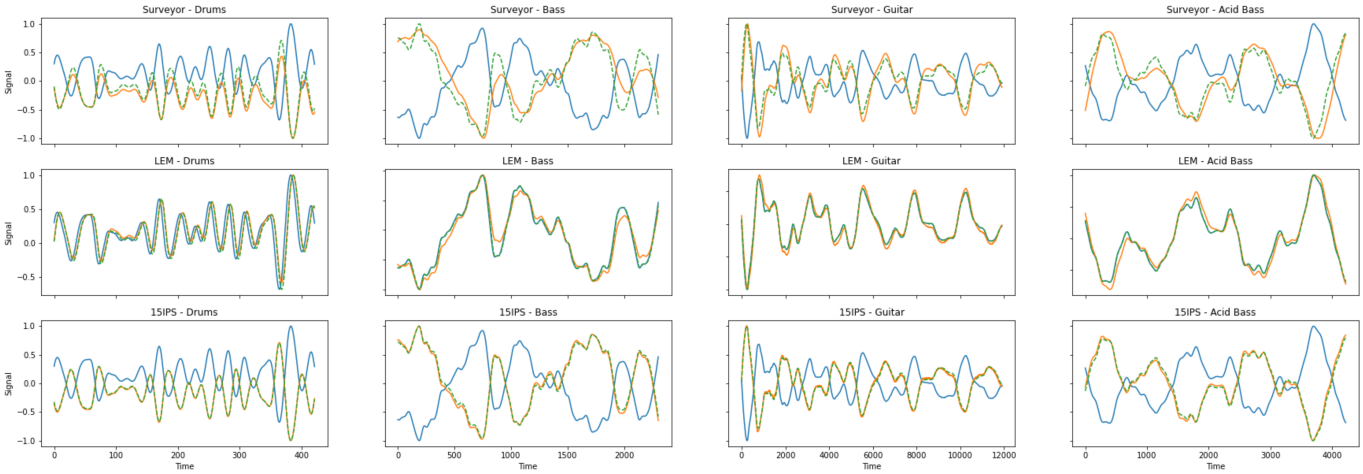


Figure 3: *Examples of the normalized phase alignment results for each individual black-box effect training (blue = input, orange = output, green = prediction*

Table 3: *Overview of the performance results for the individual models across effects and loss functions*

Model	Loss	Effect	MAE	MSE	ESR
Reference	None	Surveyor	0.161	0.066	3.942
		15 IPS	0.164	0.069	4.105
		LEM	0.020	0.001	0.068
Sequential	MSE	Surveyor	***0.033***	***0.003***	***0.177***
		15 IPS	***0.007***	***0.0005***	***0.007***
		LEM	0.017	0.0007	0.045
Sequential	MSTFT	Surveyor	0.037	0.004	0.235
		15 IPS	0.015	0.001	0.058
		LEM	0.017	0.0007	0.045
Connected	MSE	Surveyor	0.079	0.017	1.022
		15 IPS	0.010	0.0002	0.017
		LEM	***0.016***	***0.0007***	***0.042***
Connected	MSTFT	Surveyor	0.074	0.015	0.888
		15 IPS	0.089	0.021	1.236
		LEM	0.017	0.0007	0.044

5.2. Listening Test

Due to the inadequacy of the objective metrics in evaluating the perceived quality of the phase alignment in real life use-cases, such as parallel path processing scenarios, a subjective listening test is carried out. By the use of an 'audio perceptual evaluation' (APE) listening test, we examine the difference between the clean summation and the compensated dry-wet mixing of musical content, using the proposed *sequential* architecture. The APE style test extends the well-known MUSHRA test by rating different versions of the same reference on a single scale using sliders [19]. Compared to the MUSHRA test, the APE is useful for evaluating the perceived quality of dry-wet mixing as there exists no known reference. Since the audibility of dry-wet mixing highly differs relative to the use case, the participants are presented with three different musical scenarios for each compensated audio effect: a low relative mix with 75% dry signal and 25% wet signal, a middle relative mix with 50% dry signal and 50% wet signal, and a high relative mix with 25% dry signal and 75% wet signal. Each relative mix is normalized, however, no loudness compensation has been applied as volume differences in different frequency areas are natural artifacts of phase misalignment and thus represents the baseline of the listening test. We present the participants with two different audio mixes matching a real-world music mixing and mastering scenario, where the black-box effect would be applied to give the final mix a saturating "warmth". The participants are informed that they are listening to different versions of effect models and thereafter instructed to 'blindly' compare the clean and compensated versions based on their perceived level of audio quality. Sound examples can be heard on the accompanying webpage [1]. We additionally provide source code for the trained models [2].

15 convenience sampled participants without any reported hearing impairments and 3 or more years of musical experience took part in an online listening test. Individual boxplots for the evaluation of the clean and compensated audio mixes are shown in figure 4. The answers for each audio mix are summed and averaged for each participant, giving a final comparable score for the individual black-box effects across the different mix configurations.

As seen in 4, the difference between the clean and compensated versions for the 'middle' scenario with 50% dry-wet mixing is highly audible. This is evident both for the *Surveyor* and the *15IPS* effects. The 'low' mix scenario additionally performs better for the compensated version for both the surveyor and the 15IPS. In the case of the LEM effect, all dry/wet mix cases were rated to sound equally good. As seen in figure 3 this is most likely caused by the lack of phase shifts happening in the audible frequency ranges. Lastly, the scores for all the 'high' cases barely differed, which possibly is due to the fact that the dynamics of the saturated output masks the actual interference.

It is thus clear that the trained models manage to align the input

[1] https://abargum.github.io/

[2] https://github.com/abargum/diff-apf

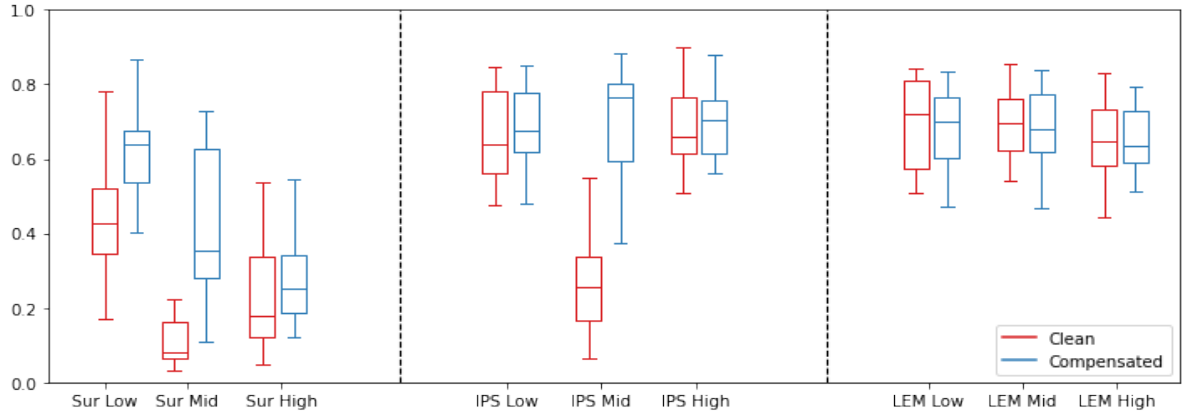


Figure 4: *Ratings across each individual mix-case*

to its respective target signal in the presented examples. This is quantitatively evident in the objective metrics in table 3 where the 'sequential' MSE model perform better on all metrics, compared to its static counterpart. The time-domain representation in figure 3, furthermore, supports the alignment of the musical signals where it clearly can be seen that the temporal envelope of the prediction matches the target. Lastly, a perceptual listening test show that especially the audio quality of the surveyor and the 15IPS models are improved in the dry-wet mixes provided.

6. CONCLUSIONS

To address the challenges of the learned phase responses in VA black box effects, this paper has presented, discussed and evaluated deep-learning techniques for automatic signal alignment. By utilizing the 'deep optimization' methodology, we propose a BiasNet-inspired architecture that approximates filter parameters used for coefficient calculations in a system of cascaded differentiable warped APFs. We thus extend the naive approach to approximating DDSP IIR filters with over-parameterized neural networks and use them to exhibit successful models for aligning the dry and wet paths of virtual analog effects. Ultimately, three black-box effects are chosen for the final training procedure. By evaluating the models on different objective metrics, we demonstrate that what we call a 'sequential' architecture efficiently tunes all-pass filter coefficients for approximating a system's phase response. It is thus demonstrated that over-parameterisation is suitable when estimating filter coefficients in more complex and non-convex scenarios. The results are supported by subjective listening tests, where 15 expert listeners rated the dry-wet mixing of VA effects to be significantly improved by the deep all-pass models, proving that the approach additionally is useful in real life use-cases.

7. ACKNOWLEDGMENTS

Many thanks to the great number of anonymous reviewers and the whole ML/DSP research team at Native Instruments Berlin in 2022 for their help, guidance and support.

8. REFERENCES

[1] J. Chowdhury, "A comparison of virtual analog modelling techniques for desktop and embedded implementations," *ArXiv*, vol. abs/2009.02833, 2020.

[2] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Proc. (ICASSP)*, 2019, pp. 471–475.

[3] E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Sound and Music Computing Conf.*, 2019, pp. 332–339.

[4] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. of the Intl. Conf. on Digital Audio Effects*, Sept. 2019, Proc. of the Intl. Conf. on Digital Audio Effects.

[5] J. D. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. Intl. Conf. Digital Audio Effects*, 2019, pp. 165–172.

[6] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Intl. Conf. on Learning Representations (ICLR)*, 2020.

[7] S. Shan, L. Hantrakul, T. Chen, M. Avent, and D. Trevelyan, "Differentiable wavetable synthesis," in *Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Proc. (ICASSP)*, 2022, pp. 4598–4602.

[8] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. Intl. Conf. Digital Audio Effects*, 2020, pp. 165–172.

[9] S. C. Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *Proc. Intl. Conf. Digital Audio Effects*, 2020, pp. 265–272.

[10] P. Bhattacharya, P. Nowak, and U. Zölzer, "Optimization of cascaded parametric peak and shelving filters with backpropagation algorithm," in *Proc. Intl. Conf. Digital Audio Effects*, 2020, pp. 101–108.

[11] G. Pepe, L. Gabrielli, S. Squartini, C. Tripodi, and N. Strozzi, "Deep optimization of parametric IIR filters for audio equalization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1136–1149, 2022.

[12] U. Zölzer, X. Amatriain, D. Arfib, J. Bonada, G. De Poli, P. Dutilleux, G. Evangelista, F. Keiler, A. Loscos, D. Rocchesso, et al., *DAFX - Digital Audio Effects*, John Wiley & Sons, 2002.

[13] J. O. Smith, *Physical Audio Signal Processing*, http://ccrma.stanford.edu/jos/pasp/, accessed 15.04.2022, online book, 2010 edition.

[14] M. Karjalainen, A. Härmä, U.K. Laine, and J. Huopaniemi, "Warped filters and their audio applications," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 1997, pp. 1–5.

[15] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," 2020.

[16] C. J. Steinmetz and J. D. Reiss, "auraloss: Audio focused loss functions in PyTorch," in *Digital Music Research Network One-day Workshop (DMRN+15)*, 2020.

[17] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," *ArXiv*, vol. abs/1910.11480, 2019.

[18] F. Esqueda, B. Kuznetsov, and J. D. Parker, "Differentiable white-box virtual analog modeling," *Proc. of the 24th Intl. Conf. on Digital Audio Effects (DAFx20in21)*, 2020/2021.

[19] B. De Man and J. Reiss, "APE: Audio perceptual evaluation toolbox for MATLAB," in *Proc. AES Convention*, 2014.

Oral Session 5:
Spatial Audio, Reverberation & Perception

DIFFERENTIABLE FEEDBACK DELAY NETWORK FOR COLORLESS REVERBERATION

Gloria Dal Santo, Karolina Prawda, Sebastian J. Schlecht, and Vesa Välimäki*

Acoustics Lab,
Department of Information and Communications Engineering,
Aalto University,
FI-02150 Espoo, Finland
{gloria.dalsanto, karolina.prawda, sebastian.schlecht, vesa.valimaki}@aalto.fi

ABSTRACT

Artificial reverberation algorithms often suffer from spectral coloration, usually in the form of metallic ringing, which impairs the perceived quality of sound. This paper proposes a method to reduce the coloration in the feedback delay network (FDN), a popular artificial reverberation algorithm. An optimization framework is employed entailing a differentiable FDN to learn a set of parameters decreasing coloration. The optimization objective is to minimize the spectral loss to obtain a flat magnitude response, with an additional temporal loss term to control the sparseness of the impulse response. The objective evaluation of the method shows a favorable narrower distribution of modal excitation while retaining the impulse response density. The subjective evaluation demonstrates that the proposed method lowers perceptual coloration of late reverberation, and also shows that the suggested optimization improves sound quality for small FDN sizes. The method proposed in this work constitutes an improvement in the design of accurate and high-quality artificial reverberation, simultaneously offering computational savings.

1. INTRODUCTION

Since the pioneering work of Schroeder and Logan [1], delay-based digital recursive structures have been used in reverberation synthesis [2]. Nowadays, one of the most widely used approaches in artificial reverberation is the feedback delay network (FDN), a system that generalizes the parallel comb-filter structure by interconnecting delays via a feedback matrix [3, 4, 5]. In FDNs, a commonly used approach is to first design a lossless prototype [6] to then achieve the desired frequency-dependent decay with attenuation filters [7, 8]. However, a common bane of systems utilizing comb filters is sound coloration [1]. Strong coloration is undesirable in artificial reverberation since it impairs the perceived sound quality.

Recent research suggests using modal decomposition to study the properties of the FDN in more detail [9]. The modal decomposition showed that the coloration in an FDN is related to the wide distribution of modal excitation values. In particular, modes with strong excitations are perceived as metallic ringing [10]. The modal excitation depends on all FDN parameters, and directly improving the coloration remains challenging. Recently, Schlecht proposed a method to achieve a uniform magnitude response and found the necessary conditions for an allpass FDN [11]. However, this approach suffers from temporal buildup of echoes [10], thus leaving the need for a more versatile method to design colorless FDNs.

Although many well-known reverb topologies, such as the Moorer-Schroeder [12], can be translated into FDN designs, the design of FDNs still presents several unresolved challenges. These arise from the inherent trade-off between computational complexity, mode density, and echo density. The cost of implementing the matrix-vector-multiplication for a single time step in an FDN increases with the number of delay lines and varies depending on the type of feedback matrix. However, the number of delay lines cannot be arbitrarily low, as there are certain dependencies between the delay lengths that become more severe as the number of delays decreases. In addition, a smaller number of delays decreases both the modal and the echo density, which leads to metallic sounding artifacts [13, 14].

Automatic tuning of FDN parameters has been previously explored in the literature, with genetic algorithms being widely used [15, 16, 17]. More recently, a multi-stage approach was employed to optimize FDN parameters to match a target room impulse response (IR)[18]. The input, output, direct gains, and delay lengths were optimized using a genetic algorithm. However, differences between the model and the target IR were revealed in the listening tests. To circumvent the challenge of optimizing infinite-impulse-response filters with differentiable machine-learning techniques, frequency sampling was used to implement a differentiable approximation of delay networks. An end-to-end deep-learning model was presented for the estimation of parameters, although only the absorption filters and input and output filters were estimated [19].

In this study, we present a novel approach to design FDNs for colorless artificial reverberation. To this end, we use a differentiable FDN (DiffFDN) in an optimization framework to learn a set of FDN parameters leading to less coloration. Specifically, we show that a narrower modal excitation distribution can be achieved without requiring the allpass property, offering more flexibility since the reverberation time (RT) values can be arbitrarily set after designing the prototype FDN. The perceptual evaluation against several common FDN designs shows that the proposed method successfully decreased perceived coloration.

The paper is organized as follows. Section 2 offers background information about FDNs and their modal decomposition. Section 3 introduces the proposed method of designing colorless FDNs. The results of the objective evaluation are presented in Section 4, and Section 5 shows the results of the listening test. Section 6 offers concluding remarks.

* Also at: Media Lab, Department of Art and Media, Aalto University, FI-02150 Espoo, Finland

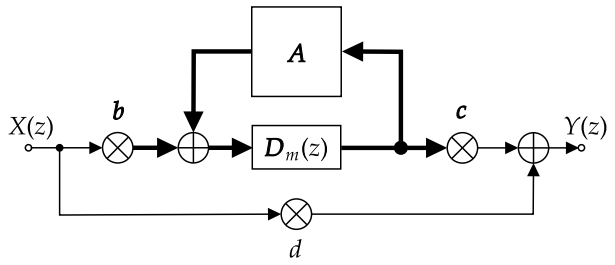


Figure 1: *Block diagram of a SISO FDN. Thin and thick lines indicate single- and multichannel connections, respectively.*

2. BACKGROUND

This section gives some background information about FDNs and presents related concepts that are relevant to the proposed method, such as modal decomposition and homogeneous decay in FDNs.

2.1. Feedback Delay Network

An FDN is a recursive system consisting of delay lines, a set of gains, and a scalar feedback matrix through which the delay outputs are coupled to the delay inputs. An example of a simple single-input single-output (SISO) FDN architecture is presented in Fig. 1. The transfer function of the FDN is

$$H(z) = \boldsymbol{c}^\top \big[\boldsymbol{D_m}(z)^{-1} - \boldsymbol{A}\big]^{-1}\boldsymbol{b} + d\,, \tag{1}$$

where $\boldsymbol{A}$ is the $N \times N$ feedback matrix, N being the number of delay lines. The $N \times 1$ column vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ and the scalar coefficient d respectively represent the input, output, and direct gains. The operator $(\cdot)^\top$ denotes the transpose. The vector $\boldsymbol{m} = [m_1, \ldots, m_N]$ defines the lengths of delays in samples. The corresponding delay matrix $\boldsymbol{D_m}(z)$ is created by taking a diagonal matrix with entries given by $[z^{-m_1}, \ldots, z^{-m_N}]$.

The system poles λ_i are the roots of the generalized characteristic polynomial $p(z)$ of the system, which is fully characterized by $\boldsymbol{m}$ and $\boldsymbol{A}$:

$$p(z) = \det(\boldsymbol{D_m}(z)^{-1} - \boldsymbol{A})\,. \tag{2}$$

The sum of the delays gives the order of the system, i.e., $\mathcal{M} = \sum_{i=1}^{N} m_i$ [20].

2.2. Modal Decomposition

The IR of the FDN can be represented as the sum of complex one-pole modes, or resonators, in the time domain [9]:

$$h(n) = \sum_{i=1}^{\mathcal{M}} h_i(n)\,. \tag{3}$$

Each mode $h_i(n)$ is defined by the pole λ_i and the residue ρ_i:

$$h_i(n) = |\rho_i||\lambda_i|^n e^{\jmath(n\angle\lambda_i + \angle\rho_i)}\,, \tag{4}$$

where $|\cdot|$ is the magnitude, $\angle$ indicates the argument of a complex number in radians, $\jmath = \sqrt{-1}$, and n indicates the discrete time index. The sum of the delay-line lengths $\mathcal{M}$ coincides with the number of poles.

The transfer function of the FDN (1) can be represented in terms of its poles and residues from its partial fraction decomposition as

$$H(z) = d + \sum_{i=1}^{\mathcal{M}} \frac{\rho_i}{1 - \lambda_i z^{-1}}\,, \tag{5}$$

which is often referred to as the modal decomposition of the FDN [9]. The excitation and initial phase of the i^{th} mode are determined by the magnitude $|\rho_i|$ and phase $\angle\rho_i$, respectively, of its corresponding residue, whereas the magnitude and the phase of the i^{th} pole, $|\lambda_i|$ and $\angle\lambda_i$, respectively, determine its decay rate and frequency.

2.3. Homogeneous FDN

For the design of an artificial reverberator, starting with a lossless prototype is beneficial. The FDN is said to be lossless if the roots of $p(z)$ have magnitude equal to one, i.e., $|\lambda_i| = 1$ for all i [21]. Frequency-dependent RT, here also denoted as T_{60}, is then easily achieved by extending the delay lines with a frequency-dependent attenuation filter [4].

In this study, we focus only on the specific case of frequency-independent homogeneous decay. This refers to the case where all modes experience the same rate of decay, i.e., $|\lambda_i| = \gamma$ for all i. Homogeneous decay is achieved with a feedback matrix $\boldsymbol{A}$ being the product of a unilossless matrix $\boldsymbol{U}$ and a diagonal matrix $\boldsymbol{\Gamma}$, whose entries are delay-proportional absorption coefficients, $\boldsymbol{\Gamma} = \text{diag}(\gamma^{\boldsymbol{m}})$. The feedback matrix can be expressed as

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Gamma}\,. \tag{6}$$

A matrix $\boldsymbol{U}$ is unilossless if, regardless of the choice of delays $\boldsymbol{m}$, its eigenvalues are unimodular and its eigenvectors are linearly independent. A matrix $\boldsymbol{U}$ satisfying the unitary condition, $\boldsymbol{U}\boldsymbol{U}^H = \boldsymbol{I}$, is also unilossless [22, 23]. As $\boldsymbol{U}$ is unilossless, the modal decay is controlled entirely by gain-per-sample parameter γ, where $0 \leqslant \gamma \leqslant 1$. The gain-per-sample in dB is

$$\gamma_{\text{dB}} = \frac{-60}{f_\text{s} T_{60}}\,, \tag{7}$$

where f_s is the sampling rate in Hz and T_{60} is the reverberation time defined as the time required for the sound level to decay by 60 dB from the initial steady-state value.

2.4. Coloration in FDN

In artificial reverberation, the properties of the resonating modes have direct implications on coloration. A flat magnitude response, implicitly achieved by the allpass property, is often desirable.

Schroeder and Logan [1] made the initial attempt to produce colorless artificial reverberation by establishing specific requirements for the reverberators in addition to a flat frequency response. Overlapping normal modes across all frequencies, equal RTs for each mode, sufficient echo density, lack of periodicity in the time domain, and no periodic or comb-like frequency responses were deemed necessary to achieve colorlessness [1]. Despite fulfilling the aforementioned conditions, however, the Schoreder series allpass did not attain complete colorlessness.

A recent study was conducted to further understand the role of modal excitation in late reverberation coloration [10]. Listening

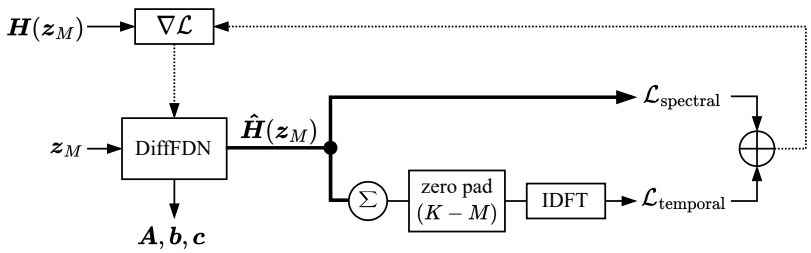


Figure 2: *Architecture of the proposed optimization workflow. Dotted lines indicate the stochastic gradient descent method of optimizing the parameters. Thin and thick lines indicate single- and multichannel connections, respectively.*

test results suggest that a narrow distribution of the modal excitation values $|\rho_i|$ tends to result in a flatter magnitude response. For large values of $|\rho_i|$, coloration starts to become noticeable. In agreement with [13], the study found that the perception of colorlessness correlates with the number of modes, and that more than 6000 modes are needed for an IR to be perceived as rather colorless.

The literature also shows that, for large values of $\mathcal{M}$, the modes of the FDNs are uniformly distributed [9], preventing additional coloration that usually results from clusters of modes. Nonetheless, a flat magnitude response and a uniform modal frequency distribution are insufficient to achieve colorlessness.

When the feedback matrix $\boldsymbol{A}$ is diagonal, the FDN takes the form of a parallel comb-filter structure. If the FDN is homogeneous, the transfer function in (1) is equivalent to a combination of comb filters, where each filter has the transfer function

$$H_i(z) = \frac{1}{1+\gamma^{m_i} z^{-m_i}}\,. \tag{8}$$

The contribution of each filter to the total energy of the response can be calculated as

$$\|H_i(z)\|^2 = \int_0^{2\pi} |H_i(e^{\imath\omega})|^2 \mathrm{d}\omega \tag{9}$$

$$= \frac{1}{1-\gamma^{2m_i}}\,. \tag{10}$$

Fundamentally, shorter delays m_i contribute more energy and produce strong, audible metallic-sounding comb peaks, whereas longer delays m_i contribute less energy and tend to be masked by the more dominant comb filters. In order to achieve colorless FDNs, we aim to avoid strongly recirculating short delays and encourage strongly exciting long delays.

2.5. Problem Statement

In this paper, we aim to optimize the feedback delay matrix $\boldsymbol{A}$, and input and output gains $\boldsymbol{b}$ and $\boldsymbol{c}$ such that the resulting IR is colorless. In this study, we keep the number and lengths of the delays fixed.

From previous studies, we know that coloration is little impacted by the choice of the frequency-dependent attenuation [10]. Thus, the optimization is performed on a long-ringing frequency-independent prototype FDN.

The proposed method utilizes two losses to improve coloration and temporal density. A stochastic gradient descent scheme is used to avoid convergence at spurious local minima. A parameter remapping guarantees a lossless FDN prototype at each optimization step.

3. FDN OPTIMIZATION

In the following, we present a method to reduce coloration in an FDN response for arbitrary RTs. Stochastic gradient descent is used to optimize the parameters of a differentiable FDN.

3.1. Differentiable FDN

This work applies the frequency-sampling method to approximate an FDN as a finite-impulse-response (FIR) filter. This is done by evaluating the delay matrix $\boldsymbol{D_m}(\boldsymbol{z_M})$ at the discrete frequency points in the vector

$$\boldsymbol{z}_M = [e^{\jmath\pi\frac{0}{M}}, e^{\jmath\pi\frac{1}{M}}, \ldots, e^{\jmath\pi\frac{M-1}{M}}], \tag{11}$$

where M indicates the total number of frequency bins equally distributed on the unit circle. The discrete-frequency transfer function of the FDN thus becomes

$$H(\boldsymbol{z}_M) = \boldsymbol{c}^\top \big[\boldsymbol{D_m}(\boldsymbol{z}_M)^{-1} - \boldsymbol{A}\big]^{-1}\boldsymbol{b} + d\,. \tag{12}$$

The diagram of the proposed architecture is shown in Fig. 2. We integrated $H(\boldsymbol{z}_M)$ into an optimization framework to estimate the set of FDN parameters based on a spectral and a temporal loss by gradient descent. The learnable parameters are the feedback matrix $\boldsymbol{A}$ and the input and output gain vectors $\boldsymbol{b}$ and $\boldsymbol{c}$, respectively. The delay lengths $\boldsymbol{m}$ are set at initialization, and kept constant during training. The direct gain d is set to zero. The FDN is set to have a homogeneous decay by forcing $\boldsymbol{A}$ to satisfy (6) for a given γ.

At each training step the estimated channel-wise transfer function $\hat{\boldsymbol{H}}(\boldsymbol{z}_M)$ is computed at M frequency bins. The input to the network is $\boldsymbol{z}_M$, where the value of M is sampled from the uniform distribution around values that ensure oversampling. This allows training the model at different sample rates, which proved to help avoiding narrow local minima and to improve convergence. To allow batch processing, the length of $\hat{\boldsymbol{H}}(\boldsymbol{z}_M)$ has to be constant for all values of M. This is achieved by zero-padding $\hat{\boldsymbol{H}}(\boldsymbol{z}_M)$ to length K. The network's output is evaluated in both the spectral and temporal domains. The IR of the system is computed from the K-point inverse discrete Fourier transform, $\hat{h} = \mathrm{IDFT}(\hat{H}(\boldsymbol{z}_M))$, where $\hat{H}$ is the system transfer function computed from the sum of the N channels. The process of zero-padding in the frequency domain results in zero-phase rate conversion [24], and allows evaluating the IR at different timestamps.

3.2. Feedback Matrix Parametrization

The unilossless matrix $\boldsymbol{U}$ is computed from the weights $\boldsymbol{W}$ of a parameterized linear layer. Matrix $\boldsymbol{U}$ is limited to the class of

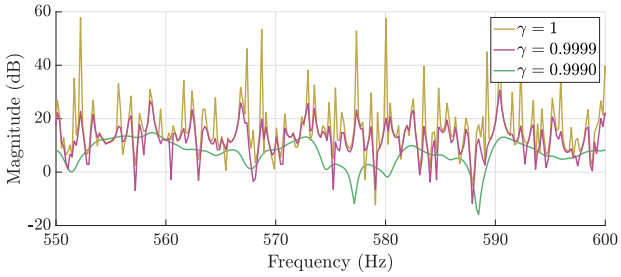


Figure 3: *Magnitude response of an FDN with random orthogonal feedback matrix and unitary input/output gains at different values of gain-per-sample value γ. For high values of γ, the resonances are better separated due to a smaller half-width.*

orthogonal matrices, satisfying the unitary condition for unilosslessness. To ensure orthogonality, at each optimization step $\boldsymbol{W}$ is mapped to a skew-symmetric matrix, and the matrix exponential is computed,

$$\boldsymbol{U} = e^{\boldsymbol{W}_{\text{Tr}} - \boldsymbol{W}_{\text{Tr}}^{\top}} , \tag{13}$$

where $\boldsymbol{W}_{\text{Tr}}$ is the upper triangular part of $\boldsymbol{W}$ and the operator $e^{(\cdot)}$ denotes the matrix exponential. The mapping in (13) implicitly ensures orthogonality of $\boldsymbol{U}$ and can be used in regular gradient descent optimizers without creating spurious minima [25].

3.3. Gain-per-sample

When $\boldsymbol{A}$ is lossless, i.e., $\gamma = 1$, the modulus of all system eigenvalues is equal to one: $|\lambda_i| = 1$. However, under this condition, evaluating $H(\boldsymbol{z}_M)$ on the unitary circle becomes unfeasible, as the discrete generalized characteristic polynomial $p(\boldsymbol{z}_M) = \det(\boldsymbol{D_m}(\boldsymbol{z}_M)^{-1} - \boldsymbol{A})$ becomes singular and non-invertible. To avoid instabilities, we use a homogeneous FDN where $\boldsymbol{A}$ is parameterized according to (6), and γ is set at initialization to a value lower than one and kept constant during optimization.

The value of γ used during optimization is chosen by examining the connection between the mean damping factor $\overline{\delta}$, used in room acoustics, and the mean spacing of resonance frequencies $\overline{\Delta f}$. To guarantee that the modes are well separated, the mean spacing of resonance frequencies should be larger than the average resonance half-width [26]

$$\overline{\Delta f} \gg \frac{\overline{\delta}}{\pi} . \tag{14}$$

In room acoustics, the limiting frequency below which the modes are well-separated is called Schroeder frequency, indicated here as $f_{\text{Schroeder}}$ [27]. This frequency marks the threshold above which an average of at least three modes falls within one resonance half-width. Using the fact that in FDNs the modal frequencies are nearly euqally distributed [9], we can derive the limiting average resonance half-width

$$\overline{\Delta f}_{|f=f_{\text{Schroeder}}} = 3\frac{f_{\text{s}}}{M} . \tag{15}$$

We can use the above conditions to determine the minimum value for T_{60} to be used during training

$$T_{60} \gg \frac{M \ln(10)}{\pi f_{\text{s}}} . \tag{16}$$

Increasing the value of T_{60} leads to modes with lower half-widths and greater separation between them. For a target T_{60}, the value of γ can be derived from (7). However, as γ approaches 1, the resonance peaks in the magnitude response become narrow, making obtaining a flat magnitude response by combining the resonances impossible. Fig. 3 shows the effect of increasing γ on the resonance width in a short section of the magnitude response. The sharp peaks visible when $\gamma = 1$ are significantly smoothed when $\gamma = 0.9990$.

Experiments showed good convergence of the loss used in the optimization when $T_{60} \leqslant 10\,\text{s}$. During inference γ is a free parameter, allowing to generate reverberation with any desired T_{60} value.

3.4. Parameters Initialization

We initialize the values of $\boldsymbol{W}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ by drawing from the normal distribution $\mathcal{N}(0, N^{-1})$.

The design of the delays is a rather non-trivial task that requires further constraints. To maximize the echo density, the delay lengths should be co-prime [28]. However, concentration of delays around a certain value may lead to perceivable strong fluctuation of energy over time. Moreover, low-order dependencies, which are integer linear combinations of delays that coincide with other integer linear combinations of delays with small coefficients, can also contribute negatively to the smoothness of the response [22]. To avoid degenerative patterns and ensure a smooth-sounding reverb, we choose delays that are logarithmically distributed coprime numbers leading to $M \geqslant 6000$.

3.5. Loss Function

The network is trained on two losses, $\mathcal{L}_{\text{spectral}}$ and $\mathcal{L}_{\text{temporal}}$, respectively, in the frequency and time domains. The spectral loss aims to minimize the frequency-domain mean-squared error between the absolute value of the predicted magnitude response for each channel and the target flat magnitude response. The temporal loss penalizes sparseness in the time domain. The total loss function is

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{spectral}}(\hat{\boldsymbol{H}}(\boldsymbol{z}_M)) + \alpha \mathcal{L}_{\text{temporal}}(\hat{h}) \\ &= \frac{1}{K}\sum_{i=1}^{N}\sum_{k=1}^{K}\Big(\Big|\hat{H}_i(\boldsymbol{z}_M[k])\Big| - 1\Big)^p + \alpha \frac{\big\|\hat{h}\big\|_2}{\big\|\hat{h}\big\|_1} , \end{aligned} \tag{17}$$

where $\hat{H}_i(\boldsymbol{z}_M)$ is the output of the network's i^{th} channel computed from the output of the i^{th} delay line and scaled by c_i. The operators $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the ℓ_1 and ℓ_2 norm, respectively. The value of the scaling factor α depends on the FDN size and is chosen during initialization to ensure that the temporal and spectral losses have similar magnitudes.

Audibility of a resonant frequency depends on its loudness and on the presence of neighbouring masker tones [29]. To account for tone masking effects, we adjust the exponent p in $\mathcal{L}_{\text{spectral}}(\hat{\boldsymbol{H}})$ based on the sign of the magnitude difference. Specifically:

$$p = \begin{cases} 2 & \text{for } \Big|\hat{H}_i(\boldsymbol{z}_M)\Big| - 1 \leqslant 0\,, \\ 4 & \text{for } \Big|\hat{H}_i(\boldsymbol{z}_M)\Big| - 1 > 0\,. \end{cases} \tag{18}$$

This adjustment ensures that higher loss values are assigned when the predicted magnitude response exceeds one. For negative differences, $\mathcal{L}_{\text{spectral}}(\hat{\boldsymbol{H}})$ corresponds to the mean squared error.

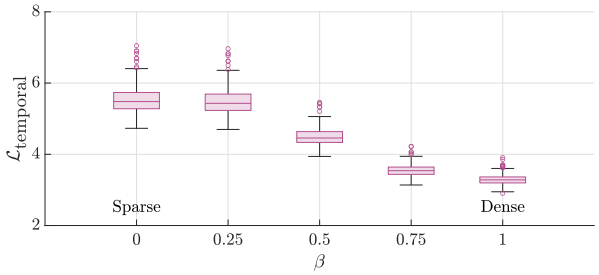


Figure 4: *Progression of temporal loss for different values of the interpolation parameter β. Density of the feedback matrix increases from left to right.*

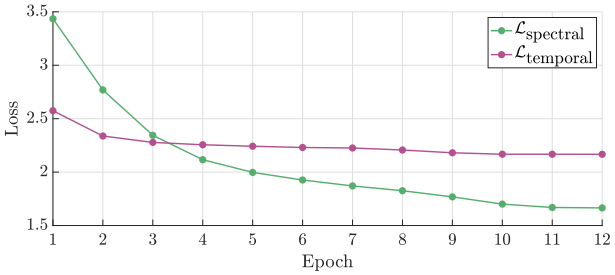


Figure 5: *Progression of spectral and temporal components of the loss function during optimization.*

The temporal loss $\mathcal{L}_{\text{temporal}}(\hat{h})$ is computed as the ratio of the ℓ_2 norm to the ℓ_1 norm of the estimated IR $\hat{h}$. We found that the absence of this term may lead to sparsity in the learnable parameters and cause the matrix $\boldsymbol{U}$ to converge towards either a diagonal matrix or its permutation. In this configuration, the magnitude response is periodic, with the spacing between peaks and troughs determined by the delay lengths, and the height of the peaks and the depth of the troughs depending on the gains. In time domain, this yields a sparse sequence of impulses whose sound is far from the intended Gaussian noise-like reverb.

To visualize the impact of the matrix on $\mathcal{L}_{\text{temporal}}(\hat{h})$, Fig. 4 summarizes the distribution of the loss values computed from an FDN with five different feedback matrices. The feedback matrix is interpolated between the values at initialization $\boldsymbol{U}$ and the identity matrix $\boldsymbol{I}$:

$$\boldsymbol{A}_\beta = e^{(1-\beta)\ \log(\boldsymbol{I}) + \beta\ \log(\boldsymbol{U})}, \tag{19}$$

where β is the interpolation parameter $0 \leqslant \beta \leqslant 1$. Operator $\log(\cdot)$ represents the matrix logarithm. For $\beta = 1$, the feedback matrix corresponds to the initial configuration $\boldsymbol{A}_1 = \boldsymbol{U}$, whereas for $\beta = 0$ matrix $\boldsymbol{A}_0$ coincides with the identity matrix $\boldsymbol{I}$. On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers stretch to include the most extreme data points that are not classified as outliers, and any outliers are plotted separately. The parameters of the FDN are initialized as described in Sec. 3.1, and the temporal loss is evaluated at 256 different values of M. The numbers in Fig. 4 show that the temporal loss $\mathcal{L}_{\text{temporal}}(\hat{h})$ grows for sparser feedback matrices, thus actively preventing convergence towards sparse matrices.

The evolution of losses at each epoch is shown in Fig. 5. Although $\mathcal{L}_{\text{spectral}}$ decreases at all displayed epochs, a near-steady value is attained by $\mathcal{L}_{\text{temporal}}$ after a few iterations. This controls the FDN and prevents convergence towards a set of comb filters.

4. OBJECTIVE EVALUATION

The following section presents the FDN configuration and the objective evaluation of the proposed method. The objective assessment is based on the modal excitation distribution.

4.1. Analyzed FDN Configurations

We evaluate a total of six FDN configurations, two sets of delay lengths for each of the three FDN sizes of $N = 4, 6, 8$. The values of the delay-line lengths are presented in Table 1. In the first delay set, the delay lengths were prime numbers distributed logarithmically. In the second delay set, only the second half of the delay lengths were logarithmically distributed, and the first half consisted of prime numbers with similar values. In all configurations, the total number of modes is $6000 < \mathcal{M} < 9000$.

Table 1: *Values of the delay-line lengths for each size N of the analyzed FDNs. In the delay set #1, all the delay lengths are logarithmically distributed prime numbers. For the delay set #2, half of the delay lengths are prime numbers with similar low values, and half are logarithmically distributed.*

N	**Delay Set #1**
4	[1499, 1889, 2381, 2999]
6	[997, 1153, 1327, 1559, 1801, 2099]
8	[809, 877, 937, 1049, 1151, 1249, 1373, 1499]
	Delay Set #2
4	[797, 839, 2381, 2999]
6	[887, 911, 941, 1699, 1951, 2053]
8	[241, 263, 281, 293, 1193, 1319, 1453, 1597]

During the training process, we used a sampling rate of $f_\text{s} = 48\,\text{kHz}$, and an inverse discrete Fourier transform of length $K = 480000$. The dataset consists of integer values M randomly selected from a uniform distribution ranging between $M_{min} = 0.8K$ and $M_{max} = K$. To train our model, we randomly selected 80% of the data from the dataset, and the remaining 20% was used for validation. The dataset size is 256 values of M. We set the batch size to 4, and employed the Adam optimizer [30] with a learning rate of $\eta = 10^{-3}$. Training was stopped after 15 epochs, as experiments showed no further improvement with extended training.

The choice of the gain-per-sample value γ is crucial when optimizing the feedback matrix. To satisfy (16) during training, we set $\gamma = 0.9999$, which implies $T_{60} = 1.439\,\text{s}$.

Configuration details and audio examples are available online [1]. The PyTorch implementation of the proposed method can be found in the dedicated repository [2]. A set of optimized FDN parameter values is readily available in the FDN Toolbox [5].

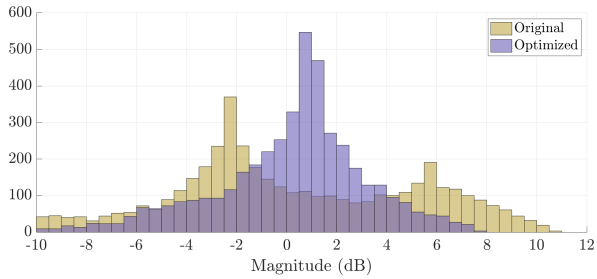


Figure 6: *Distribution of the modal excitation of an FDN with size $N = 4$ at the beginning (Original) and at the end of optimization (Optimized), which has led to a decrease of the loudest modal excitation by about 3 dB.*

4.2. Modal Excitation Distribution

We compare the FDN parameters after optimization with the corresponding FDN configurations at initialization. All the compared FDNs are homogeneous and have equal delays and gain-per-sample. We compute the modal decomposition (5) to analyze the modal excitation distribution of $|\rho_i|$.

The histograms in Fig. 6 show the distribution of the modal excitation at the beginning and at the end of the optimization processes for an FDN of size $N = 4$. The modal excitation values have been centered around 0 dB. At initialization, the distribution appears bimodal with the highest concentration of values around 6 dB and -2.5 dB. After optimization, the peak of the distribution is centered around 1 dB. The rightmost part of the distribution, which represents the modes with the highest excitation values, is important for coloration. In Fig. 6, the optimization attenuates the loudest modes by around 3 dB. The change toward narrower excitation distribution indicates an improvement in the coloration, which we further evaluate with a subjective test.

5. PERCEPTUAL EVALUATION

In the following, we describe a listening test conducted to evaluate the perceived coloration in the IRs of the differentiable FDN optimized with the proposed method.

5.1. Listening Test Procedure

The test followed the Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) recommendation [31], and it was carried out using the web audio API-based experiment software webMUSHRA developed by International Audio Laboratories Erlangen [32].

On each page, the listening test compared four sets of FDN parameters against a reference. The test items included six configurations, i.e., three FDN sizes $N = 4, 6, 8$ with two sets of delays. The sounds were compared for two different RTs. In total, there were 12 listening test pages with five stimuli each.

At the beginning of the test, a training page was presented to familiarize the subjects with the sound samples and to adjust the overall loudness. The loudness was kept unchanged during the rest of the test. The reference was a white Gaussian-noise sequence due to its ideal reverberation tail [12], and since it has a flat magnitude response by definition. During the test, the subjects were asked to rate the similarity between each of the presented items and the reference sound on a scale from 0 to 100. On each page, six sounds were assessed, including an anchor and the hidden reference. The hidden reference was an instance of white Gaussian noise different from the reference, to encourage the subject to compare samples based on their coloration rather than any possible subtle temporal features.

The test evaluated the coloration of the DiffFDN IRs for the configurations presented in Sec. 4.1. Each configuration was tested on a separate page where the number and lengths of the delays were constant, and only the feedback matrix, input and output gains were altered. In particular, the FDN implementation of the Schroeder-Moorer reverberator (SM) with N delay lines was used as the anchor, whereas for the remaining conditions, the random orthogonal feedback matrix (RO), the proposed optimized FDN (DiffFDN) and the Householder (HH) feedback matrix were used. The RO condition were the initial values of optimization of the DiffFDN. Unitary input and output gains were used for the HH condition. The direct gain d was set to zero in all cases. Each individual IR was normalized to ensure a constant root-mean-square value across conditions.

The experiment was conducted in a sound-insulated booth at the Aalto Acoustics Lab, with participants wearing Sennheiser HD650 headphones. The final items were presented to 12 listeners. One participant was excluded from the analysis as they failed to correctly identify the anchor more than four times in their responses. The average age of the participants whose results were analyzed was 28.6 years with standard deviation of 4.1, and none of them reported any hearing impairments. All the participants were either students or employees of the Aalto University Acoustics Lab, and had previous experience with the MUSHRA test.

The IRs presented in the first part of the test (expDE) had an exponential decaying envelope corresponding to $T_{60} = 2.5$ s and $\gamma = 0.99994$. The subjects were asked to compare the coloration of the FDN responses against that of decaying white Gaussian noise. To ease the grading process, the slider was labeled with 0 - *certainly colored*, 25 - *rather colored*, 50 - *fairly colored / colorless*, 75 - *rather colorless*, and 100 - *certainly colorless*.

The second part of the test (LL) focused on the coloration of the late reverberation part. It compared the non-decaying IRs with $T_{60} = \infty$ and $\gamma = 1$. In order to exclude the echo build up of the early reflection from the comparison, the test items started after the mixing time, i.e., at 6 s. Each audio example was 10 s long. The slider labels were the same as in the first part of the test.

5.2. Listening Test Results

The results of the listening test are shown in the box charts in Fig. 7 and Fig. 8 for the expDE and LL cases, respectively. The meaning of marks and whiskers on the chart is the same as in Fig. 4 (*cf.* Sec. 3.5). The shaded regions around the medians help comparing the sample medians across different box charts. Shaded regions that do not overlap indicate that the compared box charts have different medians at the 5% significance level based on a normal-distribution assumption.

Conducting the Shapiro-Wilk test [33] showed that even when excluding the reference and anchor conditions, the data did not follow a normal distribution. In addition, the Wilcoxon signed-

[1] `http://research.spa.aalto.fi/publications/papers/dafx23-colorless-fdn/`

[2] `https://github.com/gdalsanto/diff-fdn-colorless/`

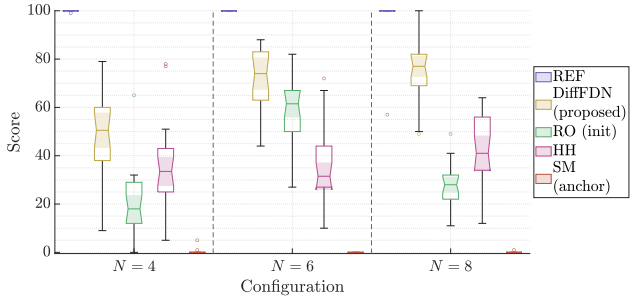


Figure 7: *Results of the listening test on exponential decaying IRs (expDE), showing that the proposed DiffFDN has the highest median score of colorlessness in all cases.*

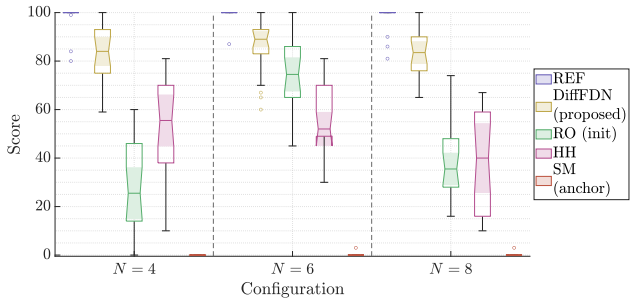


Figure 8: *Results of the listening test on the late reverberation, employing lossless FDNs (LL), showing that the proposed DiffFDN has the highest median score of colorlessness in all cases.*

rank test [34] was used to compare the distribution of the scores given to each pair of conditions within each page. To account for multiple comparisons (10 hypotheses per page), we applied the Bonferroni method to adjust the alpha level.

The p-values for all combinations of paired conditions suggest that all pairs of results are significantly different, with exception of the lossless case with $N = 8$ for RO and HH FDNs ($p = 0.68$). These results are indicated by the overlapping shaded regions of the corresponding box charts in Fig. 8. This may be due to the lack of early reflections in the lossless case, which makes differentiating between conditions difficult. Additionally, the configuration with a higher number of delays ($N = 8$) produces a denser output, which might result in a more challenging test.

The results presented in Figs. 7 and 8 show that the hidden reference and anchor signals were easily detected by most subjects, with few outliers. The median ratings for the proposed method were consistently higher than those for the remaining conditions, indicating that the optimization method was successful in improving colorlessness from the initial values.

In the first part of the test (expDE), increasing FDN sizes resulted in higher ratings for DiffFDN, with median values of 50.5, 74, and 77. The results for lossless FDNs (LL) reported a similar trend, with overall higher ratings primarily due to the elimination of the temporal build up. The proposed method was deemed rather colorless, with median ratings of 84, 89, and 83.5, respectively, for increasing FDN size. The RO matrix was rated more colorless than the HH matrix for the configuration with $N = 6$ delay lines, while it was rated more colored in the remaining configurations. This inconsistency may be attributed to the random sampling of the orthogonal matrix, which is performed without any preselection based on perceptual factors.

To emphasize the ratings relative to the proposed method, the

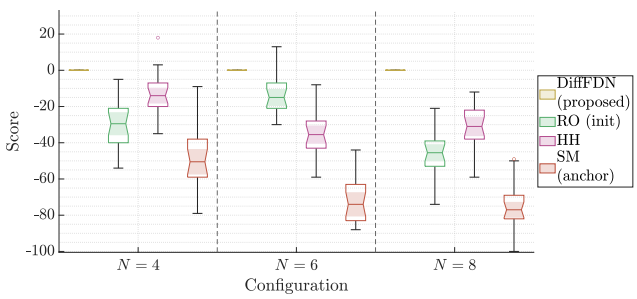


Figure 9: *Relative difference of the results of Fig. 7 from the results of the proposed DiffFDN method (expDE case).*

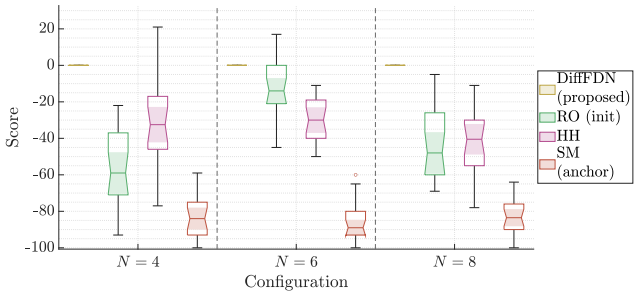


Figure 10: *Relative difference of the results of Fig. 8 from the results of the proposed DiffFDN method (LL case).*

box charts in Figs. 9 and 10 were calculated based on the difference between the DiffFDN and the remaining conditions. The ratings assigned to the reference are not displayed. The results show that in the majority of test questions, proposed method was rated higher than the remaining stimuli. Significant improvements are observed in the lossless case for $N = 4$. Specifically, the median value of the RO configuration was 59 lower than its optimized version (DiffFDN). Moreover, in both conditions, the median of the score differences and their 75th quartiles are consistently negative. The confidence intervals in Fig. 9 are noticeably narrower compared to those in Fig. 10, suggesting that the test on lossless FDNs was more challenging.

6. CONCLUSIONS

This work presents a method for designing colorless artificial reverberation using a differentiable feedback delay network (DiffFDN). The technique optimizes elements of the DiffFDN architecture—the feedback matrix as well as the input and output gains—to achieve a flat magnitude response. In addition, the temporal properties of the synthesized reverb are taken into account to avoid overly sparse results.

In the objective evaluation, we showed that the proposed method reduces the width of the modal excitation distribution, decreasing the number of loudest modes. This indicates that the DiffFDN achieves more colorless sound and flatter magnitude response of the produced reverb.

The results of the listening test show that, compared to other popular FDN designs, DiffFDN showed a significant improvement in reverberation quality. Reverberation obtained with DiffFDN was consistently graded as the most colorless among several conditions, placing it perceptually closer to white Gaussian noise than the other evaluated methods. This further confirmed the results of the objective assessment and proved that the proposed method successfully synthesizes colorless sound.

7. ACKNOWLEDGMENTS

This research belongs to the activities of the Nordic Sound and Music Computing Network (NordForsk project number 86892). The work of the first author was funded by the Aalto University School of Electrical Engineering. Special thanks go to Luis Costa for proofreading this paper.

8. REFERENCES

[1] M. R. Schroeder and B. F. Logan, ""Colorless" artificial reverberation," *IRE Trans. Audio*, vol. AU-9, no. 6, pp. 209–214, July 1961.

[2] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 5, pp. 1421–1448, July 2012.

[3] M. A. Gerzon, "Synthetic stereo reverberation: Part one," *Studio Sound*, vol. 13, pp. 632–635, Dec. 1971.

[4] J.-M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Proc. 90th AES Conv.*, Feb. 1991, pp. 1–12.

[5] S. J. Schlecht, "FDNTB: The feedback delay network toolbox," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, Sept. 2020, pp. 211–218.

[6] S. J. Schlecht and E. A. P. Habets, "On lossless feedback delay networks," *IEEE Trans. Signal Process.*, vol. 65, no. 6, pp. 1554–1564, Mar. 2017.

[7] S. J. Schlecht and E. A. P. Habets, "Accurate reverberation time control in feedback delay networks," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, Aug. 2017, pp. 337–344.

[8] K. Prawda, S. J. Schlecht, and V. Välimäki, "Improved reverberation time control for feedback delay networks," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, Sept. 2019, pp. 299–306.

[9] S. J. Schlecht and E. A. P. Habets, "Modal decomposition of feedback delay networks," *IEEE Trans. Signal Process.*, vol. 67, no. 20, pp. 5340–5351, Aug. 2019.

[10] J. Heldmann and S. J. Schlecht, "The role of modal excitation in colorless reverberation," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, Sept. 2021, pp. 206–213.

[11] S. J. Schlecht, "Allpass feedback delay networks," *IEEE Trans. Signal Process.*, vol. 69, pp. 1028–1038, Jan. 2021.

[12] J. A. Moorer, "About this reverberation business," *Computer Music J.*, vol. 3, no. 2, pp. 13–28, June 1979.

[13] M. Karjalainen and H. Järveläinen, "More about this reverberation science: Perceptually good late reverberation," in *Proc. AES Conv.*, Nov. 2001, pp. 1–8.

[14] N. Agus, H. Anderson, J.-M. Chen, S. Lui, and D. Herremans, "Perceptual evaluation of measures of spectral variance," *J. Acoust. Soc. Am.*, vol. 143, no. 6, pp. 3300–3311, June 2018.

[15] J. Coggin and W. Pirkle, "Automatic design of feedback delay network reverb parameters for impulse response matching," in *Proc. 141st AES Conv.*, Sept. 2016.

[16] M. Chemistruck, K. Marcolini, and W. Pirkle, "Generating matrix coefficients for feedback delay networks using genetic algorithm," in *Proc. 133rd AES Conv.*, Oct. 2012.

[17] J. Shen and R. Duraiswami, "Data-driven feedback delay network construction for real-time virtual room acoustics," in *Proc. 15th Int. Audio Mostly Conf.*, Sept. 2020, pp. 46–52.

[18] I. Ibnyahya and J. D. Reiss, "A method for matching room impulse responses with feedback delay networks," in *Proc. 153rd AES Conv.*, Oct. 2022.

[19] S. Lee, H.-S. Choi, and K. Lee, "Differentiable artificial reverberation," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 30, pp. 2541–2556, July 2022.

[20] D. Rocchesso and J. O. Smith, "Circulant and elliptic feedback delay networks for artificial reverberation," *IEEE Trans. Speech Audio Process.*, vol. 5, no. 1, pp. 51–63, Jan. 1997.

[21] M. A. Gerzon, "Unitary (energy-preserving) multichannel networks with feedback," *Electronics Letters*, vol. 12, no. 11, pp. 278–279, May 1976.

[22] S. J. Schlecht and E. A. P. Habets, "Feedback delay networks: Echo density and mixing Time," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, no. 2, pp. 374–383, Dec. 2017.

[23] J. Stautner and M. Puckette, "Designing multi-channel reverberators," *Comput. Music J.*, vol. 6, no. 1, pp. 52–65, Spring 1982.

[24] V. Välimäki and S. Bilbao, "Giant FFTs for sample-rate conversion," *J. Audio Eng. Soc.*, vol. 71, no. 3, pp. 88–99, Mar. 2023.

[25] M. Lezcano-Casado and D. Martınez-Rubio, "Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group," in *Proc. Int. Conf. Machine Learning*, May 2019, pp. 3794–3803.

[26] H. Kuttruff, *Room Acoustics, Fifth Edition*, CRC Press, June 2009.

[27] H. Kuttruff, "Eigenschaften und Auswertung von Nachhallkurven," *Acta Acust. United Acust.*, vol. 8, no. 4, pp. 273–280, Jan. 1958.

[28] S. J. Schlecht, *Feedback Delay Networks in Artificial Reverberation and Reverberation Enhancement*, Ph.D. thesis, Oct. 2017.

[29] H. Fastl and E. Zwicker, *Psychoacoustics: Facts and Models*, vol. 22, Springer Science & Business Media, Mar. 2006.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, Dec. 2014.

[31] ITU, "Method for the subjective assessment of intermediate quality level of audio systems," Recommendation ITU-R BS.1534-3, Oct. 2015.

[32] M. Schoeffler, S. Bartoschek, F.-R. Stöter, M. Roess, S. Westphal, B. Edler, and J. Herre, "WebMUSHRA—A comprehensive framework for web-based listening tests," *J. Open Research Software*, vol. 6, no. 1, 2018.

[33] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, Dec. 1965.

[34] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

PROBABILISTIC REVERBERATION MODEL BASED ON ECHO DENSITY AND KURTOSIS

Champ C. Darabundit *
CIRMMT †
CAML‡ , McGill University
Montréal, Canada
champ.darabundit@mail.mcgill.ca

Jonathan S. Abel
CCRMA §
Stanford University
Stanford, CA
abel@ccrma.stanford.edu

Wieslaw Woszczyk
CIRMMT
Sound Recording, McGill University
Montréal, Canada
wieslaw.woszczyk@mcgill.ca

ABSTRACT

This article proposes a probabilistic model for synthesizing room impulse responses (RIRs) for use in convolution artificial reverberators. The proposed method is based on the concept of echo density. Echo density is a measure of the number of echoes per second in an impulse response and is a demonstrated perceptual metric of artificial reverberation quality. As echo density is related to the statistical measure of kurtosis, this article demonstrates that the statistics of an RIR can be modeled using a probabilistic mixture model. A mixture model designed specifically for modeling RIRs is proposed. The proposed method is useful for statistically replicating RIRs of a measured environment, thereby synthesizing new independent observations of an acoustic space. A perceptual pilot study is carried out to evaluate the fidelity of the replication process in monophonic and stereo artificial reverberators.

1. INTRODUCTION

The reverberation of acoustic space is characterized by how a sound is reflected and absorbed as it travels from a source to a listener. Typically, a listener will observe the direct sound from a source, followed by a series of distinct echoes. These echoes are early reflections signifying the apparent geometries — such as nearby walls — of the acoustic space. The echoes in the space will rapidly build up, layering upon one another until giving way to a dense late reverberation. Over the same period, due to the absorptive properties of air and the surrounding environment, the acoustic energy will decay until the environment is actuated again [1].

The same observations can be made by a time domain analysis of a room impulse response (RIR), which may be procured through techniques such as balloon pop [2], sine sweep [3], or maximum-length sequence [4] measurements. Further analysis of RIRs in the frequency domain can highlight the behavior of modes and their decay. In the time domain, measures such as decay time (T_{60}) and echo density can be obtained, the latter of which is the focus of this article.

Absolute echo density (AED) measures the number of echoes per second, that is to say, the rate of non-zero impulses, in an RIR. In 1961, Schroeder noted that a property of "acoustically good rooms" was a high echo density and focused on designing artificial reverberators with a high echo density [5, 6]. Moorer, in 1979, observed that the distribution of echoes in an RIR becomes Gaussian in nature as the reflections become well-mixed. Echo density has been evaluated as a perceptually relevant reverberation parameter in [7, 8], and has been used as a measure to control the mixing time in feedback delay networks in [9]. Synthesis of RIRs based on a desired echo density for use in convolution reverberators has been pursued using a Poisson process [7, 8, 10], sparse FIR filters [11, 12], and velvet noise [13]. Echo density-focused methods permit RIRs synthesis from early reflections onward. In comparison, Gaussian noise synthesis methods described in [14, 15], only accurately model the late reverberation.

This article builds upon work by previous collaborators on the concept of normalized echo density (NED) [7, 8, 10, 16]. NED is a measure that compares the distribution of an RIR in a sliding time window to the Gaussian distribution and, as a result, estimates the echo density of an RIR. NED is inversely proportional to the statistical measure of kurtosis. In our contribution, we propose modeling the statistics of an RIR with a probabilistic mixture model. A mixture model characterizes a relatively complex probability distribution by modeling the distribution as a weighted sum of more basic distributions. Mixture models, particularly Gaussian mixture models (GMMs), have found wide usage in audio machine learning for tasks such as speaker identification [17–19]. GMMs have also been used to remove reverberation from sonar signals [20]. However, using mixture models to synthesize RIRs is — to the authors' knowledge — a novel approach. The proposed method has the potential to be an efficient and adaptive algorithm for synthesizing RIRs which provides better characterization of the RIR compared to prior methods.

Section 2 will review the proposed measures of echo density and the relationship between echo density and kurtosis. Section 3 will describe the method of moments used to derive the weights of the proposed mixture model and Section 4 will describe the mixture parameters used when modeling RIRs. Section 5 will detail example applications of the proposed method. Section 6 will detail a perceptual study with results presented in Section 7. Section 8 will conclude.

2. ECHO DENSITY MEASURES

Echo density, in units of echoes per second, was first proposed as a measure of reverberation quality in [5] and is now referred to as the absolute echo density (AED). This is to distinguish AED from the measure of normalized echo density (NED) [10].

* Research partially conducted while a Masters student at CCRMA
† Center for Interdisciplinary Research in Music, Media, and Technology
‡ Computational Acoustic Modeling Laboratory
§ Center for Computer Research in Music and Acoustics

2.1. Normalized and absolute echo density

NED, $\eta(t)$, was first proposed by Abel and Huang in [16] and is a statistical measure that estimates, within a time window, the departure of an RIR's distribution from the Gaussian distribution.

$$\eta(t) = \frac{1}{\text{erfc}(1/\sqrt{2})} \sum_{\tau=t-\beta}^{t+\beta-1} w(\tau)\mathbf{1}\left\{|h\left(\tau\right)| > \sigma\right\}. \tag{1}$$

Here, $h(t)$ is a time windowed portion of the RIR centered around t and 2β long in samples, $w(t)$ is a window function, and σ is the standard deviation of the $h(t)$. $\mathbf{1}\{\cdot\}$ is the indicator function, producing one when the argument is true and zero when false. We assume $h(t)$ to have zero-mean such that the standard deviation is:

$$\sigma = \left[\sum_{\tau=t-\beta-1}^{t+\beta} w(\tau)h^2(\tau)\right]^{\frac{1}{2}}. \tag{2}$$

The NED, η, can be related to AED, ρ, in echoes per second, through the following expression,

$$\rho = \frac{1}{\tau_d}\frac{\eta}{1-\eta}, \tag{3}$$

where τ_d is the echo duration in seconds. As η approaches 1, the AED in the window will increase describing a well-mixed reverberation. Conversely, as η approaches 0 the AED will decrease describing the sparse reflections found during early reflections.

2.2. Kurtotic measure of echo density

Abel and Huang, also proposed an alternative measure of echo density with a close similarity to the NED [16]. Their metric is related to the statistical metric of kurtosis α_4,

$$\eta_k(t) = \frac{\sigma}{\left[\frac{1}{3}\sum_{\tau=t-\beta}^{t+\beta-1} w(\tau)h^4(\tau)\right]^{\frac{1}{4}}} \propto (\alpha_4)^{-\frac{1}{4}}. \tag{4}$$

A measure of echo density based only on kurtosis was proposed independently by Usher in [21].

Kurtosis is defined as the fourth standard moment and has been incorrectly ascribed to the peakedness of a distribution or, alternatively, the fatness of a distribution's tails. The measure, however, has no direct bearing on these components. Kurtosis is better thought of as a movement of a distribution's mass from its shoulders to its center and tails [22, 23].

In this regard, kurtosis is an ideal measure of echo density. A window with a sparse number of echoes will have a distribution concentrated about its center and tails as the signal is dominated by silence and the occasional reflection. The distribution of said window will be highly kurtotic. Correspondingly, a well-mixed window will not be kurtotic, as its distribution is Gaussian.

3. A MIXTURE MODELING APPROACH

A probabilistic reverberation model aims to generate a synthetic RIR based on desired statistical properties. In [10], this was done using a Poisson process. This process, however, becomes computationally expensive for dense reverberations as echoes must be generated on an echo-by-echo basis. We propose using a mixture modeling approach to capture the statistics of an RIR.

The objective of a mixture model is — most often — to approximate the probability density of an empirical observation using a linear superposition of components with simpler density functions. As such, the primary task of mixture modeling is to derive the mixture parameters which best approximate a given observation. These parameters are the number of components, the parameters of individual component density functions, and the weights of each component within the mixture model.

The distribution of a RIR within a time window is relatively simple and assumptions can be made regarding the necessary number of components and their individual parameters. These parameters are discussed in Section 4.1. We are then chiefly concerned with deriving the weights of a finite set of components with known probability densities. To accomplish this we will use the method of moments, similarly described in [24, 25] instead of the more common maximum-likelihood-based expectation maximization (EM) method [17]. The method of moments is based on matching the raw moments of a mixture distribution to the raw moments of a desired distribution. Raw, central, and standard moments are common statistical measures used to evaluate statistical properties such the mean, variance, skewness, and kurtosis [26].

3.1. Raw, central, and standard moments

The n$^{\text{th}}$ raw moment of a random variable X with a continuous probability density function $f(x)$ can be computed using the expectation operator $E[\,\cdot\,]$:

$$\nu_n = E\left[X^n\right] = \int_{x\in\mathbb{R}} x^n f(x)dx. \tag{5}$$

For a discrete random variable, $E[\,\cdot\,]$ is given by:

$$E\left[X^n\right] = \sum_{l=0}^{L-1} x_l^n f(x_l). \tag{6}$$

For an empirical observation, $f(x_l) = \frac{1}{L}\ \forall x_i$. The first raw moment is the mean, represented by $\bar{X} = \nu_1$, and the central moment is a raw moment evaluated about the mean:

$$\mu_n = E\left[(X-\bar{X})^n\right]. \tag{7}$$

If the distribution has zero-mean, the central moment is equivalent to the raw moment. The second central moment is the variance $\mu_2 = \sigma^2$, the square of the standard deviation. The standard moment is the central moment scaled by the standard deviation,

$$\alpha_n = E\left[\left(\frac{x-\bar{x}}{\sigma}\right)^n\right]. \tag{8}$$

Kurtosis is, as previously mentioned, the fourth standard moment and can be expressed in terms of the second and fourth central moments

$$\alpha_4 = \frac{\mu_4}{\mu_2^2}, \tag{9}$$

In the context of modeling an RIR, parameterizing our model based on desired moments is intuitive if we aim to synthesize a response with a desired echo density and therefore kurtosis.

3.2. Method of moments

The method of moments aims to match the moments of a desired distribution, $\hat{Z}$, to the moments of a mixture distribution. The probability density function, $f_Z(x)$, of a mixture distribution Z is generated by the linear superposition of a finite set of M components with probability density functions $f_m(x)$ weighted by π_m,

$$f_Z(x) = \sum_{m=1}^{M} \pi_m f_m(x). \tag{10}$$

The weights, π_m, represent the amount by which each component is sampled. As such, the total sampling must sum to unity and each weight bounded:

$$\sum_{m=1}^{M} \pi_m = 1, \quad 0 \leq \pi_m \leq 1. \tag{11}$$

Applying (5) to (10), consider that the n$^{\text{th}}$ raw moment of the mixture distribution is the dot product of the mixture weights and the n$^{\text{th}}$ raw moment of the components:

$$E\left[Z^n\right] = \sum_{m=1}^{M} \pi_m \int_{x\in\mathbb{R}} x^n f_m(x)dx = \langle \pi \mid \nu_n \rangle \tag{12}$$

The method of moments is then formulated as a linear system relating the raw moments of a mixture's M components to the desired raw moments and accounting for the constraint in (11):

$$\boldsymbol{M}\boldsymbol{\pi} = \boldsymbol{\nu}, \tag{13}$$

where

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \nu_{1,1} & \nu_{1,2} & \cdots & \nu_{1,M} \\ \nu_{2,1} & \nu_{2,2} & \cdots & \nu_{2,M} \\ & & & \vdots \\ \nu_{n,1} & \nu_{n,2} & \cdots & \nu_{n,M} \end{bmatrix} \quad \begin{aligned} \boldsymbol{\pi} &= \begin{bmatrix} \pi_1 & \pi_2 & \cdots & \pi_M \end{bmatrix}^T \\ \boldsymbol{\nu} &= \begin{bmatrix} 1 & \hat{\nu}_1 & \cdots & \hat{\nu}_n \end{bmatrix}^T \end{aligned}$$

The moment matrix, $\boldsymbol{M}$, is a $n+1$ x M matrix formed by n$^{\text{th}}$ raw moments $\nu_{n,m}$ of the m$^{\text{th}}$ component. The weight vector, $\boldsymbol{\pi}$, is found by applying the inverse moment matrix to the desired moment vector, $\boldsymbol{\nu}$, of $\hat{Z}$ with moments $\hat{\nu}_n$. For M components, a unique system necessarily requires the computation of $n = M-1$ moments. After solving (13) for $\boldsymbol{\pi}$, the desired distribution is synthesized by random sampling of pseudo-random sequences with the same distribution as each component.

In this article, we utilize a mixture model based on the Gaussian distribution $X_i \sim \mathcal{N}(\nu_i, \sigma_i)$. Each component is parameterized by its mean μ_i and standard deviation σ_i. We chose Gaussian components because the desired response is generated by random sampling of scaled and shifted white Gaussian noise sequences. The Gaussian distribution has the following probability density function,

$$f\left(x \mid \nu, \sigma\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-\nu)^2}{\sigma^2}}. \tag{14}$$

The normal distribution, along with other distributions such as the binomial, gamma, and the Poisson families of distributions, has the property that its variance is — at most — a quadratic function of its mean value $\bar{x}$ [24, 27]. For these distributions, the moment matrix is readily generated based on only the parameters of the component distributions. Table 1 provides the first four raw moments of the Gaussian distribution.

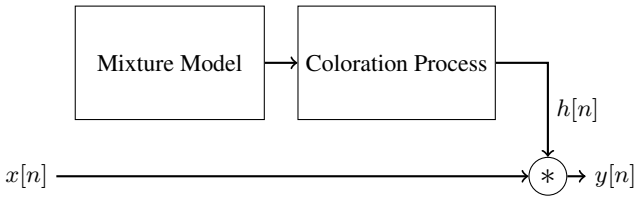


Figure 1: *Block diagram of mixture model-based RIR synthesis. A mixture model generates the desired "colorless" distribution which is filtered to generate the RIR $h[n]$. This impulse response is convolved with the input signal $x[n]$ to generate the reverberated signal $y[n]$.*

Take note that the method of moments in (13) does not guarantee $0 \leq \pi_m \leq 1$ and assumes the moments, and therefore the parameters, of the components are known. Tuning the component parameters to generate non-negative weights is an iterative optimization process [24]. However, based on prior knowledge of RIR signals, assumptions can be made regarding the necessary number of components and their individual parameters. These assumptions are described in the next section.

4. PROBABILISTIC MODELING BASED ON ECHO DENSITY

Our proposed probabilistic RIR model is compromised of two separate modeling processes, described in a block diagram in Figure 1. The first model is the mixture model which will generate what we denote the "echo response," a noise-like "colorless" impulse response based on the desired moment vector $\boldsymbol{\nu}$. Once the echo response has been generated, the color of the reverberation is modeled by a coloration process. Because the distribution of silence in each echo response is stochastic, RIRs synthesized with the same color will not be statistically correlated. The parameters of both models evolve with time.

This article is primarily concerned with the synthesis of the echo response, and we utilize an STFT-based frequency domain convolution to impose the spectra of a measured RIR onto the echo response. Similar RIR synthesis procedures have used equal rectangular bandwidth (ERB) band-wise exponential decay to color Poisson process generated noise [8]. Time-varying lowpass coloration filters have been used to color velvet noise [28]. In this section, we determine the parameters of the proposed RIR mixture model based on assumptions regarding the distribution of RIRs.

Table 1: *First four raw moments of $X_i \sim \mathcal{N}(\mu_i, \sigma_i)$*

ν_1	μ_i
ν_2	$\mu_i^2 + \sigma_i^2$
ν_3	$\mu_i^3 + 3\mu_i\sigma_i^2$
ν_4	$\mu_i^4 + 3\sigma_i^4 + 6\mu_i^2\sigma_i^2$

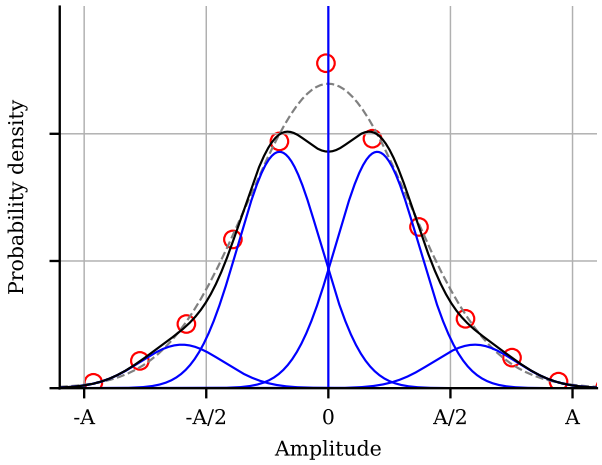


Figure 2: *Approximation of a Gaussian distribution* $f\left(x \mid 0, A/3\right)$ *(dashed grey) with the proposed mixture components (blue). The summation of the mixture components (black) matches the desired distribution except near zero where the model fails to capture the contribution from the degenerate distribution. A histogram of noise generated by the mixture — with bin centers and values indicated by the red scatter plot — demonstrates close adherence to the desired normal distribution.*

4.1. Mixture model parameters for RIR synthesis

In regards to the number of components, we assume that echo density — and consequently kurtosis — suitably characterizes the behavior of an RIR and it is unnecessary to consider moments higher than the fourth moment. By the nature of (13), a unique system necessitates a mixture with only five components.

In regards to the component parameters, consider that a large portion of an RIR signal during early reflections and into the late reverberation is the silence between echoes. To characterize this, we suggest that one component of the mixture should be a Gaussian with zero-mean and a variance that approaches an infinitely small value,

$$f_0(x) = \lim_{\sigma \to 0} f\left(x \mid \nu = 0, \sigma\right) = \delta(x). \tag{15}$$

The density function of this component can be abstractly represented by a delta function, and all raw moments of this component are equal to zero: $\nu_n = 0 \ \forall n$. This distribution is also referred to as a degenerate distribution [29].

We then assume — without loss of generalization — that the window has zero-mean and is normalized such that the samples lie within the amplitude range $\pm A$. The remaining components $(M = 4)$ are then evenly spaced in the range:

$$\nu_m = -A + \frac{2A}{M+1} m, \quad m = 1, 2, \ldots, M. \tag{16}$$

We propose that these components exhibit the same standard deviation. Overlapping the distribution is necessary to ensure a continuous distribution, but too much overlapping promotes the generation of negative weights as we are — in a sense — oversampling our distribution. Too little overlapping creates gaps in the amplitude distribution and the distribution is undersampled.

To ensure an ideal overlap of our component density functions, we proposed that the intersection between two neighboring Gaussians should sum to the maximum value of each component's density function. Determining the intersection of Gaussian probability densities $f\left(x \mid \nu_m, \sigma\right)$ and $f\left(x \mid \nu_{m+1}, \sigma\right)$ by substituting the expressions for ν_m and ν_{m+1} from (16) into (14), the point of intersection is,

$$x = \frac{A}{M+1}\left(2m - M\right). \tag{17}$$

The result is substituted back into (14) and evaluated against half the maximum value, $\frac{1}{2}\frac{1}{\sigma\sqrt{2\pi}}$. The proposed standard deviation for the components is:

$$\sigma = \frac{A}{M+1}\sqrt{\frac{1}{2\log(2)}} \tag{18}$$

Even with these parameters, it is possible to generate negative sampling weights. To counteract this, we propose that any negative weights are zero-ed and the resulting $\boldsymbol{\pi}$ vector is rescaled to observe the constraint in (11)

$$\pi_i = \begin{cases} \pi_i, & 0 \leq \pi_i \leq 1 \\ 0, & \pi_i < 0 \end{cases} \tag{19}$$

If the desired amplitude A is normalized, then the parameters of our components are predetermined and the inverse moment matrix $\boldsymbol{M}^{-1}$, can be stored ahead of computation. Since the desired distribution is assumed to have zero-mean, the desired moment vector $\boldsymbol{\nu}$ simplifies to,

$$\boldsymbol{\nu} = \begin{bmatrix} 1 & 0 & \mu_2 & 0 & \mu_4 \end{bmatrix}^T \tag{20}$$

In Figure 2, the proposed mixture is used to approximate a Gaussian distribution with $f\left(x \mid \mu = 0, \sigma = A/3\right)$. The sum of the component density functions, in black, approximates the Gaussian distribution except near zero where the contribution of the degenerate distribution is not well characterized. The histogram of a simulation, indicated by a red scatter plot, instead verifies the mixture behavior.

5. EXAMPLES

Given a desired NED in (4) and the definition in (9), it is difficult to make assumptions about the necessary values of μ_2 or μ_4. In the following examples, we demonstrate how to determine the desired moment vector for a desired AED value based on the properties of a generalized Poisson process. Alternatively, the desired moment vector can be found empirically when statistically replicating an RIR.

5.1. Static echo density based on the Poisson distribution

The generalized Poisson process proposed in [10] forms the process:

$$h(t) = \sum \alpha(t) \cdot \delta(t - \tau(t)). \tag{21}$$

The arrival times are represented as a set of Poisson impulses where $\tau(t)$ is drawn from an exponential distribution $\tau(t) \sim \exp\left\{-t/\rho(t)\right\}$ and ρ is the AED at time t. The amplitude, conversely, is drawn from a Gaussian distribution with a variance also dependent on the AED, $\alpha(t) \sim \mathcal{N}\left(0, \sqrt{1/\rho(t)}\right)$.

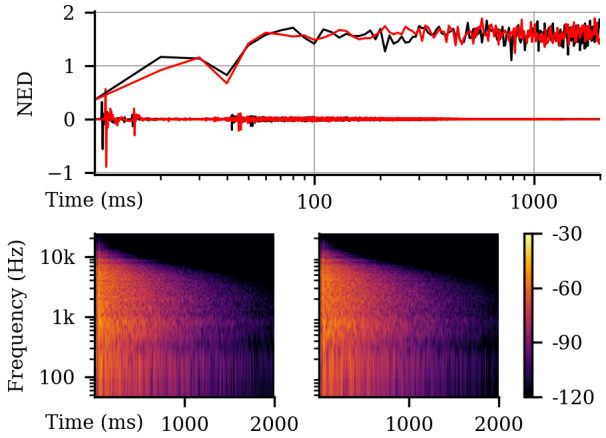


Figure 3: *Comparison of the left (black) and right (red) RIR signals and NED measured in Pollack Hall at McGill University. Spectrograms of each are displayed on the bottom left and right, respectively. The NED profiles are similar while the spectrograms are nearly identical.*

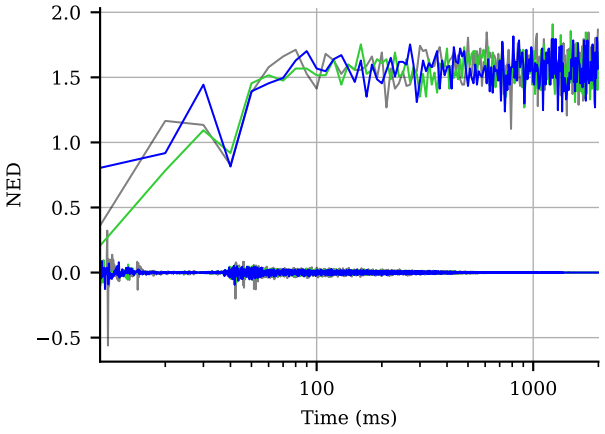


Figure 4: *RIR and NED of a measure (grey) of the Pollack Hall at McGill University replicated using the proposed method without block switching (blue) and with block switching (green). The synthesis window is 5 ms long while the analysis window was 10 ms long.*

The second and fourth moments of the Poisson process described by [10] can be derived based on the properties of a Poisson impulse [26]. A proof deriving the fourth moment of the Poisson process is provided in Section 11. The resulting second and fourth central moments are:

$$\mu_2^{PP} = \frac{1}{\rho} \cdot \left(\left(\frac{\rho}{f_s} \right)^2 + \frac{\rho}{f_s} \right) \tag{22a}$$

$$\mu_4^{PP} = \left(\frac{1}{\rho} \right)^2 \cdot \left(\left(\frac{\rho}{f_s} \right)^4 + 3\frac{\rho}{f_s} \right). \tag{22b}$$

These results can be substituted into the desired moment vector in (20). With these parameters, the proposed mixture described in Section 4.1 can generate a window of impulses with a fixed echo density. This method can be generalized to generate a synthetic RIR with a time-varying NED or AED profile.

5.2. Statistical replication of RIRs based on their moments

Another application of the method is for statistically replicating RIRs of measured rooms. The RIR replicas are akin to an independent observation of the acoustic space. Consider Figure 3, which compares the RIR measurement of Pollack Hall at McGill University with two microphones spaced roughly a human head width apart. The NED profile and spectrogram are nearly identical, and the two responses mainly differ in the arrival time of the reflections. Replicated RIRs can be used artistically in virtual acoustics to simulate different observations of a measured space.

To replicate a measured RIR, we propose using the overlap-add (OLA) algorithm [30] to analyze the raw moments of the measured RIR. Within each window, the echo response is generated and then colored in the spectral domain. In comparison to the Poisson process, the mixture model approach makes no assumptions about the distribution of the measured responses and instead attempts to holistically replicate the measured distribution.

Window size is an important parameter in the proposed method, as the generated echoes are randomly distributed within the window. This can result in temporal smearing when replicating early reflections as energy is no longer concentrated about distinct echoes. Smearing can be decreased by using a smaller window size at the cost of frequency resolution in the coloration process. A possible solution is to use a window size switching scheme akin to how transients are analyzed in audio coding [31]. In this scheme, a small window replicates early reflections and a longer window replicates the late reverberation. The result of the replication process with a single-window size and switched window size is shown in Figure 4. The main window size for both methods was 5 ms, and the small window size was 1.25 ms. The window size was switched after 30ms. These values were heuristically chosen and a formal metric based on NED merits further investigation.

The first reflections are largely deterministic based on the geometry of the acoustic space. When replicating RIRs it is better — in practice — to mix the first reflections from the measured RIR with early reflections onward from the replica. This is achieved by crossfading the measured RIR with the replicated RIR following a few early reflections. For the experiment described in Section 6, the signals were mixed 30ms after the initial direct path impulse with a mixing time of 5ms.

6. PERCEPTUAL EXPERIMENTS

Two informal perceptual pilot studies were conducted on students from CIRMMT and CCRMA. The first study compared the smoothness of noise generated with the Poisson process and the proposed mixture model. The mixture was generated for varying NED values using the parameters discussed in Section 5.1. The second study evaluated the efficacy of RIR replication using the Poisson process and the method discussed in Section 5.2. Both studies were administered online using the BeaqleJS framework [32]. All samples used in the study had their loudness normalized based on the EBU R 128 recommendation [33]. Samples

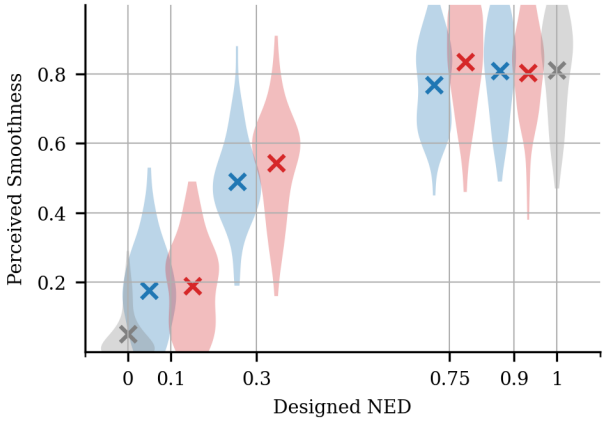


Figure 5: *Violin plot of perceived smoothness versus designed NED value. The Poisson process (blue) and mixture-generated samples (red) evaluations have been, respectively, shifted to the left and right from the designed values. The reference samples are in grey, and the "sputtery" reference with $\eta = 0.05$ is offset to the left.*

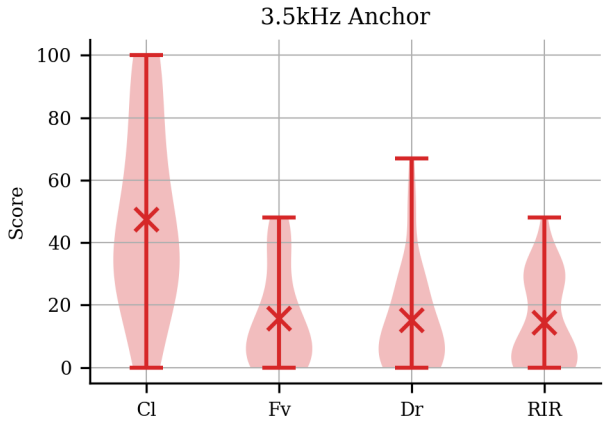


Figure 6: *Comparison of 3.5kHz lowpass anchor responses for clarinet (Cl), female voice (Fv), drums (Dr), and RIRs (RIR).*

used in the experiment can be accessed online [1].

6.1. Noise smoothness evaluation

Participants were asked to rate the smoothness of noise generated using the Poisson process and the proposed mixture for NED values $\eta = 0.1, 0.3, 0.75$, and 0.9 with a bandwidth of 10 kHz. Participants were asked to rate each noise sample on a scale from "sputtery" $[0 - \frac{1}{3})$ to "rough" $[\frac{1}{3} - \frac{2}{3})$ to "smooth" $[\frac{2}{3} - 1]$, terminology borrowed from an earlier study [10]. The participants were given two reference signals to ground their evaluations: a smooth reference — Gaussian noise — and a sputtery reference — Poisson process noise with $\eta = 0.05$.

6.2. RIR replication evaluation

Participants were asked to subjectively rate the quality of monophonic and stereo reverberated signals in a multi-stimulus test with hidden reference and anchor (MUSHRA) style test [34]. The test included a training phase where participants were familiarized with the process using a sample not included in the main evaluation and an artificial impulse response.

For the main evaluation, two environments were measured as references: Pollack Hall at McGill University and Memorial Church at Stanford University. RIR measurements of both environments were captured using two microphones representing the left and right channels. In the monophonic evaluations, only the left channel was utilized. Both channels were used in the stereo evaluations.

Test signals were generated by convolving generated and measured RIRs with anechoic audio samples. The anechoic audio samples consisted of female voice speech, drum, and clarinet signals. The voice and drum signal were chosen as they are largely impulsive in nature and the clarinet was chosen as it is pitched and less impulsive comparatively. The test RIR replicas were created using the proposed method and the Poisson process. Anchor RIRs were generated by filtering the measured RIRs at 3.5 and 7 kHz based on recommendations in [34]. Participants were additionally asked to evaluate the RIR signals by themselves.

[1] https://ccrma.stanford.edu/~champ/dafx23

7. RESULTS AND ANALYSIS

Both experiments had a total of 10 participants. However, one participant was removed from the RIR replication study as they consistently rated the anchor and reference signals as having an equal perceptual quality.

7.1. Noise smoothness results

The results of the noise smoothness study are shown in Figure 5. Our experiment demonstrates that samples generated with the proposed mixture model have a similar perceived smoothness as samples generated through the Poisson process. The proposed mixture has a higher mean perceived smoothness compared to the Poisson process for three NED levels. These results suggest that the proposed model performs similarly to prior methods for generating samples with fixed NEDs.

7.2. RIR replication study results

The overall results for all samples are shown in a violin plot in Figure 7a. The overall mean rating for the proposed method is evaluated as being second in quality to the reference. However, it is worth recognizing that participants had difficulty discriminating the hidden anchors and there were a small number of participants in the pilot study.

Evaluation was particularly difficult in the case of the clarinet sample which was the only audio sample with sustained sounds. A comparison of the 3.5kHz lowpass anchor for different samples is shown in Figure 6. This would suggest that the evaluation of RIR quality is better performed with audio signals that are transient or with the RIR signal by itself.

Further analysis is obtained if samples are delineated by monophonic or stereo test signals as in Figure 7b and Figure 7c, respectively. In Figure 7b, the performance of the proposed mixture

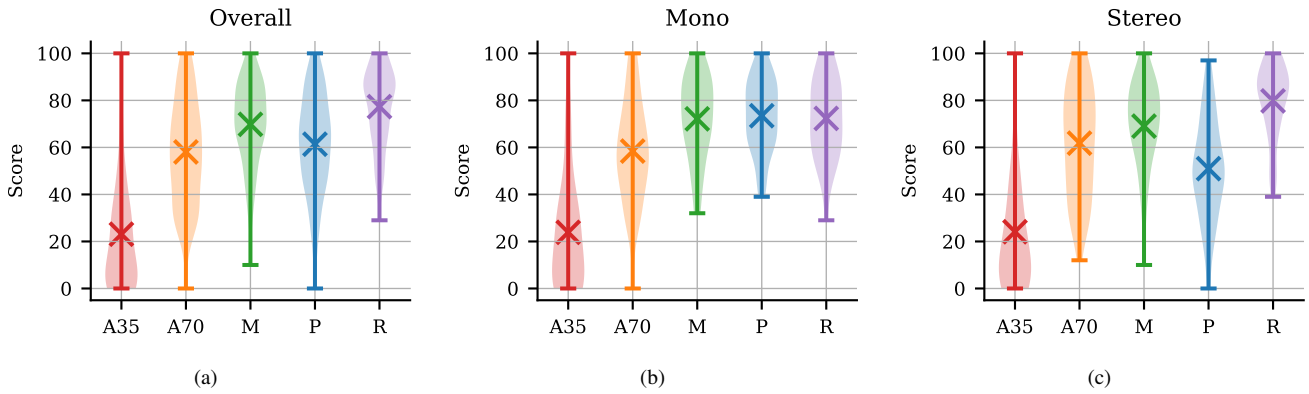


Figure 7: *Violin plot of MUSHRA-style RIR replication study. (a) compares the overall results, (b) the monophonic sample results, and (c) the stereophonic sample results. A35 corresponds to the 3.5kHz lowpass anchor, A70 to the 7.0kHz lowpass anchor, M to the proposed mixture model, P to the Poisson process, and R to the reference.*

model is similar to the Poisson process method for monophonic samples. However, in Figure 7c, the performance of the proposed mixture method is better than that of the Poisson process method stereo samples. This suggests that when rendering virtual acoustic scenes, the proposed method may provide better results.

8. CONCLUSION AND FUTURE WORK

In this article, we proposed a new method for synthesizing artificial RIRs for use in convolution reverberators. The proposed method uses a mixture model with component parameters designed specifically for RIR synthesis. A pilot perceptual study demonstrates the efficacy of the proposed method in comparison to a previously proposed RIR modeling technique. Future work is needed to properly evaluate the quality of the proposed method in a larger study. Frequency domain convolution was used to color the generated echo responses, and future work should evaluate the quality of different coloration techniques.

9. ACKNOWLEDGMENTS

The authors would like to thank all the volunteers at CIRMMT and CCRMA for taking the time to participate in the pilot study. The authors would like to thank Gary Scavone for his discussions on designing the pilot study.

10. REFERENCES

[1] Thomas D. Rossing, Ed., *Springer Handbook of Acoustics*, Springer, New York, NY, 2 edition, 2014.

[2] David Griesinger, "Beyond MLS — occupied hall measurement with FFT techniques," in *101st Convention Audio Eng. Soc.*, November 1996.

[3] Swen Müller and Paulo Massarani, "Transfer-function measurement with sweeps," *J. of the Audio Eng. Soc.*, vol. 49, no. 6, pp. 443–471, June 2001.

[4] Douglas D. Rife and John Vanderkooy, "Transfer-function measurement with maximum length sequences," *J. of the Audio Eng. Soc.*, vol. 37, no. 6, pp. 419–444, June 1989.

[5] M. R. Schroeder and B.F. Logan, "'Colorless' artifical reverberation," *J. of the Audio Eng. Soc.*, vol. 9, no. 3, pp. 192–197, July 1961.

[6] M. R. Schroeder, "Natural sounding artificial reverberation," *J. of the Audio Eng. Soc.*, vol. 10, no. 3, pp. 219–223, July 1962.

[7] Patty Huang, Jonathan S. Abel, Hiroko Terasawa, and Jonathan Berger, "Reverberation echo density psychoacoustics," in *125th Convention Audio Eng. Soc.*, October 2008.

[8] Kimberly Kawczinski and Jonathan S. Abel, "Perceptually-informed features for room reverberation modeling and prediciton," in *149th Convention Audio Eng. Soc.*, October 2020.

[9] Sebastian J. Schlecht and Emanu el A. P. Habets, "Feedback delay networks: Echo density and mixing time," in *IEEE/ACM Trans. on Audio, Speech, and Language Process.*, 2017.

[10] Patty Huang and Jonathan S. Abel, "Aspects of reverberation echo density," in *123rd Convention Audio Eng. Soc.*, October 2007.

[11] Per Rubak and Lars G. Johansen, "Artificial reverberation based on a pseudo-random impulse response," in *104th Convention Audio Eng. Soc.*, May 1998.

[12] Per Rubak and Lars G. Johansen, "Artificial reverberation based on a pseudo-random impulse response II," in *106th Convention Audio Eng. Soc.*, May 1999.

[13] Matti Karjalainen and Hanna Jäveläinen, "Reverberation modeling using velvet noise," in *30th Int. Conf. Audio Eng. Soci.*, Saariselk a, Finland, March 2007.

[14] Jean-Marc Jot, Laurent Cerveau, and Olivier Warusfel, "Analysis and synthesis of room reverberation based on a statistical time-frequency model," in *103rd Convention Audio Eng. Soc.*, September 1997.

[15] James A. Moorer, "About this reverberation business," *Comput. Music J.*, vol. 3, no. 2, pp. 13 – 27, 1979.

[16] Jonathan S. Abel and Patty Huang, "A simple, robust measure of reverberation echo density," in *121st Convention Audio Eng. Soc.*, October 2006.

[17] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag, Berlin, 2006.

[18] Geoffrey J. McLachlan and David Peel, *Finite Mixture Models*, John Wiley & Sons, Inc., New York, 2000.

[19] Douglas A. Reynolds, *A Gaussian mixture modeling approach to text-independent speaker identification*, Ph.D. thesis, Georgia Institute of Technology, August 1992.

[20] Guo Haoquan, Chu Fuzhao, and Zhu Daizhu, "Research on Gaussian mixture auto-regressive reverberation modeling and whitening algorithm," in *2021 IEEE Conf. Signal Proces., Commun., and Comput.*, 2021.

[21] John S. Usher, *Subject evaluation and electroacoustic theoretical validation of a new approach to audio upmixing*, Ph.D. thesis, McGill University, Montreal, Canada, September 2006.

[22] J.J.A. Moors, "The meaning of kurotsis: Darlington reexamined," *The American Statistician*, vol. 40, no. 4, pp. 283–284, November 1986.

[23] Kevin P. Balanda and H. L. MacGillivray, "Kurtosis: A critical review," *The American Statistician*, vol. 42, no. 2, pp. 111–119, May 1998.

[24] Bruce G. Lindsay, "Moment matricies: Applications in mixtures," *The Annals of Statistics*, vol. 17, no. 2, pp. 722–740, June 1989.

[25] D. M. Titterington, A.F.M. Smith, and U. E. Makov, *Statistical Analysis of Finite Mixture Distributions*, Wiley, New York, 1985.

[26] Athanasios Papoulis and S. Unnikrishna Pillai, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, NY, 4th edition, 2002.

[27] Carl N. Morris, "Natural exponential families with quadratic variance functions," *The Annals of Statistics*, vol. 10, no. 1, pp. 65–80, 1982.

[28] Vesa Välimäki, Bo Holm-Rasussen, Denoit Alary, and Heidi-Maria Lehtonen, "Late reverberation synthesis using filtered velvet noise," *Appl. Sci.*, vol. 7, 2017.

[29] V. Sundarapandian, *Probability, Statistics, and Queueing Theory*, PHI Learning Private, New Delhi, India, 2009.

[30] Julius O. Smith III, *Spectral Audio Signal Processing*, `ccrma.stanford.edu/~jos/sasp/`, 2011 edition, accessed April 2023.

[31] Marina Bosi, Karlheinz Brandenburg, Schuyler Quackenbush, Louis Fielder, Kenzo Akagiri, Hendrik Fuchs, Martin Dietz, Jügen Herre, Grant Davidson, and Yoshiaki Oikawa, "ISO/IEC MPEG-2 advanced audio coding," *J. of the Audio Eng. Soc.*, vol. 45, no. 10, pp. 789–814, October 1997.

[32] Sebastian Kraft and Udo Zölzer, "BeaqleJS: HTML5 and JavaScript based framework for the subjective evaluation of audio quality," in *Linux Audio Conf. 2014*, Karlsruhe, Germany, May 2014.

[33] "R128 loudness normalisation and permitted maximum level of audio signals," Recommendation, European Broadcasting Union, Geneva, August 2020.

[34] "BS.1534: Method for the subjective assessment of intermediate quality level of audio systems," Recommendation, International Telecommunication Union, October 2015.

11. APPENDIX

We derive the fourth moment of the Poisson process proposed by [10] and discussed in Section 5.1. The n[th] moment of the process described in (21) evaluates,

$$E\left[\alpha^n(t)\delta^n(t-\tau(t))\right]. \tag{23}$$

The Gaussian distributed amplitude and Poisson impulse are mutually independent, therefore their expectations are separable

$$E\left[\alpha^n(t)\right]E\left[\delta^n(t-\tau(t))\right]. \tag{24}$$

The n[th] moment of the $\alpha(t)$ can be readily found based on the Gaussian distribution, and the first four moments are provided in Table 1. The Poisson impulse can be constructed by taking the time derivative of a Poisson process which is modeled as a discrete Poisson distributed variable $X \sim \mathrm{P}\left(k \mid \lambda = \rho/f_s \cdot t\right)$. The Poisson distribution represents the probability there are k events when λ is the mean number of events. From the linear property of the expectation and time derivative operators, one can verify that the mean of the Poisson impulse is ρ/f_s.

Derivation of the second moment of the Poisson impulse is given in [26]. Here, we present the derivation of the fourth moment. The fourth moment of a Poisson process for non-overlapping times $t_1 < t_2 < t_3 < t_4$ is related to the Poisson impulse by the partial derivatives $\frac{\partial}{\partial t_i}$ of the correlation operator R,

$$R_{\delta\delta\delta\delta}(t_1,t_2,t_3,t_4) = \frac{\partial^4 R_{xxxx}(t_1,t_2,t_3,t_4)}{\partial t_1 \partial t_2 \partial t_3 \partial t_4}. \tag{25}$$

The correlation of a randomly distributed variable is equivalent to its expectation,

$$R_{xxxx}(t_1,t_2,t_3,t_4) = E\left[X(t_1)X(t_2)X(t_3)X(t_4)\right], \tag{26}$$

and t_1, t_2, t_3, t_4 are non-overlapping. The expectation above can be re-expressed as,

$$E\left[X(t_1)X(t_2)X(t_3)X(t_4)\right] = \prod_{i=1}^{4} E\left[X(t_i) - X(t_{i-1})\right], \tag{27}$$

with $x(t_0) = 0$. This product can be expanded as the first moment for all $x(t_i)$ is $\rho/f_s t$

$$R_{xxxx}(t_1,t_2,t_3,t_4) = \frac{\rho^4}{f_s^4} \cdot t_1(t_2-t_1)(t_3-t_2)(t_4-t_3) \tag{28}$$

Applying the partial time derivatives in (25), the resulting fourth moment is

$$R_{\delta\delta\delta\delta} = \frac{\rho^4}{f_s^4} + \frac{\rho}{f_s}(\delta(t_1-t_2) + \delta(t_2-t_3) + \delta(t_3-t_4)) \tag{29}$$

where the $\delta(t)$ terms account for the discontinuities between non-overlapping time segments.

A VIRTUAL INSTRUMENT FOR IFFT-BASED ADDITIVE SYNTHESIS IN THE AMBISONICS DOMAIN

Hilko Tondock and Henrik von Coler

Audio Communication Group
TU Berlin
voncoler@tu-berlin.de

ABSTRACT

Spatial additive synthesis can be efficiently implemented by applying the inverse Fourier transform to create the individual channels of Ambisonics signals. In the presented work, this approach has been implemented as an audio plugin, allowing the generation and control of basic waveforms and their spatial attributes in a typical DAW-based music production context. Triggered envelopes and low frequency oscillators can be mapped to the spectral shape, source position and source width of the resulting sounds. A technical evaluation shows the computational advantages of the proposed method for additive sounds with high numbers of partials and different Ambisonics orders. The results of a user study indicate the potential of the developed plugin for manipulating the perceived position, source width and timbre coloration.

1. INTRODUCTION

The dynamic distribution of sound on multichannel loudspeaker systems, known as spatialization or diffusion, has a long history in electronic and electroacoustic music. With a variety of algorithms, such as Vector Base Amplitude Panning (VPAB), Higher Order Ambisonics (HOA) or Wave Field Synthesis (WFS), any recorded or synthesized sound can be placed and moved in 2D or 3D space, resulting in immersive audio experiences. Spatial sound synthesis describes methods which treat spatial aspects as an integral part of the synthesis process, usually at an early stage in the algorithm. Such procedures can create sound events with inherently connected timbral and spatial properties. This concept has been investigated for many established synthesis paradigms, such as granular synthesis [1], physical modeling [2], FM Synthesis [3] and additive synthesis [4], respectively spectral modeling [5].

The approach presented in this paper is based on an efficient IFFT-based additive synthesis in the Ambisonics domain [6, 7]. This allows the synthesis of spectra with a large number of sinusoidal components, each with individual spatial attributes. To enable the use of this sound synthesis method beyond experimental music, an implementation as a VST plugin is presented. Thus, it can be included in a generic digital audio workstation (DAW) workflow. Synthesis parameters like pitch, timbre and spatial distribution can be controlled with sequences, envelopes and other modulators. This opens new possibilities, since the production of popular music for spatial audio setups and binaural playback is continuously gaining importance.

The remainder of this paper is organized as follows. Section 2 introduces the theoretical basics of the underlying algorithm. Section 3 deals with the implementation of the algorithms in the plugin and gives a brief overview of the current possibilities to integrate the plugin into digital music production environments. Section 4 presents a user study and evaluates the collected data.

2. ALGORITHM

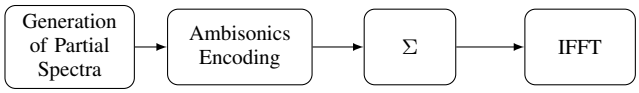


Figure 1: *Flow chart of the spatial IFFT-based additive synthesis.*

Figure 1 shows the main stages of the implemented IFFT-based additive synthesis in the Ambisonics domain. In the first step, a complex spectrum is generated for each partial. All partial spectra are then encoded into individual Ambisonics signals, which are summed to produce a single frequency-domain Ambisonics signal. Finally, each Ambisonics channel is transformed to the time domain via IFFT, resulting in an Ambisonics-encoded time domain signal. The individual steps are explained in the following sections.

2.1. Generation of Partial Spectra

In order to reduce the computing load for additive synthesis with a high number of partials, inverse DFT [8] and the inverse FFT [9] have been proposed for the signal model

$$x[n] = \sum_{i=1}^{N} a_i[n] \cos\left(\omega_i[n] T n + \varphi_i\right), \tag{1}$$

where N is the number of partials, a_i is the partial amplitude, ω_i is the partial frequency in radians, $T = \frac{1}{f_s}$ is the sampling period and φ_i the initial phase of the partial.

Partial spectra can be approximated by a small number of significant bins, around $K \approx 7$, in the magnitude spectrum, referred to as the *spectral motif* [10]. K is accountable for the approximation error and should be selected on the basis of the window properties and the desired signal-to-noise ratio (SNR). This approximation results in a computational benefit of roughly $\frac{H}{K}$, where H is the hop size, and is most effective with a suitable window that has as few decisive bins as possible. Such a window must have both a narrow main lobe and high side lobe suppression, which are conflicting requirements. Another optional approximation in [9]

expects stationary frequency, phase and amplitude of a sinusoid in one IFFT frame of size N. Hence, the associated window can be constructed as a real and even sequence, meaning symmetric with respect to the origin, thus the Fourier transform of the window is real.

In a harmonic signal, each partial corresponds to a shifted and scaled spectral motif. The inverse Fourier transform of the weighted sum of these spectral motifs produces a windowed signal in the time domain which has to be divided by the inverse window function to reverse the effect of the window. In order to concatenate frames, the overlap-add process is used, which involves weighting the time-domain signals of frames with a window that adds up to 1 in the overlapping region (e.g., a triangular window) [9]. Since the frequencies of partials in a frame are assumed to be stationary, the concatenation of adjacent frames with varying frequency components may cause modulations from phase cancellation in the overlapping regions. These distortions can be reduced by matching the phases of subsequent frames [9].

Several methods aim at improving the basic IFFT approach. Distortions can be minimized by using chirp signals instead of stationary frequencies in a frame [10, 11]. Laroche [12] presented an algorithm that avoids the overlap-add process and directly concatenates successive frames. This approach is refined by calculating optimal coefficients of the spectral motif and an optimal window function [13]. In [14], the subband sinusoidal synthesis algorithm is presented which evaluates a time-domain sinusoid at only a few samples and applies a pair of DFT to interpolate it to N samples. Aiming for higher SNR [12, 13] should be considered. If non-stationary sinusoids are mandatory, the approaches in [10, 14] will provide a foundation. The overlap-add method [9] is implemented because of its low computational cost and because there is no need for non-stationary sinusoids in the developed plugin.

2.1.1. Spectral Motif

The DTFT of a windowed sinusoid in the time domain yields the shifted Fourier transform of the window, multiplied by the complex amplitude [15, p. 797]:

$$\begin{aligned} X(e^{j\omega}) = \frac{A}{2} e^{j\varphi} W\left(e^{j(\omega-\omega_0 T)}\right) \\ + \frac{A}{2} e^{-j\varphi} W\left(e^{j(\omega+\omega_0 T)}\right), \end{aligned} \tag{2}$$

where $W\left(e^{j(\omega\pm\omega_0 T)}\right)$ is the shifted Fourier transform of the window sequence $w[n]$. Further, the DFT corresponds to equally spaced samples of the DTFT:

$$X[k] = X\left(e^{j\omega}\right)\Big|_{\omega=\frac{2\pi k}{NT}}. \tag{3}$$

The spectral motif can be created from a Fourier transform of a suitable oversampled time-domain window function, by extracting the main lobe values. The main lobe can be defined as the values between the first local minima to the left and right of the global maximum for most window functions. These values are stored in a lookup table and used later for synthesis. For this purpose, the window types Hann, Kaiser ($\beta = 8$) and Blackman-Harris are considered.

Besides the different types of windows and the window length M, it is of particular importance to emphasize the differences between true even symmetry and DFT-even symmetry windows, regarding discrete window functions [16]. True even symmetry refers to a sequence that has symmetric samples about an assumed midpoint and DFT-even symmetry relates to a true even sequence with the right endpoint removed. If the periodic continuation of a time-domain window function is symmetric with respect to the origin, the Fourier transform of this window will be real valued. In the discrete domain, if M is of even length and a symmetric window, there will be no peak value of exactly one. Instead two maximum values exist to fulfill the symmetric qualifications. In order to place the partials at the appropriate frequency location in the spectrum, it is desirable to use a window with a single maximum value of one. Also, it is a computational benefit to calculate only real values. Thus, the basis for the spectral motif is assumed to be a DFT-even symmetric sequence of an even length M.

Assuming $N = 512$ and $f_\mathrm{s} = 48000$ Hz, the frequency resolution results in $\frac{f_\mathrm{s}}{N} = 93.75$ Hz. If we assume the human ear resolves differences in frequencies of 1 Hz, a spectral motif with much finer resolution is required. Larger values for M result in a better frequency resolution in general but the main lobe width remains the same regarding the number of samples. Zero-padding is hence used to obtain an interpolated or rather oversampled version of the main lobe of the spectrum. Applying casual zero-padding (adding zeros after a signal) ruins the advantage of a real-valued spectrum. A possible solution utilizes zero-phase zero-padding to obtain a real valued spectrum, where first, the zero-frequency component is shifted to the center of the spectrum and afterwards zeros are inserted in the middle of the window without destroying the DFT-even symmetry.

Then, after the zero-phase shift is reversed, the oversampled main lobe or eventually additional side lobes of the spectrum of the window can be normalized and stored. The number of stored samples is dependent on the selected values for K and O.

2.1.2. Spectral Encoding

Since the objective is to generate real-valued signals and N is even, it suffices to calculate only $\frac{N}{2}+1$ samples of the spectrum, exploiting the symmetry properties of the DFT. The spectral motif $W[k]$ has to be placed correctly for the given frequency f_P of a partial. The floating bin location is equal to $k_\mathrm{f} = T_\mathrm{s} N f_\mathrm{P}$ in an N-point DFT. Because it is only possible to fill integer positions of k, firstly, the closest bin location is calculated by $k_\mathrm{i} = \lfloor k_\mathrm{f} + 0.5 \rfloor$ and the remaining distance $k_\mathrm{r} = k_\mathrm{i} - k_\mathrm{f}$ is stored separately.

Then, the decisive bins are calculated for $0 < k < \frac{N}{2}$ by

$$\begin{aligned} X_\mathrm{P}[k_\mathrm{i}+k] = \frac{A_\mathrm{P}}{2} W\left[\lfloor O(k_\mathrm{r}+k) + O_\mathrm{M} \rfloor\right] \cos(\varphi_\mathrm{P}) \\ + j\frac{A_\mathrm{P}}{2} W\left[\lfloor O(k_\mathrm{r}+k) + O_\mathrm{M} \rfloor\right] \sin(\varphi_\mathrm{P}) \end{aligned} \tag{4}$$

where O_M is the middle index of the spectral motif and φ_P is the current phase of the partial. Particular attention should be given to bin locations $k \leq 0$ and $k \geq \frac{N}{2}$. Directly from the definition of the DFT follows that for $k = 0$ and $k = \frac{N}{2}$:

$$X_\mathrm{P}[k] = A_\mathrm{P} W\left[\lfloor O(f_\mathrm{k}+k) + O_\mathrm{M} \rfloor\right] \cos(\varphi_\mathrm{P}). \tag{5}$$

Due to the periodicity of the DFT, frequency domain aliasing has to be considered. Based on the symmetry properties of the DFT follows for $k < 0$ and $k > \frac{N}{2}$:

$$\begin{aligned} X_\mathrm{P}[k_\mathrm{i}+k] = \frac{A_\mathrm{P}}{2} W\left[\lfloor O(k_\mathrm{r}+k) + O_\mathrm{M} \rfloor\right] \cos(\varphi_\mathrm{P}) \\ - j\frac{A_\mathrm{P}}{2} W\left[\lfloor O(k_\mathrm{r}+k) + O_\mathrm{M} \rfloor\right] \sin(\varphi_\mathrm{P}) \end{aligned} \tag{6}$$

The obtained spectrum forms the basis for the spatial encoding.

2.2. Ambisonics Encoding

Ambisonics is a surround sound technology that was developed in the 1970s [17, 18] and was further investigated, for example, in [19]. It is a spatial audio representation that encodes incoming sound sources in three dimensions, allowing for the capture and reproduction of a full-sphere sound field.

The first-order Ambisonics B-format refers to encoding pressure and velocity at the origin of a sound field to four channels. However, first-order Ambisonics is limited in terms of spatial resolution. To resolve this issue, high-order Ambisonics (HOA) is calculated using higher numbers of so called spherical harmonics. These are functions on the surface of a sphere that can describe the distribution of sound pressure. A comprehensive summary of Ambisonics-related theory can be found in [20].

A real-valued set of spherical harmonics can be defined

$\forall(n,m) \in \mathbb{N}_N$ by

$$Y_n^m(\theta,\phi) = N_n^{|m|} P_n^{|m|}(\sin\theta) \begin{cases} \cos(m\phi), & m \geq 0 \\ \sin(m\phi), & m < 0 \end{cases} \tag{7}$$

where the elevation angle θ is 0 at the horizontal plane and positive in the upwards direction, while the azimuth angle ϕ is 0 pointing in face direction and increases counter-clockwise. $N_n^{|m|}$ represents a normalization constant. The associated Legendre functions P_n^m are defined by

$$P_n^m(x) = (-1)^m \left(1-x^2\right)^{\frac{m}{2}} \frac{d^m}{dx^m} P_n(x), \tag{8}$$

with the Legendre polynomials P_n which can be expressed in the Rodrigues representation

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} \left(x^2-1\right)^n. \tag{9}$$

There are various conventions for ordering and normalizing Ambisonics channels. The ambiX format [21] is a widely used standard and is implemented in the proposed algorithm. It specifies the Ambisonics channel number (ACN) for channel ordering:

$$ACN(n,m) = n^2 + n + m + 1 \tag{10}$$

and the Schmidt semi-normalized (SN3D) convention

$$N_n^m = \begin{cases} 1, & m = 0 \\ (-1)^n \sqrt{2\frac{(n-m)!}{(n+m)!}}, & m \neq 0 \end{cases} \tag{11}$$

is proposed in terms of normalization.

2.2.1. Displacement Function

In the proposed synthesis approach, Ambisonics encoding takes place in the spectral domain, after generating the complex spectra and before applying the IFFT. Each partial of the additive synthesis is assigned individual angles θ and ϕ. Equation 7 determines the spatial gains for each Ambisonics channel, depending on these angles. The previously calculated real and imaginary parts of the spectra are then scaled by these spatial gains.

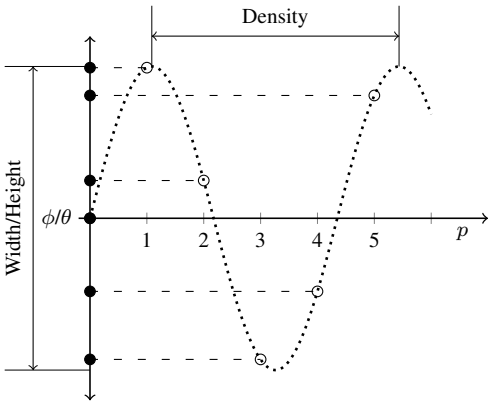


Figure 3: *Example for the placement of the first six partials of the sine type displacement function.*

With increasing number of partials, individual control of their positions is not possible. Hence, various displacement functions can be used to distribute the partials with few meta-parameters. The standard sinusoidal displacement function for a single dimension is defined as:

$$D[p] \coloneqq \frac{H}{2} \sin\left(\frac{2\pi S p}{P}\right) \tag{12}$$

where H is the width or height, S determines the horizontal or vertical dispersion pattern, P is the total number of partials and p is the partial index. The resulting values are added to the global azimuth or elevation angle of the sound, as shown in Figure 3.

2.3. IFFT

2.3.1. Overlap-add

After Ambisonics encoding, an IFFT is aplied to each Ambisonics channel. The resulting time-domain signal needs to be divided by the window function that was applied to reverse the windowing effect. However, this results in large amplitude values at the boundaries of the frame, which can cause distortion when overlapping with successive frames in the overlap-add process. To ensure equal energy in the overlapping sections, a proper weighting function must be applied. The triangle window is often used for this purpose and the calculation of the inverted synthesis window and the amplitude weighting function can be combined. To sufficiently reduce distortion at the edges, it is recommended to choose the hop size H less than or equal to 25% of the frame size [22].

2.3.2. Phase Adjustment

Phase adjustment is a technique that can be used to attenuate amplitude modulations that result from phase cancellation due to destructive interference of adjacent frames. The specific implementation depends on which signal information is stored and whether there is any frequency variation at all. In this case, only the last phase information for each partial is stored, with the initial phase of a partial located at $N/2$. The points at which adjacent overlapping frames have the same energy are $\frac{1}{2}(N+H)$ and $\frac{1}{2}(N-H)$, respectively. First, the initial phase at $\frac{1}{2}(N+H)$ of the current

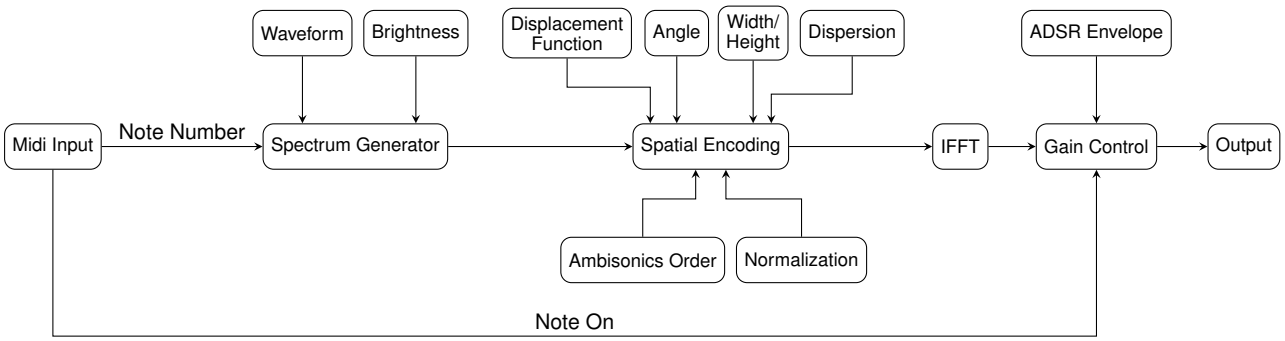


Figure 2: *Signal flow of the plugin.*

partial is calculated as follows:

$$\varphi_t\left[\frac{N}{2}\right] = \left(\varphi_{t-1}\left[\frac{1}{2}(N+H)\right] + \pi f_t H T_\mathrm{s}\right) \bmod 2\pi, \quad (13)$$

This value is then used for partial placement, and the new phase is stored for the next callback, calculated by:

$$\varphi_t\left[\frac{1}{2}(N+H)\right] = \left(\varphi_t\left[\frac{N}{2}\right] + \pi f_t H T_\mathrm{s}\right) \bmod 2\pi. \quad (14)$$

If the frequency remains constant throughout the lifetime of a partial, the calculation reduces to a single floating-point remainder function call. However, it is important to note that the adjustment only works effectively when the frequencies of consecutive frames are close together.

3. IMPLEMENTATION

3.1. Plugin

Figure 2 shows the signal flow of the plugin. Table 1 lists the adjustable parameters and their range of values. This first version of the plugin features the basic waveforms triangle, sawtooth, and square wave. All these waveforms are well suited for the production of lead and bass sounds in popular electronic music, especially when the high frequency content is changed over time by temporal envelopes. A pure sine wave has been added as an anchor for the test phase, since it emphasizes artifacts and distortions. The pitch of the waveform is controlled by MIDI. Since the plugin itself does not feature pitch modulation capabilities, a signal just consists of stationary sinusoids after being generated based on the note-on event. Assuming stationary sinusoids, a real-valued spectral motif can be constructed. Frequency-domain additive synthesis can handle aliasing by constraining the maximum permitted frequency of a partial below half of the sampling rate. To get control over the overtones, the Brightness parameter is implemented as follows:

$$A[p] = A[p]e^{-\frac{p}{d}}, \quad (15)$$

where $A[p]$ is the amplitude of the current partial index p and d is an adjustable damping factor.

The plugin is based on the JUCE[1] framework and integrates the KFR[2] library for FFT/IFFT and is made publicly available[3].

[1]https://github.com/juce-framework/JUCE
[2]https://github.com/kfrlib/kfr
[3]https://github.com/ringbuffer-org/spadd

Table 1: *Plug-in parameters and range of values.*

Parameter	Range of Values
Waveform	Sine, Triangle, Sawtooth, Square, Noise
Noise Density	1 to 10000
Brightness	0.5 to 250
Distance	1.0 to 100.0
Horizontal Displacement Function	Sine, Cosine, Sawtooth, Square
Azimuth Angle	-180 to 180 degree
Width	0 to 180 degree
Horizontal Dispersion	0 to 300
Vertical Displacement Function	Sine, Cosine, Sawtooth, Square
Elevation Angle	0 to 90 degree
Height	0 to 90 degree
Vertical Dispersion	0 to 300
Ambisonics Order	0^{th},1^{st}, 2^{nd}, 3^{rd}
Normalization	SN3D, N3D
Gain Attack	0 to 5 s
Gain Decay	0 to 8 s
Gain Sustain	0 to 1
Gain Release	0 to 8 s

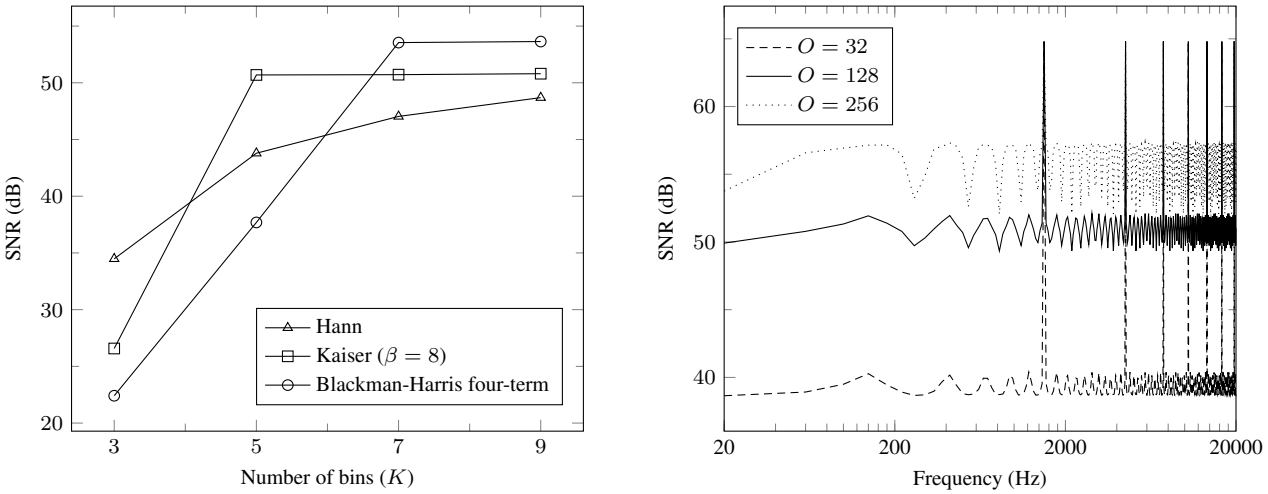
3.1.1. Distortion Analysis

Distortion artifacts mainly stem from the overlap - add process, which are minimized here by calculating an appropriate phase adjustment. Other adjustable sources of distortion are O and K, regarding the type of window function. In order to inspect the distortion and to choose suitable parameters for the implemented algorithm, the SNR is calculated as

$$\mathrm{SNR} = 10\log_{10}\left(\frac{E_\mathrm{signal}}{E_\mathrm{noise}}\right) \quad (16)$$

where E is the energy of a signal $x[n]$ of length L:

$$E = \sum_{n=0}^{L-1} |x[n]|^2. \quad (17)$$



(a) (b)

Figure 4: *Signal-to-noise ratio for (a) different window functions with increasing number of bins (K) and fixed overlap (O = 128) and (b) for the Kaiser window (β = 8) with K = 5 dependent on O and the frequency.*

E_{signal} in this case takes a sinusoidal reference signal consistent with the phase of the synthesized output signal as a basis and E_{noise} relates to the difference between the reference signal and the output signal. The first output frame is discarded because there is no overlap-add data yet. Note that this property generally affects transient signals, e.g. for $H = 256$ and $f_s = 48000$ Hz an attack time of 5.3 ms occurs. Figure 4(a) shows the SNR for different windows dependent on K with fixed O.

As expected, window functions with a narrower main lobe converge towards a maximal possible SNR earlier, whereas window functions with a higher side lobe suppression reach a higher maximal SNR at the expense of higher K. Hence, to achieve even higher SNR, a Kaiser window with $\beta > 8$ or a Blackman-Harris window with more than four coefficients could be applied. The SNR is also dependent on O and the frequency, as shown in Figure 4(b). The peaks of the curves emerge, if the frequency matches an exact bin location. Note that O does not have to be a power of two in general but can be computed efficiently. An SNR ≥ 40 dB was considered to be sufficient to exclude audible artifacts in [12]. Hence, the Kaiser window ($\beta = 8$) with $K = 5$ and $O = 256$ and the Blackman-Harris four-term window with $K = 7$ and $O = 128$ are taken into account during implementation.

3.1.2. Performance Analysis

Music production setups usually utilize either internal or external sound cards (interfaces) for enhanced usability and performance. Audio drivers typically provide a range of buffer sizes between 16 and 2048 samples, which leads to round-trip latencies between 0.7 and 43 milliseconds, assuming $f_s = 48000$ Hz. Measurements were taken on an Intel Core i7-7700HQ, representing today's average consumer CPU. The benchmark test utilizes a scope based timer, containing only the frequency and time-domain specific process functions, that means all function calls which are the same for both methods are excluded, e.g. constructing the basic signal information or midi processing. The plugin was embedded in a digital audio workstation during testing and ran over 2000 callbacks. Random partials were generated every callback, taking into account the branches which stem from (4), (5) and (6). Figure 5 contains the results of the measurements.

As Figure 5(a) shows, the frequency-domain approach is faster than the time-domain approach above 10 partials. 100 partials are sufficient for synthesizing the deterministic part of sounds from most musical instruments, considering the upper frequency limit of our auditory system and the fundamental frequencies of musical sounds. For 100 partials, the frequency domain approach is roughly twice as fast as the time domain approach ($0.239\,\text{ms}$ / $0.112\,\text{ms}$). For low frequency sounds, especially synthetic ones, more partials may be necessary. If non-harmonic partials are considered in order to create stochastic components, the number of partials can be increased to several hundreds [23]. For 1000 partials, the performance ratio between time domain and frequency is about 6 ($1.783\,\text{ms}$ / $0.295\,\text{ms}$), which indicates a significant advantage of the IFFT approach.

As shown in Figure 5(b), the performance load increases with the Ambisonics order, independent of the number of partials.

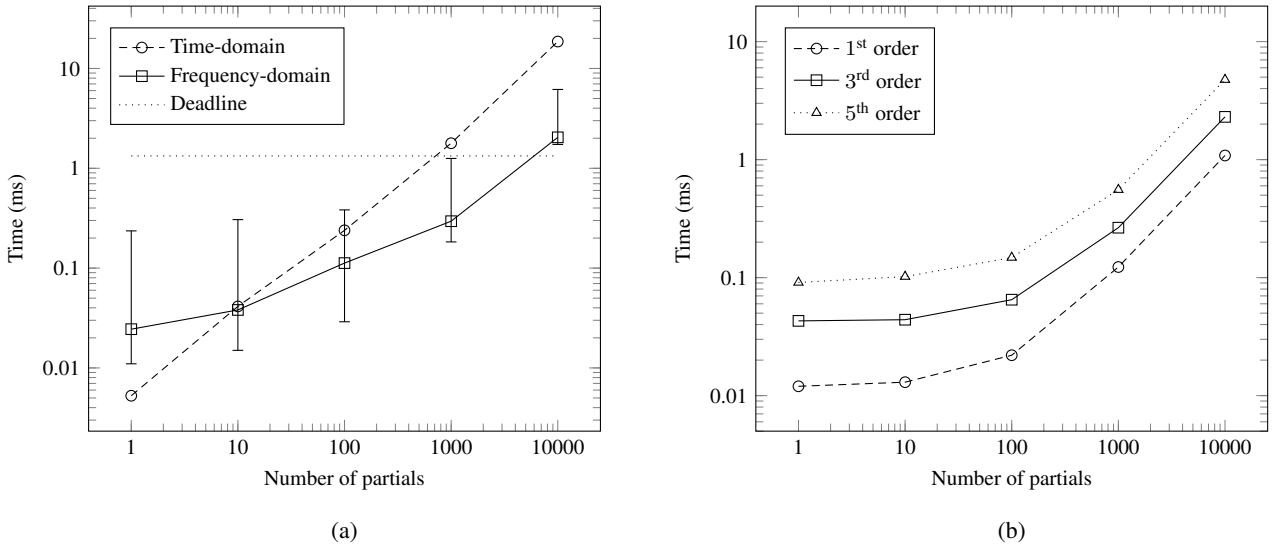
3.2. DAW Integration

Typically, working with digital audio material involves using a digital audio workstation (DAW) for recording, editing or synthesizing audio content. DAWs mostly support one or more plugin formats, providing flexibility by enabling the use of third-party effects and instruments. For the developed plugin, the main requirements are a graphical user interface, support on all major platforms, multichannel capabilities, and a free licensing model. VST 3[4] and CLAP[5] have been considered in this context, but since the JUCE framework does not natively support CLAP, the plugin uses VST 3.

To use Ambisonics, the setup has to support multichannel layouts, which means the DAW must be able to process multiple input and output sample streams simultaneously. However, many DAWs

[4]https://steinberg.net/developers

[5]https://cleveraudio.org



(a) (b)

Figure 5: *Benchmark results of (a) 3rd order Ambisonics for the frequency-domain (K = 7, O = 128 and N = 256) and the time-domain (naive wavetable) with the deadline indicating a buffer size of 64 samples and f_s = 48000 Hz. The minimum and maximum values are shown only for the frequency-domain due to transparency. (b) Comparison of different Ambisonics orders for the frequency-domain (K = 7, O = 128 and N = 1024) and a buffer size of 256 samples.*

do not support proper multichannel workflow. Reaper[6] includes extensive multichannel support but does not provide extended parameter modulation capabilities easily. Ableton Live[7] has no native support for multichannel plugins, but it integrates Max[8] as Max4Live, allowing the integration of multichannel plugins. The only drawback is that the GUI of the plugins cannot be directly accessed anymore, and must be controlled through Max4Live parameters. However, comprehensive parameter modulation is a preferable feature for the user study.

Since Abisonics decoding is not integrated into the plugin, an external decoder has to be inserted after the synthesis plugin. At this stage, the AllRADecoder[9] of the IEM plugin suite is used with 3rd order Ambisonics.

4. USER STUDY

A user study was conducted to investigate the perception of the source spread in relation to the localizability. User feedback and open responses were collected to evaluate additional and general aspects of the instrument.

4.1. Setup and Procedure

A total of 12 participants with a mean age of 27.25 years (SD = 3.6 years), took part in the study. They were seated at a desk in the center of a 21 channel loudspeaker system in a dome configuration.

The developed synthesizer plugin (Blackman-Harris four-term window with $K = 7$ and $O = 128$) and the AllRADecoder were integrated into a Max4Live instrument running in Ableton Live 11 on Windows 10. The Ableton Push Controller was used in the study, and parameters were automated by Ableton tools like LFO and Shaper. As the plugin is integrated in Max4Live, only essential parameters were made accessible through the GUI. Control over Ambisonics order and normalization, the gain envelope, and selection of the displacement function (fixed to sine type) were non-adjustable and hidden from the user.

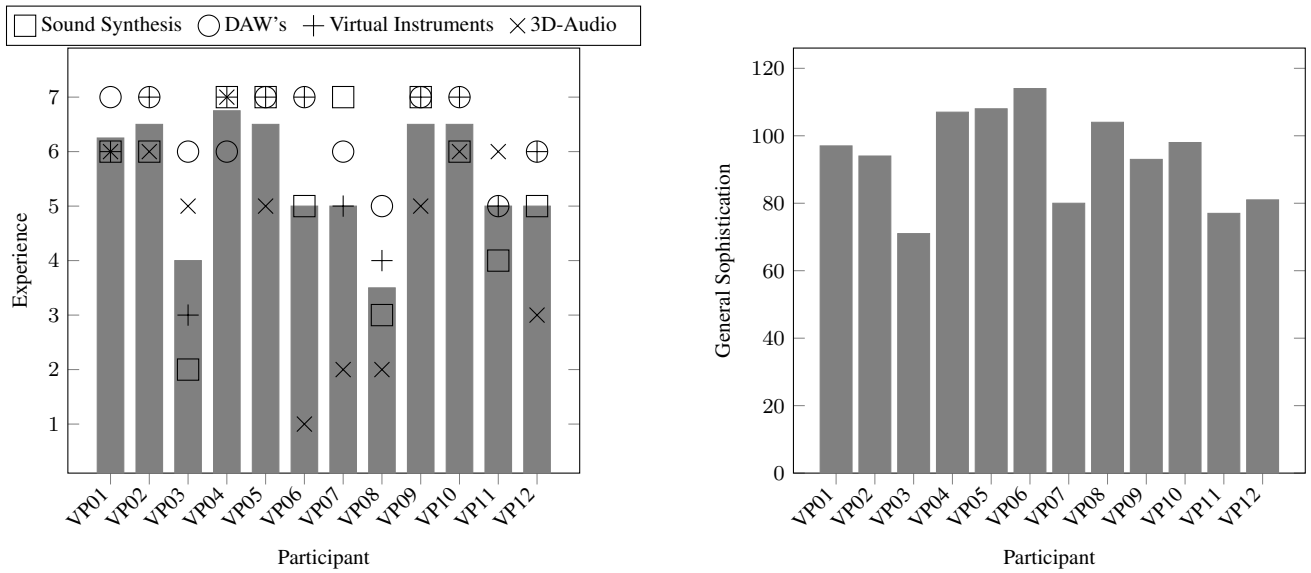
In the first part of the study, test subjects used the Ableton Push controller to manually adjust the spatial and timbral parameters. In the second part, parameters were automated by envelopes and low-frequency oscillators. Finally, the test subjects were able to give open feedback. The subjects' experience was evaluated using a seven-point balanced Likert scale ranging from "Completely Disagree" (1) to "Completely Agree" (7) and the general sophistication of the Gold-MSI [24, 25]. The parameter settings were assessed on a five-point Likert scale, and the perceptual qualities were named based on [26]. Higher values stand for a greater development of the corresponding quality of perception.

4.2. Results

The results of the users' expertise are shown in Figure 6. Figure 6(a) shows the self-reported experience regarding sound synthesis, DAWs, virtual instruments, and 3D-audio. The scores of the general sophistication of the Gold-MSI Figure are displayed in Figure 6(b).

Figure 7 presents the results for the auditory qualities that were examined in relation to spatial expansion. A paired t-test shows a statistically significant difference in the localizability between the conditions with and without spatial spread of the sound source (t(11) = 2.14, p = .028, one-tailed). Specifically, the mean localizability score was higher without spatial extension (M = 4.25, SD = 0.7) than with spatial extension (M = 3.5, SD = 1.1). The continuous change of the horizontal and vertical dispersion were examined for roughness (M = 2.83, SD = 0.99) and degree-of-liking (M = 3.67, SD = 0.75).

[6] https://www.reaper.fm
[7] https://ableton.com/live
[8] https://cycling74.com/products/max
[9] https://plugins.iem.at



(a) (b)

Figure 6: *Experience of the participants of the user study. (a) Self-reported expertise of specialized topics where the gray bar indicates an average value and (b) determination of the general sophistication of the Gold-MSI with a maximum attainable test score of 126.*

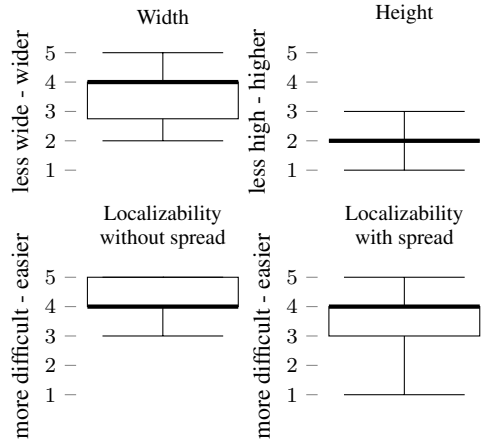


Figure 7: *Results of the investigated perceptual qualities in relation to spatial expansion.*

In the open responses it was noted that sounds were more difficult to localize in the vertical plane, and the Height parameter was not perceived as particularly influential by some subjects. A test subject explicitly mentioned a phaser effect occurring with extreme parameter modulations. Another test subject described the modulated expansion of the signal as more of a tone coloring and only to a limited extent as a spatial expansion. The changing overtone distribution emanating from the displacement function, depending on the number of partials, was explicitly rated as good by one test subject.

4.3. Discussion

Although sine waves are in general harder to localize, the results show an influence of angles and spread on the perception of the synthesized sound. Figure 7 and the t-test indicate that the spatial expansion of the signal works to a certain extend. According to the results, sources with increased spread are harder to localize. This holds true for most real or virtual sound sources and validates the source widening effect in the proposed approach.

The weaker perception of height extension, compared to width, has also been noted by some of the test subjects. This can be attributed to the loudspeaker setup, which is more sparse along the elevation axis. In addition, the human auditory system has a higher resolution for the azimuth than for the elevation of sound source positions.

The temporal change of the spread S can provoke a flanger-like effect. While the resulting tone-coloring effect is perceived as rough, the roughness is not necessarily perceived as unpleasant. In conclusion, the resulting effect has timbral qualities, which may not be achieved without the spatial modulation. Considering one isolated Ambisonics channel, moving notches in the amplitude spectrum occur which can be compared to a moving comb filter. Partials that are present in one channel at a point in time, are attenuated in other channels. Thus, the corresponding frequency modulated comb filters follow the same frequency but with temporal shifts. The flavour of the tonal coloration varies with the type of the displacement function.

5. CONCLUSION

The presented synthesizer plugin makes IFFT-based additive synthesis in the Ambisonics domain available in conventional music production workflows on the DAW. The chosen approach demonstrates a sufficient SNR and the performance analysis reveals an

expected advantage over additive synthesis in the time-domain. Furthermore, the IFFT does not represent a bottleneck with respect to the amount of calculations required for each channel of HOA. Therefore, signals with a high overtone density can also be efficiently synthesized for higher Ambisonics orders.

The user study results indicate that the chosen method for spatial distribution can be used to influence source position and width, as well as tone coloration. For a more detailed investigation, the experimental design needs to be changed and more test users should be included.

The integration of multichannel capabilities is still in its early stages in many DAWs, which makes it difficult to use plugins for Ambisonics and related technologies without additional customization. Considering the rise of spatial audio production techniques in music production, movie sound, extended reality (XR) and video games, this problem is likely to disappear in the near future. Synthesizers with an increased focus on spatial abilities are thus predestined to become a standard in these domains. Future work will focus on other waveforms than the basic ones used in this version. This will also include partial trajectories from previously analyzed recordings of musical instruments.

6. REFERENCES

[1] A. McLeran, C. Roads, B. L Sturm, and J. J Shynk, "Granular sound spatialization using dictionary-based methods," in *Proc. 5th Sound and Music Computing Conf. (SMC)*, 2008.

[2] A. Müller and R. Rabenstein, "Physical modeling for spatial sound synthesis," in *Proc. Digital Audio Effects (DAFx-09)*, 2009.

[3] R. McGee, "Spatial modulation synthesis," in *Proc. Int. Computer Music Conference (ICMC)*, 2015.

[4] D. Topper, M. Burtner, and S. Serafin, "Spatio-operational spectral (SOS) synthesis," in *Proc. Digital Audio Effects (DAFx-02)*, Singapore, 2002.

[5] H. von Coler, *A System for Expressive Spectro-spatial Sound Synthesis*, PhD Thesis, Technische Universität Berlin, Berlin, 2021.

[6] C. Verron, M. Aramaki, R. Kronland-Martinet, and G. Pallone, "A spatialized additive synthesizer," in *Proc. Inaugural Int. Conf. Music Commun. Science (ICoMCS)*, Sydney, Australia, Dec. 5-7, 2007.

[7] C. Verron, M. Aramaki, R. Kronland-Martinet, and G. Pallone, "A 3-D immersive synthesizer for environmental sounds," *IEEE Trans. Audio, Speech and Language Processing*, vol. 18, no. 6, pp. 1550–1561, Aug. 2010.

[8] H. A. Chamberlain, "Experimental fourier series universal tone generator," *J. Audio Engineering Society*, vol. 24, no. 4, pp. 271–276, May 1976.

[9] X. Rodet and P. Depalle, "Spectral envelopes and inverse FFT synthesis," in *Proc. 93rd AES Conv.*, San Francisco, CA, US, Oct. 1-4, 1992.

[10] M. Goodwin and A. Kogon, "Overlap-add synthesis of nonstationary sinusoids," in *Proc. Int. Computer Music Conf.*, Banff, Alta., Canada, 1995.

[11] M. Goodwin and X. Rodet, "Efficient Fourier synthesis of nonstationary sinusoids," in *Proc. Int. Computer Music Conf.*, San Francisco, CA, 1994.

[12] J. Laroche, "Synthesis of sinusoids via non-overlapping inverse Fourier transform," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 4, pp. 471–477, Jul. 2000.

[13] R. Kutil, "Optimized sinusoid synthesis via inverse truncated Fourier transform," *IEEE Trans. Audio, Speech and Language Processing*, vol. 17, no. 2, pp. 221–230, Feb. 2009.

[14] X. Wen and M. Sandler, "Fast additive sinusoidal synthesis with a subband sinusoidal method," *IEEE Signal Processing Letters*, vol. 20, no. 5, pp. 467–470, May 2013.

[15] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Pearson Higher Education, Inc., Upper Saddle River, NJ, US, third edition, 2010.

[16] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proc. IEEE*, vol. 66, no. 1, pp. 51–83, Jan. 1978.

[17] M. A. Gerzon, "Periphony: With-height sound reproduction," *J. Audio Eng. Soc*, vol. 21, no. 1, pp. 2–10, 1973.

[18] P. Fellgett, "Ambisonic reproduction of directionality in surround-sound systems," *Nature*, vol. 252, no. 5484, pp. 534–538, 1974.

[19] J. Daniel, R. Nicol, and S. Moreau, "Further investigations of high-order ambisonics and wavefield synthesis for holophonic sound imaging," in *Proc. 114th AES Conv.*, Amsterdam, Netherlands, Mar. 22-25, 2003.

[20] T. Carpentier, "Normalization schemes in ambisonic: Does it matter?," in *Proc. 142th AES Conv.*, Berlin, Germany, May 20-23, 2017.

[21] C. Nachbar, F. Zotter, E. Deleflie, and A. Sontacchi, "AmbiX – a suggested ambisonics format," in *Proc. Ambisonics Symp.*, Lexington, KY, Jun. 2-3, 2011.

[22] U. Zölzer, Ed., *DAFX: Digital audio effects*, chapter Spectral processing, p. 408, J. Wiley & Sons, The Atrium, Southern Gate, Chichester PO19 8SQ, UK, second edition, 2011.

[23] Adrian Freed, Xavier Rodet, and Philippe Depalle, "Synthesis and control of hundreds of sinusoidal partials on a desktop computer without custom hardware," in *ICSPAT (International Conference on Signal Processing Applications & Technology*, 1992.

[24] D. Müllensiefen, B. Gingras, J. Musil, and L. Stewart, "Measuring the facets of musicality: The goldsmiths musical sophistication index (Gold-MSI)," *Personality and Individual Differences*, vol. 60, pp. S35, 2014.

[25] N. K. Schaal, A. R. Bauer, and D. Müllensiefen, "Der Gold-MSI: Replikation und Validierung eines Fragebogeninstrumentes zur Messung musikalischer Erfahrenheit anhand einer deutschen Stichprobe," *Musicae Scientiae*, vol. 18, no. 4, pp. 423–447, 2014.

[26] A. Lindau, V. Erbes, S. Lepa, H. Maempel, F. Brinkmann, and S. Weinzierl, "A spatial audio quality inventory for virtual acoustic environments (SAQI)," *Acta Acustica united with Acustica*, vol. 100, no. 5, pp. 984–994, Sep./Oct. 2014.

THE THRESHOLD OF PERCEPTUAL SIGNIFICANCE FOR TV SOUNDTRACKS

Robert J. Acheson and Trevor R. Agus

SARC
Queen's University, Belfast
Belfast, Northern Ireland
racheson04@qub.ac.uk | t.agus@qub.ac.uk

ABSTRACT

Hearing loss affects 1.5 billion people world-wide [1], affecting many aspects of life, including the ability to hear the television. Simply increasing the volume may restore audibility of the quietest elements, but at a cost of making other elements undesirably loud. Therefore, at the very least, dynamic range compression could also be useful, fitted to an individual's frequency-dependent hearing loss. However, it is not clear whether the audibility of the quietest parts of TV audio needs to be preserved. This experiment aims to measure which elements of the audio are important by presenting normal-hearing listeners with binary masked versions of TV audio presented at 60 dB(A), muting audio below a given sensation level. It was hypothesised that spectro-temporal regions with the most power density would dominate perception, such that the less active regions may not be missed. To find this threshold of perceptual significance, a two-alternative forced choice signal detection experiment was designed in which excerpts from BBC television shows were binary masked and presented to the participants, with the task to identify which clips sounded more processed. The results suggest that discarding audio below 10 phons would rarely be noticed by most listeners.

1. INTRODUCTION

This paper is part of a larger project which aims to improve the listening experience for people with hearing loss when watching the television. Hearing loss affects one in five people in the UK [2]. The most widely adopted device for improving hearing is the behind-the-ear (BTE) hearing aid [3] however, it has inherent limitations since it must work in real time with a low power computer. Another approach to improving audibility is the use of clean audio for multimedia [4] i.e., audio which has all sounds except dialogue attenuated. However, most clean audio solutions consist of algorithms used to remove or attenuate background noise—including music and sound effects, which are fundamental elements of television soundtracks which contribute to the narrative of television programmes.

1.1. Hearing Aids

There are two main categories of hearing loss: conductive and sensorineural [5]. Both hearing losses cause a raised threshold of audibility. In order to restore audibility of quiet sounds, they must be amplified, although loud sounds cannot be amplified as this would cause discomfort for the person with hearing loss. Hearing aids provide multi-band dynamic-range compression in order to restore audibility in the affected frequency ranges to improve audibility, without amplifying sounds beyond the level of discomfort [6]. Due to their small size, only a small battery can be fitted to hearing aids. This means the potential computational power of the hearing aid is limited, in order to preserve battery life. The hearing aid is also required to work in real time; due to its low computational power and impossibility of a "look ahead" feature, i.e., the ability to view the oncoming amplitude envelope, the juxtaposition of quiet and loud sounds can often impair audibility due to over-correction, often referred to as overshoot [7]. For example, if someone were to clap their hands, the hearing aid would attenuate the frequency ranges in which the clap is present in order to prevent damage to the user's ears. However, the slow release-time of the hearing aid means that any conversation following the clap may also be attenuated for a brief time.

Hearing aids are limited in ways which affect both the frequency domain and the temporal domain. The frequency bands on which the multi-channel dynamic-range compression is applied are broad, resulting in low frequency resolution. Temporal resolution limitations are caused by the necessity of real-time operation in the form of latency. With a view to improve on the hearing aid for pre-recorded entertainment purposes, the audio can be processed offline. This means that the frequency and temporal resolutions can be greatly improved upon, and the main limitation is computation time. However, considering that the audio is pre-recorded, computation time is less of a pressing issue.

2. PROPOSED SOLUTION

The proposed offline process is based on Ray Meddis' BioAid algorithm, which is used in [8]. BioAid is similar to the proposed solution in that it uses a filter bank at the beginning and end of the signal chain and is psychoacoustically inspired, however the filter banks used in the proposed solution are comprised of auditory gammatone filters rather than Butterworth filters. The proposed solution to be used in this project takes the form of a program designed in MATLAB and consists of two main parts: the analysis tool and the adaption tool, see figure 1. The analysis tool is used to examine the incoming audio and compare the amplitudes of frequency bands to a predefined threshold. It can then generate an adapted amplitude envelope for each frequency band based on the input amplitude. The adaption tool uses the amplitude envelopes generated by the analysis tool to adjust the amplitude envelopes of the frequency bands. Presumably, the audio broadcast on television or streaming services has been mixed such that the important elements are audible to people with normal hearing. Furthermore, if a person with normal hearing cannot tell when some of the au-

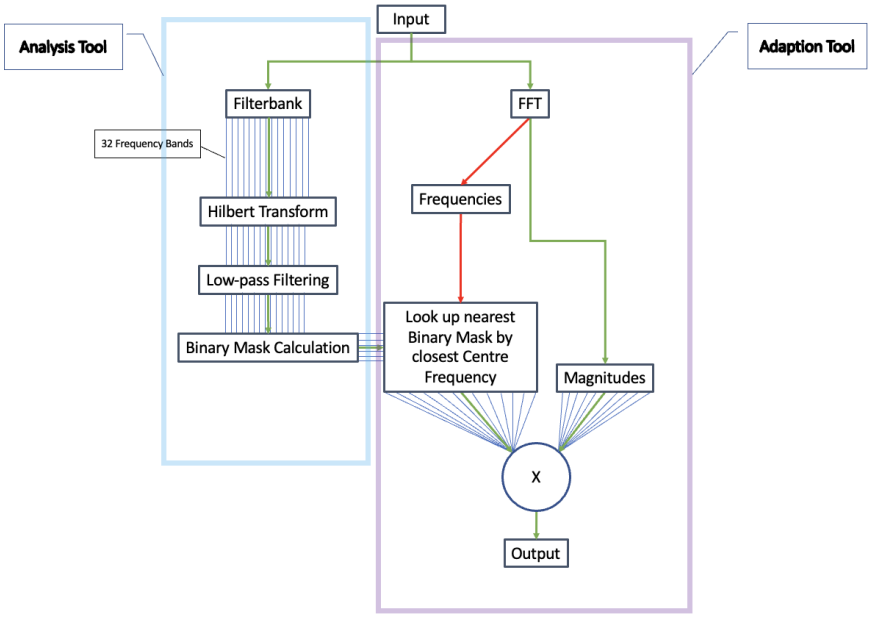


Figure 1: *Signal Flow Chart for the Analysis and Adaption Tool, see sections 2.1-2.3.*

dio has been removed, then it could be deemed unimportant to the listening experience. Using this logic, the important elements of the audio can be identified by muting parts of the dynamic range and testing to see if normal hearing people can hear a difference.

2.1. Analysis Tool

To analyse the audio from "the ear's perspective", we use a gammatone auditory filterbank. The analysis tool takes an audio file as the input and filters it into frequency bands using the gammatone filterbank. This auditory filter bank is comprised of n band-pass gammatone filters, which are modelled on the filters found in the human cochlea [9]. This means that the audio is separated in the program with similar frequency resolution to the cochlea. After filtering the audio into frequency bands, the amplitude envelope of each frequency band is calculated in order to compare it to the given threshold.

2.1.1. Filter Bank

The gammatone filters used in the filterbank are modelled on the human cochlea, designed by Hohmann [9] and implemented using the Auditory Modelling Tool- box 1.2 [10]. The bandwidths of these filters are based on experimental data (Equivalent Rectangular Bandwidth, ERBs). These bandwidths are closely related to the critical bandwidth of the auditory filters found in the human cochlea and were measured using the notched noise method [11]. The notched noise method involves measuring the audibility threshold of a signal in a masker, where the signal is a fixed frequency sine tone, and the masker is noise with a notch centred on the signal frequency. The ERB of a filter can approximated with the following formula, see equation 1 [12]. To avoid phase delays and group delays, the audio is filtered twice, the second filtering being time reversed such that any phase or group delays caused by the filters are corrected. To maintain the bandwidths of the original filterbank, the bandwidths of the original filters are doubled.

$$ERB(f) = 0.108f + 24.7 \tag{1}$$

The filterbank takes a mono input and has parallel outputs, one for each of n filters. Each of the frequency bands can be denoted by the centre frequency (f_c) of the gammatone filter used to create them.

2.1.2. Amplitude Envelope Measurement

The amplitude envelope of each frequency band is calculated to provide a means of comparing the input signal to the normal hearing (NH) audibility thresholds. The amplitude envelope of each channel is measured here using the Hilbert transform. This is an efficient way to measure the amplitude envelope for signals resembling sinusoids as used in [13], [14], and [15].

2.1.3. Low Pass Filtering

The fastest amplitude modulations from the output of the Hilbert transform were removed by a low-pass filter. As seen in Figure 2, the magnitude of the Hilbert transform contains the overall amplitude envelope of the input signal and a rectified version of the temporal fine structure. The low pass filter reduces all frequencies over half the bandwidth of the frequency band leaving a smoother amplitude envelope for the fitting algorithm to compare the input signal to the given threshold.

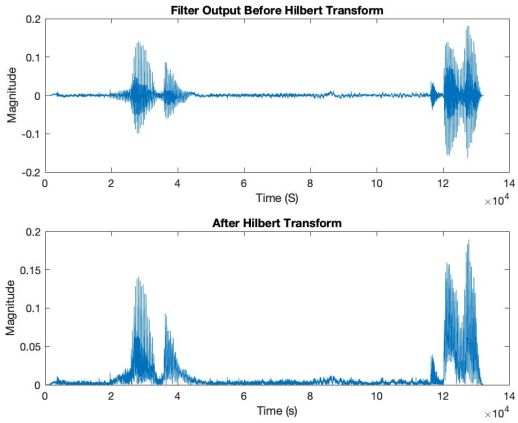


Figure 2: *Amplitude Rectification Using Hilbert Transform*

2.2. Binary Mask

To test what of the audio above the threshold of audibility is perceived by a person with normal hearing, the smooth binary algorithm was created. Inspired by binary masking used in image processing, as used in [16], the objective is to superimpose a mask of ones and zeros onto the audio such that each amplitude below a given threshold is assigned the value zero and each amplitude above the threshold is assigned the value one. Each amplitude with an assigned value of zero can then be muted. By moving this threshold and testing whether a perceptual difference has been detected, the "threshold of perceptual significance" (TPS) can be measured. This TPS is the point above the threshold of audibility at which changes to the audio are perceptually significant. The audio below the measured TPS can be permanently muted resulting in faster computation times without changing the perception of the audio in its original form.

Some considerations taken when designing the binary masks were the naturally fast amplitude modulations, measurement of the "threshold of perceptual significance", and ensuring no latency is present.

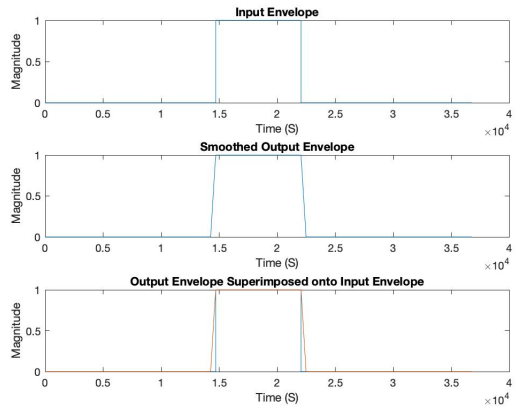


Figure 3: *Plots Showing Amplitude Ramping to Smooth Binary Mask Function*

The binary mask has fast amplitude modulation by nature; however, this leads to the creation of artifacts which are not desirable when processing audio. To remedy this, linear amplitude ramping was employed to smooth the binary mask envelope and prevent extremely fast changes in amplitude (see Figure 3). The ramping is configured in such a way that should an attack and release ramp overlap, the maximum of the two ramps is chosen (see Figure 4).

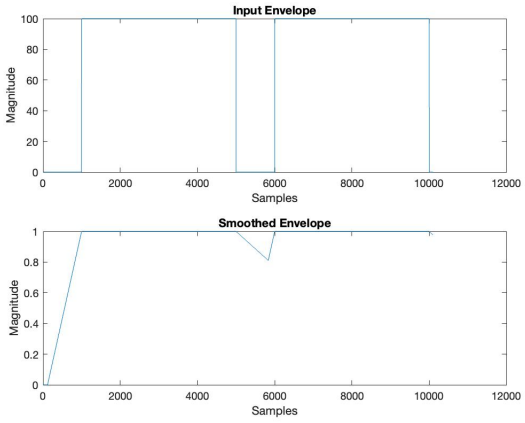


Figure 4: *Binary Mask Ramps with Overlapping Release and Attack*

Due to the fact that humans don't perceive loudness on a linear scale, the binary mask is measured in phons. This means that the threshold for the binary mask algorithm will be situated at the same perceptual loudness level across all frequencies using the ISO 226 Equal-Loudness-Level Contour model by Jeff Tackett [17].

2.3. Adaption Tool

The adaption tool performs a Fast Fourier Transform (FFT) on the input audio file to break it up into its comprising sine tones which can be described in terms of frequency, magnitude, and phase. It then performs a "Slow Inverse Fourier Transform" (SIFT). This SIFT re-synthesises the audio by synthesising each of the comprising sine tones individually, allowing for fine amplitude control at each frequency over time. This fine control is achieved because each sine wave can be discretely amplitude modulated up to speeds matching the sample rate of the audio. The binary mask calculated previously is used to alter the amplitude of the input audio frequency components. It should be noted that to maintain intelligibility of speech, the amplitude modulation should not be excessively fast such that the speech amplitude envelopes are distorted. Similarly, large changes should not be made to consecutive frequencies to prevent artefacts such as ringing. The formula for the SIFT is as seen in Equation 2:

$$y = \frac{1}{N}\sum_{n=0}^{N} x[f_c, n]e^{-2\pi jfn}.g[f_c, n] \tag{2}$$

where g is the binary mask, N is the total number of samples, x is the input sample, and y is the output sample.

3. METHODS

3.1. Participants

Participants for the experiment were recruited through email. Five participants with normal hearing were recruited (three male and two female), aged from 20–24, with a mean age of 22. To confirm their normal hearing, an audiometer was used to test 250 Hz–8 kHz down to 10 phons.

3.2. Comparison Procedure

For the duration of the experiment, the participant was seated in a sound-treated booth with a window. A monitor was placed outside the window, visible to the participant. The participants were given verbal instruction before the trials. The experiment took the form of a series of two-alternate forced choice trials. In each trial, the participant was presented two clips of audio through a pair of Sennheiser HD600 headphones, each lasting three seconds: one with binary mask processing, which will be referred to as processed; and one without binary mask processing, which will be referred to as unprocessed. The duration of the onset and offset smoothing ramps for the binary were set to 10ms and 50ms respectively. Both signals were low-pass filtered as the ISO 226 model used does not support thresholds above 12.5kHz. The processed and unprocessed clips presented in each trial were not necessarily from the same source. The order of the processed and unprocessed clips for each trial was randomised. The processed audio clips were selected randomly from a pool of audio clips for a given threshold measured in phons. The thresholds were spaced equally in intervals of five phons (5–35 phons). The participant was asked to select which of the two audio clips sounded more processed using '[1]'and '[2]'on a computer keyboard to select the first and second audio clips respectively. Their response to each trial was recorded.

3.3. Stimuli

The audio was sourced from BBC iPlayer. Seven television show genres were chosen from the iPlayer menu (Drama, News, Music, Documentary, Sport, Comedy, and Entertainment) and two television programs were picked at random from each genre using a random number generator (with the exception of Drama, for which only one was chosen). Three audio clips were recorded from each television show with randomised start times, each lasting five minutes. The audio used in the experiment was generated using these five-minute-long excerpts. For each of the excerpts a clip of three seconds in length was extracted with a randomised start time. This was repeated for each threshold level in phons to provide a bank of 39 audio clips per threshold level.

4. RESULTS

Figure 5 shows the averaged participant responses fitted to a cumulative normal distribution model. As hypothesised, the TPS is above the auditory thresholds, at approximately 17.1 phons based on a 75% threshold. The lowest level of performance at 51.7% indicates little noticeable difference between the processed and unprocessed clips. The peak performance at 92% indicates an obvious difference between the processed and unprocessed clips.

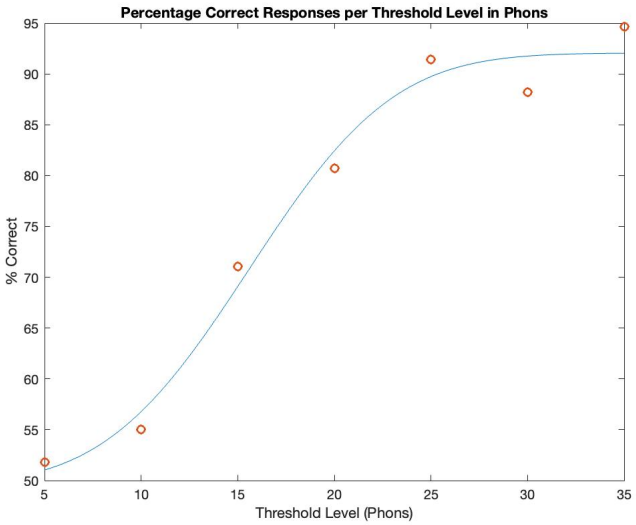


Figure 5: *The average accuracy of responses across all participants (red circles) and a cumulative normal curve fitted by the least-squares method (blue line).*

5. DISCUSSION

The results are consistent with the hypothesis that the spectro-temporal regions with less power can often be removed with little detriment to the listening experience. This suggests that the whole of the dynamic range does not need to be preserved at all times. Should this be the case, dynamic range compression algorithms used to adapt audio for people with hearing loss are not required to be as aggressive, which in turn reduces the risk of introducing audible artifacts.

In the current implementation of this experiment, a presentation level of 60 dB(A) was used, which returned a TPS of 17.1 phons however, it is not clear whether varying the presentation level would change the TPS. This could be investigated directly in a future experiment. Measuring the TPS for a range of presentation levels would potentially allow for automation of the binary mask threshold based on the presentation level, i.e. the TV volume in a real-world scenario.

When watching TV in real-world conditions, there would be more background noise and other distractions present compared to the experiment conditions. This could mean that the TPS may, in fact, be higher than shown in the results for real-world scenarios. With the absence of this background noise, it is possible that the participants are able to identify the processed clips due to the addition of silence as opposed to the absence of important elements of audio.

6. CONCLUSION

The focus of this experiment was to identify the key elements to people with normal hearing of television audio by muting parts of the audio hypothesised to be unimportant to the listening experience. The results show that at a presentation level of 60 dB(A), most audio under 10 phons could potentially be muted without affecting the listening experience. This means that not all of the

dynamic range needs to be preserved at all times, allowing for less aggressive dynamic range compression algorithms, and therefore a reduced risk of introducing audible artifacts. Potential areas of future research have been highlighted in section 5 to improve the binary mask algorithm's flexibility in terms of presentation level.

7. REFERENCES

[1] World Health Organization, "Deafness and hearing loss," Available at https://www.who.int/health-topics/hearing-loss.

[2] RNID, "Facts and figures," Available at https://rnid.org.uk/about-us/research-and-policy/facts-and-figures/.

[3] NHS, "Hearing aids and implants," Available at https://www.nhs.uk/live-well/healthy-body/hearing-aids/.

[4] B. Shirley and P. Kendrick, "The clean audio project: Digital TV as assistive technology," 2006.

[5] Starkey, "Types and causes of hearing loss," Available at https://www.starkey.com/hearing-loss/types-and-causes.

[6] K. Patel and Issa M. S. Panahi, "Frequency-based multi-band adaptive compression for hearing aid application," *The Journal of the Acoustic Society*, vol. 146, no. 4, pp. 2959–2959, 2019, Available at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7380331/.

[7] Brian C. J. Moore, "Perceptual consequences of cochlear hearing loss and their implications for the design of hearing aids," *Ear and Hearing*, vol. 17, no. 2, pp. 133–161, 1996.

[8] R. Meddis, W. Lecluyse, N. R. Clarke, and T. Jurgens, "Exploration of a physiologically-inspired hearing- aid algorithm using a computer model mimicking impaired hearing," *International Journal of Audiology*, vol. 55, pp. 346–357, 2016.

[9] V. Hohmann, "Frequency analysis and synthesis using a gammatone filterbank," *Acta Acust.*, vol. 88, pp. 433–442, 2002.

[10] P. Majdak, C. Hollomey, and Robert Baumgartner, "Amt 1.2: A toolbox for reproducible research in auditory modeling," *Acta Acust.*, vol. 6, pp. 19, 2022.

[11] B. C. J. Moore and B. R. Glasberg, "Derivation of auditory filter shapes from notched-noise data," *Hearing Research*, vol. 47, no. 1–2, pp. 103–188, 1990.

[12] Julius O. Smith and Jonathan S. Abel, "Equivalent rectangular bandwidth," Available at https://ccrma.stanford.edu/ jos/bbt/Equivalent$_Rectangular_Bandwidth.html$.

[13] C. Lorenzi, G. Gilbert, H. Carn, S. Garnier, and B. C. J. Moore, "Speech perception problems of the hearing impaired reflect inability to use temporal fine structure," Available at https://www.pnas.org/doi/pdf/10.1073/pnas.0607364103.

[14] G. Gilbert, M. A. Akeroyd, and S. Gatehouse, "Discrimination of release time constants in hearing-aid compressors," *International Journal of Audiology*, vol. 47, no. 4, pp. 189–198, 2008.

[15] J. C. Ziegler, C. Pech-Georgel, F. George, and C. Lorenzi, "Speech-perception-in-noise deficits in dyslexia," *Developmental Science*, vol. 12, no. 5, pp. 732–745, 2009.

[16] Jinyang Liang, Sih-Ying Wu, Rudolph N Kohn Jr, Michael F Becker, and Daniel J Heinzen, "Grayscale laser image formation using a programmable binary mask," *Optical Engineering*, vol. 51, no. 10, pp. 108201–108201, 2012.

[17] Jeff Tackett, "Iso 226 equal-loudness-level contour signal," Available at https://uk.mathworks.com/matlabcentral/fileexchange/7028-iso-226-equal-loudness-level-contour-signal.

AN ACTIVE LEARNING PROCEDURE FOR THE INTERAURAL TIME DIFFERENCE DISCRIMINATION THRESHOLD

Andrea Gulli, Federico Fontana
Department of Mathematics, Computer Science and Physics
Università di Udine, Udine, Italy
gulli.andrea@spes.uniud.it

Stefania Serafin
Department of Architecture, Design, and Media Technology
Aalborg University, Copenhagen, Denmark
sts@create.aau.dk

Michele Geronazzo
Department of Engineering and Management
Università di Padova, Padua, Italy
michele.geronazzo@unipd.it

ABSTRACT

Measuring the auditory lateralization elicited by interaural time difference (ITD) cues involves the estimation of a psychometric function (PF). The shape of this function usually follows from the analysis of the subjective data and models the probability of correctly localizing the angular position of a sound source. The present study describes and evaluates a procedure for progressively fitting a PF, using Gaussian process classification of the subjective responses produced during a binary decision experiment. The process refines adaptively an approximated PF, following Bayesian inference. At each trial, it suggests the most informative auditory stimulus for function refinement according to Bayesian active learning by disagreement (BALD) mutual information. In this paper, the procedure was modified to accommodate two-alternative forced choice (2AFC) experimental methods and then was compared with a standard adaptive "three-down, one-up" staircase procedure. Our process approximates the average threshold ITD 79.4% correct level of lateralization with a mean accuracy increase of 8.9% over the Weibull function fitted on the data of the same test. The final accuracy for the Just Noticeable Difference (JND) in ITD is achieved with only 37.6% of the trials needed by a standard lateralization test.

1. INTRODUCTION

The ability to localize sound sources is of considerable importance for humans and animals; it determines the direction of objects to be sought or avoided and the appropriate direction to direct visual attention. Although auditory localization may rely on the sound arriving at one ear, the most reliable localization cues depend on the acoustic waves arriving at both ears [1]. The difference between the two paths from a sound source to the ears creates an interaural time difference (ITD). In parallel, an interaural level difference (ILD) occurs due to the head shadow on the contralateral ear. In humans, the cue that enables sound localization most accurately (up to 1 degree in azimuth) is the ITD [2]. Experiments capable of isolating ITD used pairs of "on the ear" stimulators, namely headphones. Early headphone-based tests reported ITD detection thresholds at microsecond scale, that is, orders of magnitude smaller than all other sensory modalities were able to detect [3].

When headphones are worn, the sound source image is localized inside the head. The term "lateralization" was then adopted to describe the apparent sound source position inside the head. On the other hand, headphones allow for precise control of interaural differences and do not generate room echoes. Therefore, lateralization were preferred to localization tests when studying sound source perception inside the laboratory [4]. Since the 1950s, accurate studies have been systematically conducted to determine ITD thresholds [5]. The lowest were reported to be close to 10 μs. However, the participants' hearing and training level necessary to achieve those thresholds have become clear only recently, along with the stimuli and measurement technique required for measuring them [6]. Specifically, the stimulus that produced the lowest ITD threshold was Gaussian noise, bandpass filtered from 20 to 1400 Hz and presented at a sound pressure level of 70 dB. The most accurate method was a two-interval procedure with an interstimulus interval of 50 ms. The mean ITD threshold in this condition at the 75% corrected level was 6.9 μs for trained listeners, and 18.1 μs for untrained listeners. However, other studies report higher ITD values as *normal* hearing thresholds, i.e., with mean equal to 263 μs and standard deviation equal to 112 μs [7].

We present an accelerated procedure for reliably determining individual lateralization thresholds, and compare it to standard approaches to subjective ITD measurement. We will focus on untrained participants with the goal of significantly shortening the test sessions. This feature would be desirable especially when specific groups of users are targeted, such as young patients whose binaural acuity needs to be tested. Moreover, the same procedure can quickly calibrate and individualize immersive audio technologies for the most diverse applications and virtual environments [8].

1.1. Psychometric function estimation

A psychometric function (PF) maps the subjective performance during a perceptual task against a stimulus magnitude, such as brightness or other intensity levels. Performance is measured as the percentage of correct responses, or responses where the participant was able to detect the stimulus. Ideally, a PF is estimated at informative sample points on a continuous scale. The level set estimation (LSE) problem consists of identifying the regions where an initially unknown PF $\mathrm{f}(x)$ lies above or below a particular threshold ϑ. In general, a level set S is the set on which f exceeds some critical value (e.g., $S = x : \mathrm{f}(x) > \vartheta$). Efficient LSE is an active learning problem [9], involving techniques that use surrogate models to perform active sampling [10]. The active learning configuration consists of the definition of an acquisition function that classifies the data points to be labeled according to the current state of the model and a hand-designed information measure to be maximized [11]. In Bayesian active-learning (BAL), the basic idea is to define a statistical model and then tune its parameters in due data

collection. Typically, the model is initialized with a weakly informative prior distribution which expresses the uncertainty about these parameters before the start of the experiment. Then, recorded data provide likelihood terms to be combined with the prior in a posterior distribution, reflecting the beliefs about the parameters from the data collected so far. The stimulus for every subsequent trial is selected so as to maximize some utility measure that is integrated with the current posterior. One of the first BAL procedures in psychophysics that used Bayesian principles for both modeling the response and choosing the parameters for the next trial [12] was designed to classify a subject into one of nine audiometric groups, and was then validated with numerical simulations. The stimulus for the subsequent trial was chosen to maximize the mutual information between the current and the following unknown estimate by selecting it with the least expected entropy [13]. Selection was made by computing the posterior probabilities across all candidate stimuli for the next trial.

A general BAL procedure for classification and preference tasks that uses Gaussian Processes (GP) [14] for estimating a subjective response is called Bayesian Active Learning by Disagreement (BALD) [15]. These are the approximation technique and the acquisition function employed in this study. GP-based Bayesian inference has been recently employed in machine learning applications across various disciplines [16], and specifically in audiology [17, 18]. GPs in fact incorporate prior hypotheses about the mean, the smoothness between class boundaries, and the covariance between data points. BALD active learning with GP classification (GPC) has already been used in auditory applications, e.g. for optimal setting of a hearing aid [19], and for determining audiograms [20, 21], equal-loudness contours [22], and psychometric functions [23]. However, it has never been employed in the measurement of ITD thresholds.

This paper presents: Sec. 2 the mathematical background of GP and BALD classification; Sec. 3 the characteristics of the specific model; Sec. 4 the test determining individual lateralization thresholds with BALD; Sec. 5 the results, and Sec. 6 their discussion. Sec. 7 concludes the paper. As we will see from our results, GPC with active learning is a valid approximation for the PF, with a RMSE computed on the whole test set which is smaller than 10% concerning the conventional Weibull fitting [24]; it achieves similar performances as the default procedure (mean error equal to 5.1 μs) by requiring only 37.6% of the trials otherwise needed by the standard procedure.

2. THEORETICAL BAKGROUND

2.1. Gaussian Processes Classification

Let $\mathrm{f} : \mathbb{R} \rightarrow \mathbb{R}$ be a latent function on an arbitrary input space $\mathbf{X}$. A GP is a convenient technique for encoding prior knowledge about f that can be later updated via Bayesian inference in light of the observed data. A GP is a collection of random variables, any finite subset of which jointly forms a Gaussian distribution. Therefore, a GP is a particular case of a stochastic process. Like the multivariate Gaussian distribution, a GP is completely specified by its first two moments: a mean function $\mu(\mathbf{x})$ and a positive semidefinite covariance function $K(\mathbf{x}, \mathbf{x}')$. The mean function expresses the central tendency of the latent function, while the covariance function accounts for its correlation structure. Given μ and K, the latent function f can be endowed with a GP prior distribution

$$p(\mathrm{f}) = \mathcal{GP}\left(\mu(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')\right). \tag{1}$$

Given a GP prior on f and some observations over the input space, a prediction can be performed about the behavior of f for unobserved inputs using Bayesian inference, computed by Bayes' rule:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}.$$

According to Bayes' theorem, the joint posterior of the latent function at training and test inputs given the training observations is

$$p(\mathbf{f}, \mathrm{f}_*|\mathbf{X}, \mathbf{y}, x_*) = \frac{p(\mathbf{f}, \mathrm{f}_*|\mathbf{X}, x_*)p(\mathbf{y}|\mathbf{f})}{p(\mathbf{y}|\mathbf{X})}, \tag{2}$$

where $\mathrm{f}(x_*) = \mathrm{f}_*$, and $\mathbf{f} = \mathrm{f}(\mathbf{X})$. The predictive posterior distribution can be determined by marginalizing out the training set latent variables and substituting in (2),

$$\begin{aligned} p(\mathrm{f}_*|\mathbf{X}, \mathbf{y}, x_*) &= \int p(\mathbf{f}, \mathrm{f}_*|\mathbf{X}, \mathbf{y}, x_*)d\mathbf{f} \\ &= \frac{1}{p(\mathbf{y}|\mathbf{X})} \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}, \mathrm{f}_*|\mathbf{X}, x_*)d\mathbf{f}, \end{aligned} \tag{3}$$

and by definition of the GP, the joint probability $p(\mathbf{f}, \mathrm{f}_*|\mathbf{X}, x_*)$ is a multivariate Gaussian. With the posterior, we can compute a probabilistic prediction of the latent function f at the new input locations $\mathbf{X}_*$, taking into consideration the previously observed samples $(\mathbf{y}, \mathbf{X})$. The posterior mean and the posterior covariance on f provide information about the updated beliefs and the remaining uncertainty about the latent function. The likelihood $p(\mathbf{y}|\mathbf{f})$ describes the relationship between the latent function values $\mathbf{f}$ and the observations $\mathbf{y}$ at the training inputs $\mathbf{X}$.

The focus of this study is one-dimensional binary classification, where observed outputs can only assume two values: 1 (success) or 0 (failure). The latent function f is not directly observed but is instead a hidden function, where larger values of f generate higher probabilities of success. To obtain the probabilistic distribution $p(y = 1|\mathrm{f})$, f is "squashed" using a monotonically increasing sigmoid function Φ to the range [0,1]. For a binary observation y_i associated with an input $x_i \in \mathbf{X}$,

$$p(y_i = 1|\mathrm{f}) = \Phi(\mathrm{f}_i) = \Phi\left(\mathrm{f}(x_i)\right). \tag{4}$$

One largely used and convenient choice of Φ to deal with binary classification problems is the inverse-logit, also known as Bernoulli-logistic function likelihood, given by

$$\Phi(\mathrm{f}_i) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\mathrm{f}_i} e^{-\frac{t^2}{2}} dt, \tag{5}$$

and assuming that the labels $\mathbf{y} = (y_1, \ldots, y_N)$ of the N training data points are conditional independent if (latent) $\mathbf{f}$ are known,

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} p(y_i|\mathrm{f}(x_i)). \tag{6}$$

The predictive posterior distribution in (3) with (6) now becomes

$$p(\mathrm{f}_*|\mathbf{X}, \mathbf{y}, x_*) = \frac{1}{p(\mathbf{y}|\mathbf{X})} \int \prod_{i=1}^{N} \Phi(\mathrm{f}_i)p(\mathbf{f}, \mathrm{f}_*|\mathbf{X}, x_*)d\mathbf{f}. \tag{7}$$

From (7), the probabilistic prediction of class identity for a test observation y_* can be computed as:

$$\begin{aligned} p(y_* = 1|\mathbf{X}, \mathbf{y}, x_*) &= \int p(y_* = 1|\mathrm{f}_*)p(\mathrm{f}_*|\mathbf{X}, \mathbf{y}, x_*)d\mathrm{f}_* \\ &= \int \Phi(\mathrm{f}_*)p(\mathrm{f}_*|\mathbf{X}, \mathbf{y}, x_*)d\mathrm{f}_*. \end{aligned} \tag{8}$$

The non-Gaussian likelihood of (5) for the classification framework given by (4) makes the integrals in (7) and (8) analytically intractable. Dropping the conditioning on the training and test data points for ease of notation and applying the reverse chain rule, we have

$$p(\mathbf{f}, \mathrm{f}_*|\mathbf{y}) = p(\mathrm{f}_*|\mathbf{f})p(\mathbf{f}|\mathbf{y}). \tag{9}$$

The first term of (9) can be computed by applying the multivariate Gaussian conditional rule to the GP prior, while the second term can be approximated with variational inference [25]. A multivariate Gaussian variational distribution $q(\mathbf{f})$ approximating the posterior $p(\mathbf{f}|\mathbf{y})$ is found through the minimization of the Kullback–Leibler divergence (KL divergence) [26] $\mathrm{KL}(q(\mathbf{f})||p(\mathbf{f}|\mathbf{y}))$. Since this similarity measure is also intractable, the variational evidence lower buond (ELBO) is used as a proxy for the KL divergence minimization:

$$\begin{aligned} \mathcal{L}_{\text{ELBO}} = \; & \mathbb{E}_{p(y,\mathbf{x})}\left[\mathbb{E}_{p(\mathrm{f}|\mathbf{u},\mathbf{x})q(\mathbf{u})}\left[\log p(y\,|\,\mathrm{f})\right]\right] \\ & - \; \mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right] \\ \approx \; & \sum_{i=1}^{N} \mathbb{E}_{q(\mathrm{f}_i)}\left[\log p(y_i\,|\,\mathrm{f}_i)\right] - \; \mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right], \end{aligned} \tag{10}$$

where N is the number of data points, $q(\mathbf{u})$ is the Gaussian variational distribution computed at the inducing function values $\mathbf{u}$, $q(\mathrm{f}_i)$ is the marginal of $p(\mathrm{f}_i|\mathbf{u}, x_i)q(\mathbf{u})$, $p(\mathbf{u})$ is the GP prior distribution for the inducing function values. The ELBO is the lower bound of the log marginal likelihood $\log(p(\mathbf{y}))$, also called model evidence, and it is an expression containing all the parameters defining the GP prior and the variational distribution; thus, gradient descent can be used to maximize the ELBO concerning the model parameters to find concrete values for those parameters. In practice, the negation of (10) will be used as the "loss" function to determine the "hyperparameters" of the GP prior distribution.

2.2. Bayesian Active Learning by Disagreement

The fundamental principle of active learning requires that a model actively selects input queries $x_i \in \mathbf{X}$ and observes the system's response y_i, rather than passively collecting (x_i, y_i) pairs. The goal of information theoretic active learning is to reduce the number of possible hypotheses in the fastest way, i.e., to minimize the uncertainty about the parameters using Shannon's entropy [27]. The objective is to seek the data point $\mathbf{x}$ that maximizes the decrease in expected posterior entropy [15]:

$$\arg\max_{\mathbf{x}} \mathrm{H}[\boldsymbol{\theta}|\mathcal{D}] - \mathbb{E}_{\,y\sim p(y|\mathbf{x},\mathcal{D})}[\mathrm{H}[\boldsymbol{\theta}|\mathcal{D}]],$$

or, equivalently, the point maximizing the conditional mutual information between the unknown output and the parameters $\boldsymbol{\theta}$, given a training dataset $\mathcal{D}$:

$$\arg\max_{\mathbf{x}} \mathrm{H}[y|\mathbf{x},\mathcal{D}] - \mathbb{E}_{\,\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathcal{D})}[\mathrm{H}[y|\mathbf{x},\boldsymbol{\theta}]].$$

BALD searches the $\mathbf{x}$ for which the model is marginally most uncertain about y, holding confident individual settings of the model parameters.

The BALD algorithm for GPC consists of two steps. First, it applies an approximate inference algorithm for GPCs to obtain the posterior predictive mean $\mu_{\mathbf{x},\mathcal{D}}$ and variance $\sigma^2_{\mathbf{x},\mathcal{D}}$ for each point of interest $\mathbf{x}$. Then, it selects a query $\mathbf{x}$ that maximizes the mutual information. The first term can be expressed in terms of the binary entropy function h:

$$\begin{aligned} \mathrm{H}[y|\mathbf{x},\mathcal{D}] &\approx \mathrm{h}\left(\int \Phi(\mathrm{f}_\mathbf{x})\mathcal{N}(\mathrm{f}_\mathbf{x}|\mu_{\mathbf{x},\mathcal{D}}, \sigma^2_{\mathbf{x},\mathcal{D}})d\mathrm{f}_\mathbf{x}\right) \\ &= \mathrm{h}\left(\left(\Phi\left(\frac{\mu_{\mathbf{x},\mathcal{D}}}{\sqrt{\sigma^2_{\mathbf{x},\mathcal{D}}+1}}\right)\right)\right), \end{aligned} \tag{11}$$

with $\mathrm{h}(p) = -p\log p - (1-p)\log(1-p)(p)$. The second term $\mathbb{E}_{\,\mathrm{f}\sim p(\mathrm{f}|\mathcal{D})}[\mathrm{H}[y|\mathbf{x},\boldsymbol{\theta}]]$ can be approximated to

$$\mathbb{E}_{\,\mathrm{f}\sim p(\mathrm{f}|\mathcal{D})}[\mathrm{H}[y|\mathbf{x},\boldsymbol{\theta}]] \approx \frac{C}{\sqrt{\sigma^2_{\mathbf{x},\mathcal{D}}+C^2}} \exp\left(-\frac{\mu^2_{\mathbf{x},\mathcal{D}}}{2\left(\sigma^2_{\mathbf{x},\mathcal{D}}+C^2\right)}\right),$$

where $C = \sqrt{\pi \ln 2/2}$.

3. GPC FOR 2AFC

Finding the most accurate way to approximate the PF describing lateralization ability is the main objective of this work. A particularly important value in two alternative forced choice (2AFC) tests is the point at which the PF assumes a certain percentage, typically 70.7%, 75%, 76%, or 79.4% [28], with which the ITD threshold is associated. In these psychometric tests, the probability of assigning the correct classification label to a specific stimulus cannot be less than 50%, and the maximum probability must consider a percentage of errors given by a lapse rate close to zero [29].For these reasons the Gaussian-modeled latent function is squashed into the interval [0.5, 1]. The mean and covariance functions of the GP prior given by (1) are respectively set to a constant function $\mu(x) = \mu$, and the radial basis function

$$K(x, x') = \exp\left(-\frac{1}{2}(x-x')^\top \theta^{-2}(x-x')\right), \tag{12}$$

where θ is a length scale parameter. The hyperparameters' vector $\boldsymbol{\theta} = (\mu, \theta)$ is determined during the training, i.e., during the minimization of the negative log likelihood given by the negation of the variational ELBO in (10). The variational ELBO is modeled to approximate the likelihood once it is scaled to the restricted probability values. This likelihood is computed as

$$\Phi\left(C'\right) = \frac{1}{4}\left(1 + \mathrm{erf}\left(\frac{C'}{\sqrt{2}}\right)\right) + \frac{1}{2}, \tag{13}$$

where $C' = \mu_{\mathbf{x},\mathcal{D}}/\sqrt{\sigma^2_{\mathbf{x},\mathcal{D}}+1}$, $\mu_{\mathbf{x},\mathcal{D}}$ and $\sigma^2_{\mathbf{x},\mathcal{D}}$ are respectively the mean and the variance of the Gaussian-modeled latent function, and erf is the error function:

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z \exp -t^2 dt. \tag{14}$$

To take this scaling into account while computing the acquisition function of the BALD procedure, we rescale the likelihood between 0 and 1 at the Shannon's entropy input. This way, a maximum entropy starting around the 75% probability point is obtained.

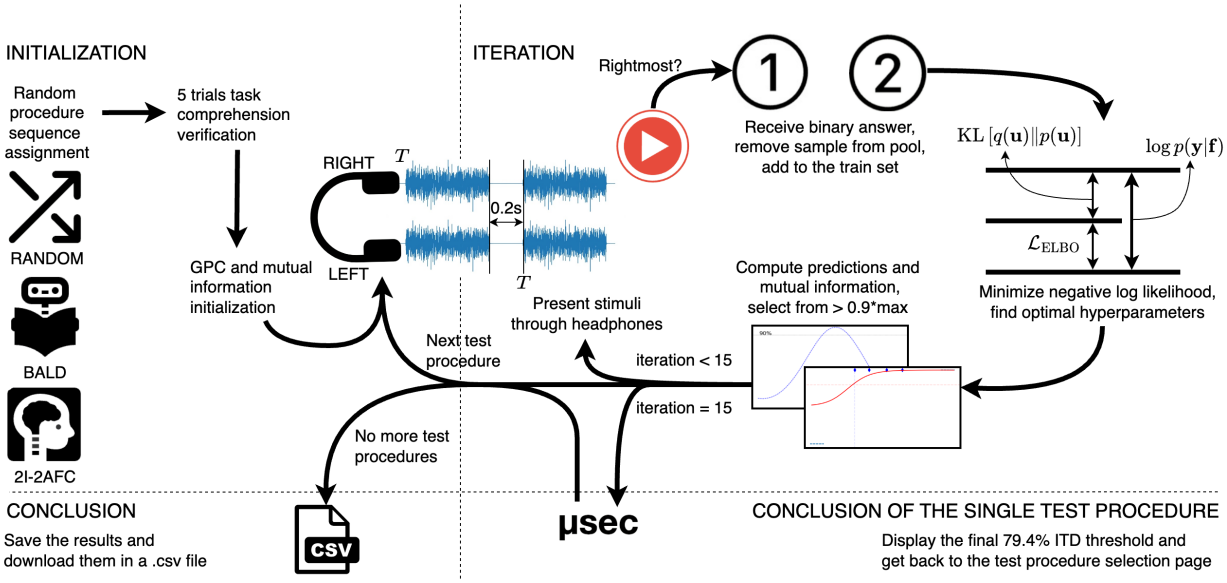


Figure 1: *Test protocol and experimental flow.*

4. THE EXPERIMENT ON ITD DISCRIMINATION THRESHOLDS

4.1. Participants

Seventeen young adults (5 male and 12 female, mean age: 31.82 ± 6.38 years) reporting no hearing deficiencies participated in the experiment. Twelve had audiometric thresholds equal to or less than 20 dB hearing level (HL) at octave-spaced frequencies from 125 to 8000 Hz. For those five participants without audiometry, the answers to Sanders' questionnaire [1] reported no hearing difficulties. One participant was excluded from the experiment since reporting to be unable to concentrate sufficiently during the test. All participants reported no prior experience with a binaural hearing test.

4.2. Acoustic Stimulus and Apparatus

Narrowband noise was synthesized at 10 MHz sampling rate, using the Python TorchAudio software package [30]. Band-pass filtering was performed in the frequency domain so as to limit the noise frequency band to the [20–1400] Hz range [6]. The amplitude level was calibrated to 70 ± 1 dB SPL using an NTi Audio XL2 sound level meter. After temporal gating, a short noise burst lasting 0.5 s was created having a 50 ms squared-cosine onset and offset.

A burst sequence was formed by intertwining identical noise bursts with silence lasting 0.2 s. The stimulus was formed by pairing a delayed version of this sequence and a new version of the same sequence, obtained by zero-padding the original until reaching the same length as the former. The delay could be varied so as to define a desired ITD $= T_R - T_L = T - (-T) = 2T$:

$$\begin{aligned} \text{RIGHT} \quad & \overbrace{0,\ldots,0}^{T},\text{burst},\overbrace{0,\ldots\ldots,0}^{0.2\text{ s}},\text{burst},\overbrace{0,\ldots,0}^{\text{padding}} \\ \text{LEFT} \quad & \text{burst},\underbrace{0,\ldots,0}_{\text{padding}},\underbrace{0,\ldots\ldots,0}_{0.2\text{ s}},\underbrace{0,\ldots,0}_{T},\text{burst} \end{aligned} . \tag{15}$$

The stimulus was presented on two audio channels through a pair of AKG K240 MKII semi-open headphones, whose frequency response was flattened using the AutoEQ software[2]. Sounds were reproduced by a 13" MacBook Pro M2 laptop computer after sampling them down to 96 kHz, through interpolation with a sync function windowed by a Hann window. This sampling rate was the highest available in the laptop's audio interface, in practice limiting the lowest ITD to 10 μs. A GUI enabling attendance to the task was realized in HTML, CSS, and JavaScript programming languages as a custom Flask web application. The test took place in a room with background noise equal to 20 ± 2.5 dB SPL.

4.3. Task and Experimental Protocol

The experimental protocol is illustrated in Fig. 1. During the test, each participant was sitting in front of the laptop computer running the GUI. At each trial, the task consisted of listening to two subsequent and randomly balanced stimuli, and then choosing which sound was the rightmost, by selecting it with the mouse on the computer screen. At the beginning of each session, five pilot trials were presented having ITD levels equal to 240, 200, 160, 120, and 80 μs. Correct guesses in all such trials were necessary for the measurements to start in correspondence with the sixth trial. A session lasted approximately 10 minutes.

The protocol was designed for determining the PF 79.4% threshold, describing the subjective lateralization performance as a function of T defined in (15). The target (i.e., rightmost) stimulus

[1] https://www.aooi.it/contents/attachment/c4/ref121.pdf, a validated questionnaire to evaluate the actual level of communication in various situations, e.g., at home or in a social environment, accessed Feb 28, 2023.

[2] https://githubq.com/jaakkopasanen/AutoEq, accessed Feb 28, 2023.

was lateralized twice as much as a nominal ITD, and the reference source (i.e., leftmost) stimulus was instead lateralized with an opposite ITD. Hence, an ITD threshold equal to $2T$ means that a participant discriminated the target ITD by $T\mu$s from the reference ITD of $-T\mu$s. The presentation of symmetric ITDs minimizes hemispheric effects, and ensures that participants could not perform the task based on perceived changes in interaural coherence [28].

The protocol implemented three different procedures:

- adaptive two-interval 2AFC (2I-2AFC hereafter),
- GPC with BALD active learning (BALD hereafter),
- GPC with random acquisition function (RANDOM hereafter).

Accordingly, every session included three series of trials respectively implementing such procedures in a randomly balanced order. When GPC was used, thus enabling active learning and random selection of ITDs, the number of trials was empirically set to 15. We compared these three procedures to observe the efficacy of GPC in ITD threshold estimation and the contribution of the BALD algorithm. In the standard 2I-2AFC procedure, individual thresholds were computed using the adaptive track reversal technique or a Weibull fitting of the participants' answers. We consider this procedure as a reference since it is commonly used to determine JND thresholds for many types of stimuli, moreover it can be designed to converge on a desired percentage. Conversely, the GPC approximation was compared to a Weibull fitting of the data collected using the other two procedures. Reversals were included in the comparison even though they are known to have high bias and smaller precision [31], considering their difference from a Weibull fitting in the same manner as it happens for the upper limit of a reasonable estimate. The RANDOM procedure was implemented to test the GPC technique separately from active learning, i.e., to analyze the GPC on training points other than highly informative data points. Assuming a participant performs consistently across all tests, the RANDOM procedure will converge, but in a variable number of iterations dependent on the samples' random distribution.

4.3.1. Default 2I-2AFC

An adaptive "three-down, one-up" staircase procedure was chosen, i.e., T was decreased after three correct responses and increased after one wrong response. Theoretically this procedure estimates the 79.4% correct level on the PF [32]. The corresponding series of trials started with an ITD threshold that was above the threshold estimated during the pilot series, equal to 80 μs, i.e., presenting stimuli with $T = 40$ μs. T step-size was initially a factor of 2 and then reduced to 1.414 and 1.189 after the first and second "down-up reversal", respectively. This series terminated after six reversals at the smallest step size. T was varied by logarithmic steps [32].

Psychometric functions were estimated using a parametric fit of a Weibull function to all responses with a non-linear least squares optimization:

$$\Psi(x;\gamma,\lambda) = \gamma + \frac{(1-\gamma-\lambda)}{2}\left(1 + \text{erf}\left(\frac{x-\mu_\Psi}{\sqrt{2\cdot\sigma_\Psi^2}}\right)\right), \quad (16)$$

where γ is the guess rate (i.e., 0.5), λ is the miss rate in the range [0.01,0.05], and erf is the error function of (14).

4.3.2. GPC

The GPC started with a training set of 20 points: 10 points between 0 μs and 9 μs, all labeled as wrong answers, and 10 points between 91 μs and 100 μs, all labeled as correct answers. T corresponding to 79.4% of correct answers was found to allow the fitted curve to reach the closest value to that percentage up to 1 μs. In Fig. 2, the two plots respectively represent the predictions of the GPC and Weibull fit, on a test set consisting of 100 equally-spaced points between 1 and 100 μs. Both identify the 79.4% correct answers point with a difference equal to 8 μs.

The GPC training, i.e., the hyperparameters' vector $\boldsymbol{\theta}$ optimization, was performed on the normalized data (mean equal to 0 and standard deviation equal to 1) and furthermore constrained to search only positive values. In this regard, the Adam optimizer [33] was employed with a learning rate equal to 0.1 and a number of iterations set to 300. The GPC was implemented in GPyTorch [34], a software platform for scalable GP inference built on PyTorch.

4.3.3. BALD

The BALD acquisition function was computed at each step to determine the stimulus for the next trial iteratively. T of the binaural stimulus was randomly selected among all possible points in the pool, achieving mutual information levels higher than 90% of its maximum value in that set. The initial pool was made to correspond to the test set, i.e., the 100 points between 1 and 100 μs. Once a sample was labeled, it was removed from the pool so that each point was labeled once. After the first random selection, a new sample was chosen from that restricted subset of the pool being at least 5 μs far from every previously selected sample. If no sample satisfied this condition, a random choice was made from the samples achieving mutual information levels higher than 90% of its maximum value in the current pool's subset. The first T value was randomly drawn between 46 μs and 64 μs.

5. RESULTS

The number of trials during the 2I-2AFC procedure had a mean value across participants equal to 39.94 and a standard deviation

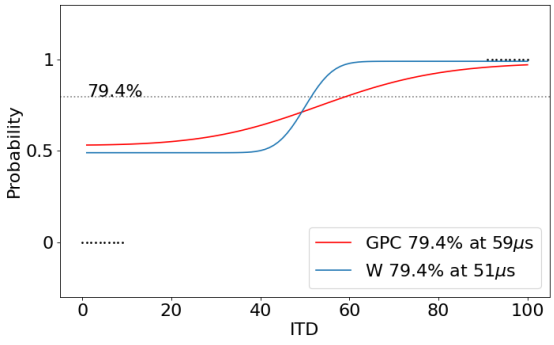


Figure 2: *Starting training data for the GPC (black dots) and predictions of both the GPC (red line) and the fitted Weibull function (W, blue line). The dotted horizontal line indicates the 79.4% level of correct answers.*

Table 1: *Individual 79.4% thresholds and means (μ) and standard deviations (σ) across participants for each technique. The "Rev" column lists T values found with the reversals procedure.*

#	@ 79.4% (μs)					
	2I-2AFC		BALD		RANDOM	
	W	Rev	W	GPC	W	GPC
1	10	14.1	6	1	48	34
2	57	67.2	54	54	56	43
3	25	33.6	23	19	31	22
4	14	11.9	16	10	1	10
5	22	33.6	34	26	2	17
6	39	40.0	57	41	16	13
7	30	33.6	35	23	16	8
8	39	47.6	38	32	57	39
9	46	67.2	48	53	52	52
10	23	28.3	26	22	25	23
11	40	47.6	38	32	54	44
12	10	14.1	19	17	20	15
13	58	67.3	52	55	47	30
14	21	23.8	23	18	20	15
15	39	47.6	51	36	57	66
16	45	47.6	55	49	38	22
17	31	33.6	29	23	39	39
μ	32.3	38.8	35.5	30.1	34.1	28.9
σ	14.4	17.4	15.0	15.5	18.7	15.9

equal to 5.70. The correct percentage had a mean equal to 80.20% and standard deviation equal to 3.85%. The repeated measures ANOVA test conducted on the number of trials asserted that the difference between the averages is big enough to be statistically significant ($p < 0.001$), and the magnitude of the difference between the averages is large (effect size $\eta > 0.9$). Table 1 shows the 79.4% ITDs for every test procedure, computed with the two respective aforementioned techniques. Mauchly's test of sphericity indicated that the assumption of sphericity had been violated, both for the individual T values ($\chi^2(14) = 74.11$, $p < 0.001$), and their logarithms ($\chi^2(14) = 92.848$, $p < 0.001$), therefore a Greenhouse-Geisser correlation was used (respectively, $\epsilon = 0.321$, and $\epsilon = 0.338$). The repeated measures ANOVA test did not reveal a significant main effect of either the fitting technique, or the testing procedure ($p > 0.1$). The means and standard deviations across participants of the differences of the 79.4% threshold points are shown in Table 2, along with their absolute values, between approximation techniques and procedures. The former reveals the tendency of the approximation to return an optimistic rather than pessimistic estimate of the ITD threshold, while the latter gives a measure of the divergence between the two approximations.

The comparison between the PF curve found with the Weibull function fitted to the reference test procedure data and the GPC, and the same Weibull fitting on the BALD and the random sampling procedures has been made using the root mean square error (RMSE). The lowest RMSE was found between the Weibull fittings in the reference test procedure and the BALD procedure ($\mu = 6.39\%, \sigma = 2.44\%$), followed by the GPC in the same test procedure ($\mu = 7.10\%, \sigma = 2.73\%$). Both the approximation techniques in the random procedure have a mean RMSE above 10% (Weibull: $\mu = 10.81\%, \sigma = 5.20\%$, GPC: $\mu = 11.10\%, \sigma = 4.65\%$). Figure 3 displays the data of a single individual collected in a complete test session.

Table 2: *Means (μ) and standard deviations (σ) of the differences (Δ) and the absolute values of the differences ($|\Delta|$) of the 79.4% T values found in the three procedures with the three techniques. Each approximation was compared with the one provided by the Weibull fitting on the same data, as well as with the approximation computed by that fitting on the reference 2I-2AFC test procedure ("W_{REF}").*

Procedure	Methods	Δ (μs)		$\lvert\Delta\rvert$ (μs)	
		μ	σ	μ	σ
2I-2AFC	Rev, W	6.5	5.1	6.7	4.8
BALD	GPC, W_{REF}	-2.2	5.1	5.1	2.4
	W, W_{REF}	3.2	6.6	5.6	4.7
	GPC, W	-5.5	5.3	6.4	4.1
RANDOM	GPC, W_{REF}	-3.4	15.4	12.1	10.1
	W, W_{REF}	1.8	15.2	12.4	9.1
	GPC, W	-5.1	9.3	9.0	5.7

6. DISCUSSION

The mean values displayed in Table 1 are in line with the thresholds found in literature [7]. Particularly in our experiment, in which stimuli with a sampling frequency below 100 KHz were reproduced, no ITDs smaller than 10 μs could be presented to the listeners. However, our participants were not trained for the specific task, and hence they were not expected to perceive ITD thresholds below this value. Conversely, it was demonstrated that humans can improve their lateralization skills after a dedicated training [35, 28].

The ITD is typically allowed to vary on a logarithmic scale in the lateralization literature, and data analysis is typically done using geometric means and standard deviation [36, 28]. However, alternative methodologies [35, 37] employ different test conditions in terms of the stimuli presented to the participant, and the use of logarithmic scaling allowed them to find one individual psychometric curve. Second, the goal of those experiments was to determine the smallest perceivable ITD, hence the use of the geometric mean and standard deviation to "zoom in" on the end scale of that measure. On the other hand, we want to determine the individual ITD thresholds with the same accuracy for any level of lateralization ability, particularly considering a possible application of the proposed procedure in audiology tests. Moreover, we expect that the same individual would perform similarly in each test because the stimulus and the task were kept unchanged during the whole session. Thus, data analysis is performed with an arithmetic mean and standard deviation.

The differences Δ in Table 2 reveal that the GPC with the BALD procedure is on average the closest 79.4% approximation to the reference 2I-2AFC procedure with the specific Weibull fitting (5.1 μs), providing a mean accuracy increase of 8.9% over the Weibull function fitted to the same data. This difference has the lowest standard deviation (2.4 μs). The GPC in the BALD procedure scores second for what concerns the signed difference (-2.2 μs), revealing an optimistic tendency; only the Weibull fitting in the random test procedure has a smaller one (1.8 μs), however the former has the lowest standard deviation while the latter has one among the highest, hence reflecting the prominent role of chance associated with that test procedure. The standard deviations of the differences Δ of the RANDOM procedure reflect the random number of iterations required by the GPC in that procedure to converge.

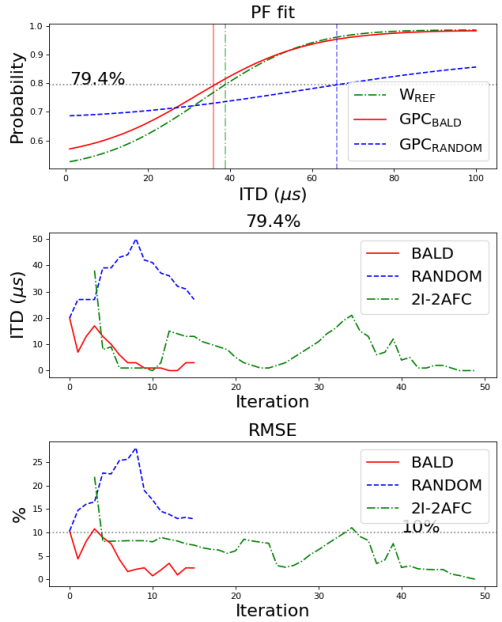


Figure 3: *Data of a single individual collected in a complete session. The top figure shows the GPC PF fittings and* $\mathrm{W_{REF}}$ *fitting. In the middle figure are displayed the absolute values of the differences of the 79.4% threshold points, while the bottom figure shows the RMSEs between the GPCs and the* $\mathrm{W_{REF}}$*, computed at every iteration step of the three procedures. The Weibull fitting in the 2I-2AFC procedure starts at the third iteration because the covariance of the parameters could not be estimated with less than three data points.*

Even if the analyses of variance did not reveal any statistically significant main difference, the BALD selection of samples is also valuable for the Weibull fitting since both the approximations used on that procedure's data have the smallest differences in absolute value and the least dispersed as well.

The RMSEs reported at the end of Sec. 5 confirm the suitability of GPC to approximate the PF in a 2AFC procedure and the ability of the BALD algorithm to identify optimal sampling; a further hint of this is that the standard deviations of the RANDOM procedure are the highest.

Future work may explore different directions. The proposed experimental protocol may be employed to test pre-trained individuals instead of novices and feed the GPC with a logarithmically transformed input to observe which fit gives the best results. Another direction may evaluate the BALD procedure without validated prior knowledge, such as for the hearing impaired, and check whether the starting training set needs to be modified or if other approximations are more suitable than GPC. The web application was chosen to share the test between different institutions and audiological research labs, allowing them to conduct the experiment remotely. In the current study, the application was used on a local server. The next step will include a thorough verification for deployment on a cloud platform.

7. CONCLUSIONS

In this study, a test for the fast determination of the Just Noticeable Difference in interaural time differences has been proposed and evaluated. A psychometric function was progressively fitted using Gaussian process classification of the subjective responses and Bayesian active learning by disagreement, aptly modified to accommodate a two-alternative forced choice experimental procedure. The results of its comparison with a standard adaptive "three-down, one-up" staircase procedure show that our process computes the closest approximation of the average threshold ITD 79.4% correct level of lateralization concerning the commonly used Weibull fitting on the reference test, with a mean accuracy increase of 8.9% over the Weibull function fitted on the data of the same test. The final accuracy was achieved with only 37.6% of the trials the standard adaptive staircase procedure needs.

The data and the web application are freely available at `https://zenodo.org/record/7808559` and `https://github.com/gullogullo/ITDtest`, respectively.

8. ACKNOWLEDGMENTS

We thank the Multisensory Experience Laboratory at the Aalborg University, Denmark, for making this research possible. Ali Adjorlu suggested design ideas for the GUI. The staff of the Complex Structure of Otorhinolaryngology and Audiology at the Institute for Maternal and Child Health IRCCS "Burlo Garofolo" in Trieste, Italy, participated in the experiment.

9. REFERENCES

[1] Lord Rayleigh, "Xii. on our perception of sound direction," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 13, no. 74, pp. 214–232, 1907.

[2] Allen William Mills, "On the minimum audible angle," *The Journal of the Acoustical Society of America*, vol. 30, no. 4, pp. 237–246, 1958.

[3] Otto Klemm, "Investigations of the localization of sound stimuli iv: On the influence of binaural time difference on localization," *Arch. Ges. Psychol.*, vol. 40, pp. 117–145, 1920.

[4] Brian CJ Moore, *An introduction to the psychology of hearing*, Brill, 2012.

[5] Roy G Klumpp and Herman R Eady, "Some measurements of interaural time difference thresholds," *The Journal of the Acoustical Society of America*, vol. 28, no. 5, pp. 859–860, 1956.

[6] Sinthiya Thavam and Mathias Dietz, "Smallest perceivable interaural time differences," *The Journal of the Acoustical Society of America*, vol. 145, no. 1, pp. 458–468, 2019.

[7] Kubo Tetsushi, Tetsushi Sakashita, Makoto Kusuki, Kazushi Kyunai, Keita Ueno, Chie Hikawa, Tadashi Wada, Toshiyuki Shibata, Yoshiaki Nakai, and Takeshi, "Sound lateralization and speech discrimination in patients with sensorineural hearing loss," *Acta Oto-Laryngologica*, vol. 118, no. 543, pp. 63–69, 1998.

[8] Michele Geronazzo and Stefania Serafin, *Sonic Interactions in Virtual Environments*, Springer Nature, 2023.

[9] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell, "Active learning with gaussian processes for object categorization," in *2007 IEEE 11th international conference on computer vision*. IEEE, 2007, pp. 1–8.

[10] Benjamin Letham, Phillip Guan, Chase Tymms, Eytan Bakshy, and Michael Shvartsman, "Look-ahead acquisition functions for bernoulli level set estimation," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 8493–8513.

[11] Manuel Haussmann, Fred A Hamprecht, and Melih Kandemir, "Deep active learning with adaptive acquisition," *arXiv preprint arXiv:1906.11471*, 2019.

[12] Alan B Cobo-Lewis, "An adaptive psychophysical method for subject classification," *Perception & Psychophysics*, vol. 59, no. 7, pp. 989–1003, 1997.

[13] Claude E Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[14] Carl Edward Rasmussen, Christopher KI Williams, et al., *Gaussian processes for machine learning*, vol. 1, Springer, 2006.

[15] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel, "Bayesian active learning for classification and preference learning," *arXiv preprint arXiv:1112.5745*, 2011.

[16] Mijung Park, Greg Horwitz, and Jonathan Pillow, "Active learning of neural response functions with gaussian processes," *Advances in neural information processing systems*, vol. 24, 2011.

[17] Bert de Vries, "A gaussian process mixture prior for hearing loss modeling," in *Benelearn 2017: Proceedings of the Twenty-Sixth Benelux Conference on Machine Learning, Technische Universiteit Eindhoven, 9-10 June 2017*, 2017, p. 74.

[18] Dennis L Barbour, Rebecca T Howard, Xinyu D Song, Nikki Metzger, Kiron A Sukesan, James C DiLorenzo, Braham RD Snyder, Jeff Y Chen, Eleanor A Degen, Jenna M Buchbinder, et al., "Online machine learning audiometry," *Ear and hearing*, vol. 40, no. 4, pp. 918, 2019.

[19] Jens Brehm Bagger Nielsen, Jakob Nielsen, and Jan Larsen, "Perception-based personalization of hearing aids using gaussian processes and active learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 162–173, 2014.

[20] Josef Schlittenlacher, Richard E Turner, and Brian CJ Moore, "Audiogram estimation using bayesian active learning," *The Journal of the Acoustical Society of America*, vol. 144, no. 1, pp. 421–430, 2018.

[21] Xinyu D Song, Brittany M Wallace, Jacob R Gardner, Noah M Ledbetter, Kilian Q Weinberger, and Dennis L Barbour, "Fast, continuous audiogram estimation using machine learning," *Ear and hearing*, vol. 36, no. 6, pp. e326, 2015.

[22] Josef Schlittenlacher and Brian CJ Moore, "Fast estimation of equal-loudness contours using bayesian active learning and direct scaling," *Acoustical Science and Technology*, vol. 41, no. 1, pp. 358–360, 2020.

[23] Xinyu D Song, Roman Garnett, and Dennis L Barbour, "Psychometric function estimation by probabilistic classification," *The Journal of the Acoustical Society of America*, vol. 141, no. 4, pp. 2513–2525, 2017.

[24] Keith A May and Joshua A Solomon, "Four theorems on the psychometric function," *PLoS One*, vol. 8, no. 10, pp. e74815, 2013.

[25] James Hensman, Alex Matthews, and Zoubin Ghahramani, "Scalable variational gaussian process classification," *arXiv preprint arXiv:1411.2005*, 2014.

[26] David M Blei, Alp Kucukelbir, and Jon D McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[27] Thomas M Cover, *Elements of information theory*, John Wiley & Sons, 1999.

[28] Mathias Dietz, Stephan D Ewert, and Volker Hohmann, "Lateralization based on interaural differences in the second-order amplitude modulator," *The Journal of the Acoustical Society of America*, vol. 131, no. 1, pp. 398–408, 2012.

[29] Felix A Wichmann and N Jeremy Hill, "The psychometric function: I. fitting, sampling, and goodness of fit," *Perception & psychophysics*, vol. 63, no. 8, pp. 1293–1313, 2001.

[30] Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Anjali Chourdia, Artyom Astafurov, Caroline Chen, Ching-Feng Yeh, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jay Mahadeokar, Jeff Hwang, Ji Chen, Peter Goldsborough, Prabhat Roy, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, Vincent Quenneville-Bélair, and Yangyang Shi, "Torchaudio: Building blocks for audio and speech processing," *arXiv preprint arXiv:2110.15018*, 2021.

[31] Miguel A Garcıa-Pérez, "Forced-choice staircases with fixed step sizes: asymptotic and small-sample properties," *Vision research*, vol. 38, no. 12, pp. 1861–1881, 1998.

[32] William A Yost, Robert Turner, and Byron Bergert, "Comparison among four psychophysical procedures used in lateralization," *Perception & Psychophysics*, vol. 15, no. 3, pp. 483–487, 1974.

[33] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," *Advances in neural information processing systems*, vol. 31, 2018.

[35] Beverly A Wright and Matthew B Fitzgerald, "Different patterns of human discrimination learning for two interaural cues to sound-source location," *Proceedings of the National Academy of Sciences*, vol. 98, no. 21, pp. 12307–12312, 2001.

[36] Kourosh Saberi, "Some considerations on the use of adaptive methods for estimating interaural-delay thresholds," *The Journal of the Acoustical Society of America*, vol. 98, no. 3, pp. 1803–1806, 1995.

[37] Jennifer E Mossop and John F Culling, "Lateralization of large interaural delays," *The Journal of the Acoustical Society of America*, vol. 104, no. 3, pp. 1574–1579, 1998.

[38] Christopher KI Williams and David Barber, "Bayesian classification with gaussian processes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 12, pp. 1342–1351, 1998.

[39] Thomas P Minka, "Expectation propagation for approximate bayesian inference," *arXiv preprint arXiv:1301.2294*, 2013.

[40] Julien Bect, David Ginsbourger, Ling Li, Victor Picheny, and Emmanuel Vazquez, "Sequential design of computer experiments for the estimation of a probability of failure," *Statistics and Computing*, vol. 22, pp. 773–793, 2012.

[41] Mohammad Khan, *Variational learning for latent Gaussian model of discrete data*, Ph.D. thesis, University of British Columbia, 2012.

[42] Andrew B Watson and Denis G Pelli, "Quest: A bayesian adaptive psychometric method," *Perception & psychophysics*, vol. 33, no. 2, pp. 113–120, 1983.

[43] Josef Schlittenlacher, Richard E Turner, and Brian CJ Moore, "The implementation of efficient hearing tests using machine learning," in *Proceedings of the International Symposium on Auditory and Audiological Research*, 2019, vol. 7, pp. 1–12.

[44] Felix A Wichmann and N Jeremy Hill, "The psychometric function: Ii. bootstrap-based confidence intervals and sampling," *Perception & psychophysics*, vol. 63, pp. 1314–1329, 2001.

[45] Harry CCH Levitt, "Transformed up-down methods in psychoacoustics," *The Journal of the Acoustical society of America*, vol. 49, no. 2B, pp. 467–477, 1971.

[46] Brent Bryan, Robert C Nichol, Christopher R Genovese, Jeff Schneider, Christopher J Miller, and Larry Wasserman, "Active learning for identifying function threshold boundaries," *Advances in neural information processing systems*, vol. 18, 2005.

[47] Lucy Owen, Jonathan Browder, Benjamin Letham, Gideon Stocek, Chase Tymms, and Michael Shvartsman, "Adaptive nonparametric psychophysics," *arXiv preprint arXiv:2104.09549*, 2021.

DECORRELATION FOR IMMERSIVE AUDIO APPLICATIONS AND SOUND EFFECTS

Sascha Disch

Fraunhofer IIS
Fraunhofer Institute for Integrated Circuits
Erlangen, Germany
sascha.disch@iis.fraunhofer.de

ABSTRACT

Audio decorrelation is a fundamental building block for immersive audio applications. It has applications in parametric spatial audio coding, audio upmix, audio sound effects and audio rendering for virtual or augmented reality applications. In this paper, we provide insights into the practical design considerations of an audio decorrelator on the example of the decorrelator contained within the upcoming MPEG-I Immersive Audio ISO standard [1]. We describe the desirable properties of such a decorrelator, common approaches for implementation and our particular technology choices for the decorrelator used in MPEG-I for rendering sound sources with homogeneous extent.

1. INTRODUCTION

Historically, many electronic sound effects were designed with the aim to make the sound of an instrument more prominent or "larger" [2]. Usually this came at the price of certain originally unintended changes in sound timbre, modulation or pitch variations. Examples of such effects are (stereophonic) chorus, flanger, phaser and the like or, if it comes to multitrack recording, the studio technique of doubling and layering instrumental parts or voices. However, these artifacts and characteristic sound changes soon were welcomed and accepted by producers and artists and artistically integrated into their art performances.

Opposed to these effects, the aim of audio decorrelation is to as well as possible avoid any perceivable change in the original audio signal and provide decorrelated, but subjectively indistinguishable copies of the original input signal. Such decorrelated signals can then be used e.g. for spatially distributing multichannel sound or modeling spatial source extent. The latter denotes the technique of rendering a sound "larger" than the one emitted from a point source, but rather as if originating from a spatially extended sound source. Ideally, this can be accomplished without any unintended artifacts and timbre changes.

Applications for audio decorrelation are in perceptual coding of audio (parametric stereo/multichannel codecs) [3][4][5], subjective sound enhancement like audio upmix for channel signals and ambisonic signals and also for Virtual Reality (VR) and Augmented Reality (AR) rendering applications [6]. VR and AR rendering applications implement 6-Degrees-of-Freedom (6DoF) audio rendering for an interactive listener position in 3-dimensional space (3DoF) plus allowing for arbitrary listener's head rotary movements in all three possible axes (pitch, yaw, roll). One example of such a render is the upcoming MPEG-I Immersive Audio Standard [7] [1][8].

The decorrelation described in this paper has been designed especially for use within said MPEG-I Immersive Audio standard. In the MPEG-I renderer, it is utilized for modeling sound sources with spatial extent [9].

However, the decorrelator is pretty much self-contained and may find uses in other application including spatially enhanced audio effects. A combination of decorrelation with well-known traditional audio effects, eg. echo, delay, panning, tremolo, appears to be interesting to spatially widen and distribute the raw effect signal.

2. BACKGROUND

2.1. Previous Work

In the past, there have been a number of publications on audio signal decorrelation, for instance from Kendall [10], Boueri et al. [11] and Kermit et al. [12]. A publication of Potard et al. [13] addressed the rendering of sound source width through decorrelation and Anemüller et al. [9] have utilized the decorrelator presented in this paper to model spatially extended sound sources.

Purnhagen et al. [14] proposed audio decorrelation for parametric spatial audio coding, e.g. so-called parametric stereo. Here, the decorrelated signal can be seen as the side signal of a Mid/Side (M/S) representation of a stereo signal whereas the original input signal is the mid signal. Two decorrelated output signals are then obtained from the M/S representation by conversion to Left/Right (L/R) stereo. An adaptive M/S mix can be used to adjust the perceived spatial width.

Newer publications discuss the insight that transients should be treated differently from other signal parts in decorrelation as suggested by Disch et al. [15] and Penniman [16]. The reasoning behind this is two-fold: on one hand, unwanted transient dispersion shall be avoided as an audible artefact. On the other hand, convincing multichannel reproduction of spatial width of ambience signals consisting of dense transient events (e.g. applause) is not so much achieved through altered waveforms as produced by common decorrelation techniques, but rather through a widespread spatial redistribution of these individual transient events by fine grain temporal panning [15][17].

2.2. Decorrelator Design Criteria

Audio decorrelators for practical real-time applications ideally have to fulfill the following requirements:

- Reasonable computational complexity

- Low latency
- Stream processing capability (single pass processing)

A general audio signal may conceptually be composed of a mixture of many different signal classes, like steady tones, transients, speech, talking or singing voice or noise. All these signal classes contained in the signal need to be processed by the decorrelator with little artifacts. So, a well-designed decorrelator should have a number of perceptual properties:

- Shall preserve sound timbre
- Shall not sound reverberated
- Shall avoid modulations of amplitude or pitch
- Shall allow precise reproduction of transients
- Shall decorrelate mixtures of spatially distributed transient events (applause, rain drops, fire crackling)

Audio decorrelation processing with reasonable latency and computational complexity is most successfully attempted by application of filters [18]. Allpass filters have the useful property to heavily influence the phase of a signal through their phase response while having a flat magnitude response. Although the signal waveform at a given point in time is dissimilar to the original signal, yet its timbre is still the same. The most simple allpass filter is a pure delay corresponding to a linear phase response. However, when input signal and filter output are combined, strong regularly spaced comb-filtering results and timbre colorization occures.

Therefore, allpass filters that sufficiently alter the signal's phase in a more uneven way are preferably used, e.g. a combination of different allpass filters in series or multiple allpass filters in a nested structure, where the delay element of one allpass filter is replaced by a second allpass filter. These filters usually have a long, ringing impulse response that is audible as a reverberant tail of the processed sound.

For that reason, key to successful audio decorrelation using allpass filters is the preservation or the reconstruction of the signal's temporal and spectral envelope. Any reverberant tail caused by the long impulse response of the filters can be suppressed by application of envelope shaping and thereby will become inaudible.

The phase response of suitable allpass filters is also likely to impair transients in the signal mix through phase dispersion, e.g. for sharp signal onsets like plugged strings or piano tones. A dedicated handling of such transients within the decorrelator processing can further improve perceptual quality.

3. DECORRELATOR CONCEPT

3.1. Overview

The MPEG-I decorrelator is designed to match the criteria described above with respect to perceptual quality but also with respect to latency, complexity and stream processing capability.

It is based on overlapping DFTs for analysis of the incoming signal and synthesis of the decorrelated signal. A combination of a pre-delay and an allpass filter chain is applied in the (subsampled) DFT subband domain followed by a time/Frequency (t/F) envelope shaper. The envelope shaper applies an individual temporal envelope shaping in the DFT's spectral subbands. A short DFT block size of 256 samples at 48kHz sampling rate keeps latency low at ~2.7ms and also enables direct t/F envelope shaping downstream in DFT domain with sufficient temporal resolution.

A dedicated transient handling excludes transients from being subjected to phase dispersion. Excluding onset transients from decorrelation of most point sources (apart from e.g. mono recordings of multi-transient mixtures) appears to not impair the overall listening impression of spatial extent, which is the main task of the MPEG-I decorrelator. As explained in subsection 2.1, for said ambient mixtures of spatially distributed transient events, the individual transient events would have to be spatially re-distributed by other techniques e.g. by rapid amplitude panning [15].

Consequently, transient signal blocks are copied unaltered into the decorrelator output signal. When adding/subtracting original signal and processed signal in the output assembly stage, a scaling factor accounts for the addition of these coherent signal parts and ensures equal level with the incoherently added/subtracted decorrelated signals.

The transient handling entirely aims at an artefact-free reproduction of transient onsets. In consequence, relevant time constants of the transient handling are designed to be shorter than pre-delay/filterdelay such that any traces of delayed and dispersed transient onsets are excluded from the output signal.

Figure 1 depicts the block diagram of the decorrelator. To create two decorrelated output signals y_{Out1} and y_{Out2} from one input signal x, the MPEG-I decorrelator performs the steps that are described in the following.

Novel contributions - to our best knownledge - of our new decorrelator design are the application of low order Schroeder allpass filters in the subsampled DFT domain on complex spectral subband signals, the transient handling functionality and the computationally efficient co-use of the DFT for subband filtering and subband envelope estimation and shaping.

3.2. Input Buffering

The decorrelator has an internal processing cycle, managed by an input ring buffer, of 256 sample frames with a 128-sample overlap between subsequent processing frames. A sine window according to Equation 1 is applied where $m = 0...M-1$ is the time index within a frame n and $M = 256$.

$$x_{Win}(m) = x_{In}(m) \cdot \sin\left((m + 0.5) \cdot \pi / M\right) \tag{1}$$

Next, a 256-point Discrete Fourier Transform (DFT) is performed on the windowed input frame $x_{Win}(m)$ to obtain $K = 129$ complex valued frequency bins $X_{DS}(n)$ for each frame n. In the following processing, the temporal sequences of individual DFT coefficients are treated as a set of spectral subband signals with n being the time index subsampled by a factor of 128 with respect to m.

3.3. Pre-Delay and Allpass Filters

The $k = 0...K-1$ complex DFT coefficients of each DFT subband (index k is omitted where possible for better readability) are passed through a delay, as illustrated in Equation 2, where D_{d} is the pre-delay. This pre-delay is set to 4 frames (~10.7ms).

$$X_{Del}(n) = X_{DS}(n - D_d) \tag{2}$$

Next, the delayed signal $X_{Del}(n)$ is allpass filtered according to Equation 3

$$H(z) = \frac{\gamma + z^{-D}}{1 + \gamma \cdot z^{-D}} \tag{3}$$

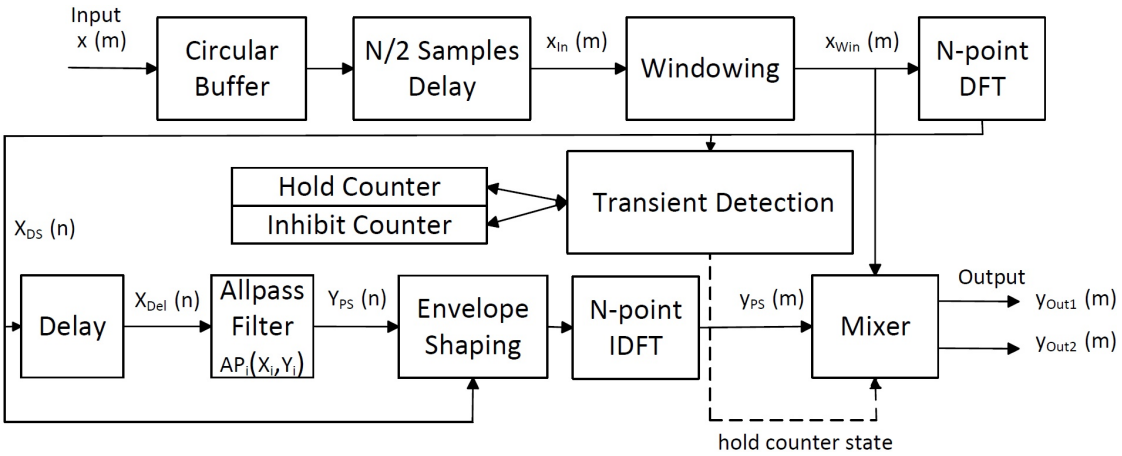


Figure 1: *Decorrelator signal flow overview. Based on overlapping DFTs for analysis of the incoming audio signal, a combination of pre-delay and allpass filter chain is applied in the DFT subband domain followed by a time/Frequency (t/F) envelope shaper and the output assembly stage controlled by a transient handling. The transient detection information (dashed line) influences the internal mixing process of the output audio signals.*

The chained series of i allpass filters AP_i is further illustrated in Equation 4, where filter coefficients are $\gamma = 0.7$ and D_{ap_i} represents the amount of delay of each allpass filter, which is set to 1, 2, 3, 5 frames for $i = 1, 2, 3, 4$.

$$Y_i(n) = \gamma \cdot X_i(n) + X_i(n - D_{ap_i}) - \gamma \cdot Y_i(n - D_{ap_i}) \quad (4)$$

In the following steps, $X_{DS}(n)$ is denoted the direct signal (DS), while $Y_{PS}(n)$, which went through the delay and the series of allpass filters, is called the processed signal (PS).

As a side note - if needed in future applications, a set of mutually independent decorrelators can be obtained by variation of predelay/allpass filter parameters and structures as sketched in subsection 2.2.

3.4. Transient Handling

In the decorrelator's output, transients are excluded from the predelay and allpass processing. This is managed by the transient handling. A transient in the current frame is detected by calculating whether the energy of the current frame, summed over $k = 4...K$ frequency bins, is stronger than the previous frame by a threshold $T = 2.8$. The accuracy of transient detection is not overly critical; a false positive transient detection result, for example, will just exclude the falsely classified signal part from decorrelation causing no further artefacts. For this reason, the implemented transient detection algorithm was kept rather simple and computationally efficient.

Two counters control the transient processing: a so-called hold counter and an inhibition counter. The hold counter controls the duration of the detected transient event and the inhibit counter enforces a minimum temporal separation between detected transient events. Both are initially set to their inactive state 0. When a transient is detected $t_n = 1$ and the inhibition counter is inactive, a hold counter is started for the next 8 frames (~21.3ms) to control a muting of the PS in the output mix. Also, the inhibition counter starts counting to prevent a hold counter start in the next 56 frames (~149.3ms). In addition, if another transient is detected during this inhibition time, this will re-start the inhibition counter, and the inhibition time will be increased from 56 to 64 frames. With these time constants, pulse trains with a pulse frequency greater than approximately 6Hz will not be subjected to the dedicated transient processing. The threshold of 6Hz was chosen since at that frequency pulses are still clearly perceived as separate events in fast succession (where preservation of transient perceptual quality is essential) rather than a continuous tonal sound (where good decorrelation is desireable). When active counters reach their maximum count, they are reset to their inactive state 0.

The durations reflected in the counters for transient handling have been chosen to be larger than pre-delay and allpass filter delay to suppress any traces of dispersed transients in the PS. Additionally, a hysteresis is implemented for avoiding on/off toggling that may happen e.g. when the input signal contains low frequency pulse trains with slight temporal variations of pulse distance around the inhibition time.

The energy in current frame is calculated using the DS. The energy of the current frame is smoothed by a first order IIR lowpass filter with "forgetting factor" $\delta = 0.4$. Equation 5 illustrates how energy of the current frame, $E_{Tr}(n)$, is calculated from the energy of the previous frame, $E_{Tr}(n-1)$, and the DS, $X_{DS,k}(n)$. Equation 6 defines the calculation of the transient detection state parameter t_n.

$$E_{Tr}(n) = \delta \cdot (\sum_{k=4}^{K} X_{DS,k}\,(n)^2) + (1 - \delta) \cdot E_{Tr}\,(n-1) \quad (5)$$

$$t_n = \begin{cases} 1 & \text{if } E_{Tr}\,(n) > T \cdot E_{Tr}\,(n-1) \\ 0 & otherwise \end{cases} \quad (6)$$

3.5. Envelope Shaping

The following equations explain the t/F envelope shaping. For each time frame n, in the output $\hat{Y}_{PS}\,(n)$, each frequency bin of the PS is amplified or attenuated if it is weaker or stronger by a factor of $\beta = 1.5$ as compared to the DS. Equation 7 and Equation

5 illustrate the estimation of the energy of the current PS, $E_{p,k}(n)$, and of the current DS, $E_{d,k}(n)$, smoothed by a first order IIR lowpass filter with "forgetting factor" $\alpha = 0.4$. Equation 9 represents the t/F shaping process depending on the energy relations. Finally, the PS is scaled with a fixed factor $f = 1.1$ that was empirically determined to adjust the mean signal energy to match the DS energy.

$$E_{p,k}(n) = \alpha \cdot Y_{PS,k}(n)^2 + (1-\alpha) \cdot E_{p,k}(n-1) \quad (7)$$

$$E_{d,k}(n) = \alpha \cdot X_{DS,k}(n)^2 + (1-\alpha) \cdot E_{d,k}(n-1) \quad (8)$$

$$\hat{Y}_{PS}(n) = \begin{cases} f \cdot Y_{PS}(n) \cdot \sqrt{\frac{\beta \cdot E_{d,k}(n)}{E_{p,k}(n)}} & \text{if } E_{p,k}(n) > \beta \cdot E_{d,k}(n) \\ f \cdot Y_{PS}(n) \cdot \sqrt{\frac{E_{d,k}(n)}{\beta \cdot E_{p,k}(n)}} & \text{if } E_{d,k}(n) > \beta \cdot E_{p,k}(n) \\ f \cdot Y_{PS}(n) & otherwise \end{cases} \quad (9)$$

A 256-point Inverse Discrete Fourier Transform (IDFT) is performed on $\hat{Y}_{PS}(n)$ to convert the processed frequency bins back to 256 time domain samples. To prepare for overlap-add in the output signal assembly, the signal is windowed with the same sine window as used for analysis.

3.6. Output Signal Assembly

The decorrelated output is generated as illustrated in Equation 10. In regular decorrelator operation, when the hold counter is inactive, the two decorrelated output frames are obtained from the M/S representation by conversion to L/R stereo as seen from the sum and difference computation of $x_{Win}(m)$ and $y_{PS}(m)$, respectively. If a transient was detected and the hold counter is activated, the two output frames will carry an identical signal consisting of the windowed input signal weighted by a scaling factor. The scaling factor $\frac{1}{\sqrt{2}}$ adjusts the energy of the "transient part" $2 \cdot x_{Win}(m)$ to match the energy of the "non-transient part" that is obtained by combining two incoherent signals $x_{Win}(m)$ and $y_{PS}(m)$. The windowing (computationally efficient inherited from the overlapping DFT frames) ensures a smooth cross fade when switching between the regular and the transient operation modes.

$$\begin{cases} \left\{ \begin{array}{l} y_{Out,1}(m) = x_{Win}(m) + y_{PS}(m) \\ y_{Out,2}(m) = x_{Win}(m) - y_{PS}(m) \end{array} \right\} & \text{if } hold\ cnt\ inactive \\ \left\{ \begin{array}{l} y_{Out,1}(m) = \frac{2}{\sqrt{2}} \cdot x_{Win}(m) \\ y_{Out,2}(m) = \frac{2}{\sqrt{2}} \cdot x_{Win}(m) \end{array} \right\} & \text{if } hold\ cnt\ active \end{cases} \quad (10)$$

Finally, the windowed frames are re-combined in an overlap-add procedure with 128-sample overlap between subsequent processing frames.

4. COMPLEXITY

The decorrelator was designed for high perceptual quality at lowest possible computational complexity. Allpass filtering is implemented using very simple subband IIR filters. The application of pre-delay and allpass filtering is done in (subsampled) DFT domain. The analysis DFT of the input signal - the direct signal X_{DS} - can thereby be efficiently used for both filtering and as input to t/F-envelope shaping. The DFT windowing is co-used for transient handling controlled output signal cross fade as can be seen in Equation 10.

If a set of many mutually decorrelated output signals is desired, the entire set may advantageously share said analysis DFT and parts of the envelope shaping calculations, specifically the energy estimation according to Equation 8 and the transient handling processing according to Equations 5 and 6, since these are solely dependent on the common direct input signal $X_{DS}(n)$.

The exact complexity is determined by DSP hardware and on the DSP implementation, so here a pen-and-paper estimate is given. The following Table 1 provides an overview on the number of arithmetic operations of the different parts of the decorrelator. Operations (OP) are multiplications (MUL) or additions (ADD); Divisions (DIV) and square roots (SQRT) are assumed to equal 25 OP each (which is a very conservative assumption that might be too high on modern DSP hardware).

The processing takes place in 50% overlapped windowed 256 sample blocks. According to [19], a 256-point real-data FFT/IFFT consumes 3022 operations. All further operations in DFT domain are to be calculated on 129 complex subband signals, hence the factors of 129 and 2, respectively. To arrive at operations per sample, all numbers are normalized on the 128 sample time domain stride of the overlapped DFT.

The computational complexity of the decorrelator is a moderate 7.4 MOPS (Million Operations per Second).

5. QUALITY

Some properties of the new decorrelator design presented in this paper have been compared with the ones of the MPEG Surround (MPS) decorrelator [5] as a baseline. Figures 2 and 3 display plots of mean correlation of the two output signals, Figures 4 and 5 display plots of mean magnitude differences of the two output signals as a function of frequency for the MPS decorrelator and the new decorrelator design. The measurements were obtained using pink noise as an input signal. From the plots it is apparent that the remaining correlation of the output channels is always below 0.045 and the magnitude differences are always less than 0.3 dB in the relevant audio frequency range. So, the new decorrelator design is on par with the perfomance of the MPS decorrelator while avoiding the much higher delay and computational costs caused by the Quadrature Mirror Filterbanks (QMF) of the MPS decorrelator. Thorough expert listening confirmed a high overall perceptual quality and a transparent transient reproduction of the new design. Nevertheless, a formal perceptual evaluation will be done in the future.

6. APPLICATION IN MPEG-I

In MPEG-I Immersive Audio, sources with homogeneous spatial extent are modelled by a number of decorrelated point sources that are spatially distributed in the VR scene geometry. The necessary decorrelated point source signals are derived from one or more "original" input signals that are the direct signals input to the decorrelator. One example of such a source with spatial extent is the park fountain in the MPEG-I VR test scene depicted in Figure 6. In the MPEG-I Immersive Audio Reference Software, the

Table 1: *Decorrelator complexity estimation. Operations (OP) equal multiplications (MUL) or additions (ADD); Divisions (DIV) and square roots (SQRT) are assumed to count 25 OP each.*

	Operations (OP) per 128 sample frame	Sum of OP/frame
Analysis window (Equation 1)	$256 \cdot MUL$	256
256-point real-data FFT/IFFT	$2 \cdot FFT$ (see [19])	$2 \cdot 3022 = 6044$
4 Schroeder allpass filters on complex data (Equation 4)	$4 \cdot 129 \cdot 2 \cdot 2 \cdot MUL; 4 \cdot 129 \cdot 2 \cdot 2 \cdot ADD$	$32 \cdot 129 = 4128$
Energy envelope estimation (Equation 78)	$2 \cdot 3 \cdot 129 \cdot MUL; 2 \cdot 2 \cdot 129 \cdot ADD;$	$(6 + 4) \cdot 129 = 1290$
Transient detection (Equation 5)	$3 \cdot 126 \cdot MUL; 2 \cdot 126 \cdot ADD$	$(3 + 2) \cdot 129 = 645$
t/F Envelope shaper (Equation 9)	$2 \cdot 129 \cdot MUL; 1 \cdot 129 \cdot DIV; 1 \cdot 129 \cdot SQRT$	$(2 + 25 + 25) \cdot 129 = 6708$
Synthesis window	$256 \cdot MUL$	256
Overlap+add	$256 \cdot ADD$	256
Output assembly (Equation 10)	$2 \cdot 128 \cdot ADD$	256
Total		19839

	Total OP/sample	OP/sec@ $f_s = 48kHz$
Decorrelator	155	**approx. 7.4 MOPS**

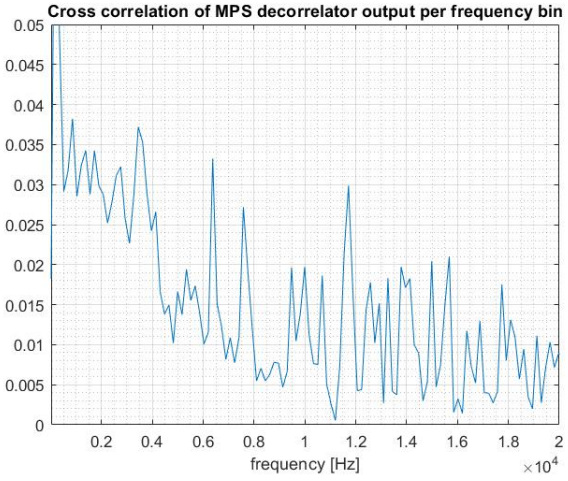


Figure 2: *Baseline MPS Decorrelator: mean correlation between decorrelator output channels as a function of frequency.*

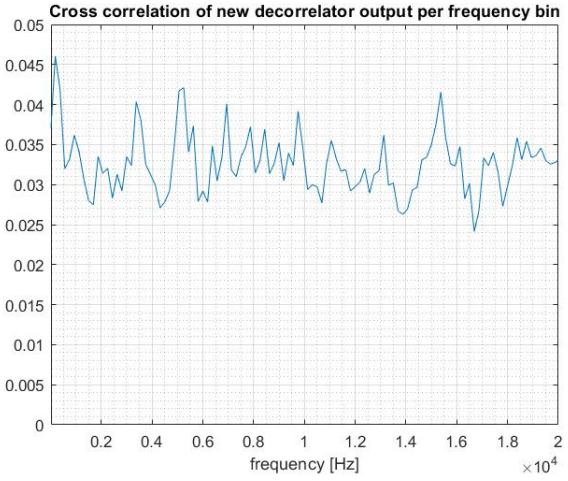


Figure 3: *New decorrelator design: mean correlation between decorrelator output channels as a function of frequency.*

presented decorrelator is available as an implementation in C++ [8].

7. CONCLUSIONS

In this paper, we describe the design considerations behind the decorrelator that is part of the upcoming MPEG-I Immersive audio standard and that is used for rendering sound sources with homogeneous extent. We present details of the actual implementation and provide computational complexity numbers showing a moderate total complexity of 7.4 MOPS for two decorrelated signals at 48kHz sampling rate. We also provide measurements on correlation and spectral magnitude differences of our new decorrelator design compared to the MPEG Surround (MPS) decorrelator as a baseline, showing that regarding these measurements the new design is on par with the baseline while minimizing processing delay and computational costs. We suggest that the new decorrelator design at hand might also be useful for integration in other spatial audio applications and audio effects.

8. REFERENCES

[1] Jürgen Herre and Sascha Disch, "MPEG-I Immersive Audio - reference model for the new virtual / augmented reality audio standard," *JAES Special Issue on Audio for Virtual and Augmented Reality*, 2023.

[2] Harald Bode, "History of electronic sound modification," *Journal of the Audio Engineering Society (JAES)*, vol. 32/10, pp. 730–739, 1984.

[3] Heiko Purnhagen, Jonas Engdegard, Jonas Roden, and Lars Liljeryd, "Synthetic ambience in parametric stereo coding," in *Audio Engineering Society Convention 116*, May 2004.

[4] Christof Faller and Frank Baumgarte, "Binaural cue coding-part ii:schemes and applications," *Speech and Audio Processing, IEEE Transactions on*, vol. 11, pp. 520 – 531, 12 2003.

[5] J. Herre, K. Kjörling, J. Breebaart, C. Faller, S. Disch, H. Purnhagen, J. Koppens, J. Hilpert, J. Rödén, W. Oomen,

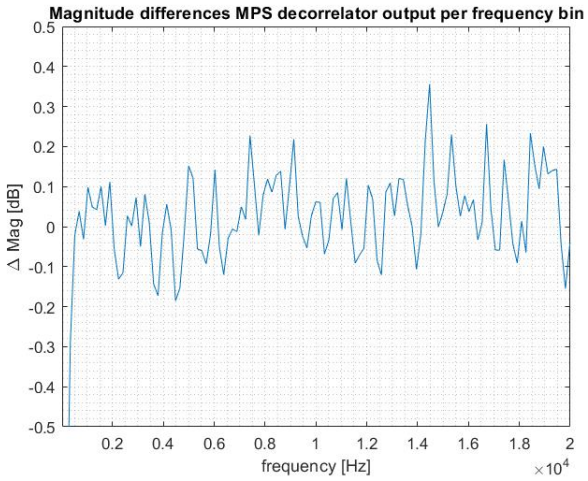


Figure 4: *Baseline MPS Decorrelator: mean magnitude differences between decorrelator output channels as a function of frequency.*

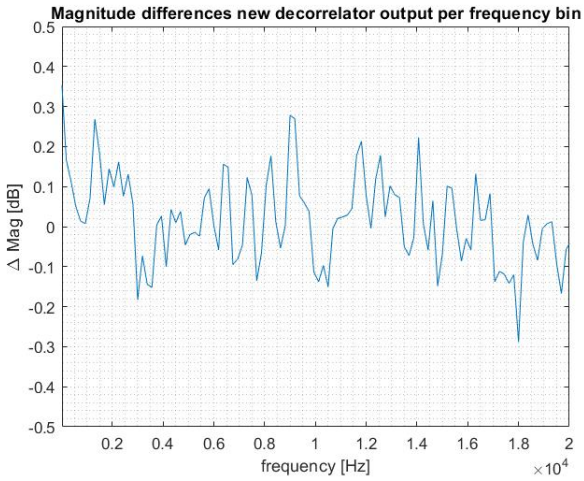


Figure 5: *New decorrelator design: mean magnitude differences between decorrelator output channels as a function of frequency.*

Figure 6: *MPEG-I test scene "Park" with the fountain's sound modeled as extended sound source.*

K. Linzmeier, and K. S. Chong, "MPEG Surround - the ISO/MPEG standard for efficient and compatible multichannel audio coding," *Journal of the AES*, vol. 56, pp. 932–955, nov 2008.

[6] Jean-Marc Jot, "Efficient models for reverberation and distance rendering in computer music and virtual audio reality," in *International Conference on Mathematics and Computing*, 1997.

[7] Schuyler R. Quackenbush and Jürgen Herre, "MPEG standards for compressed representation of immersive audio," *Proceedings of the IEEE*, vol. 109, no. 9, pp. 1578–1589, 2021.

[8] "https://www.mpeg.org/standards/mpeg-i/4/," 2023.

[9] Carlotta Anemüller, Alexander Adami, and Jürgen Jürgen Herre, "Efficient binaural rendering of spatially extended sound sources," *JAES Special Issue on Audio for Virtual and Augmented Reality*, 2023.

[10] Gary Kendall, "The decorrelation of audio signals and its impact on spatial imagery," *Computer Music Journal*, vol. 19, 12 1996.

[11] Maurice Boueri and Chris Kyriakakis, "Audio signal decorrelation based on a critical band approach," in *Audio Engineering Society Convention 117*, Oct 2004.

[12] Elliot Kermit-Canfield and Jonathan Abel, "Signal decorrelation using perceptually informed allpass filters," *Proceedings of the 19th International Conference on Digital Audio Effects (DAFx-16)*, 2016.

[13] Guillaume Potard and Ian Burnett, "Decorrelation techniques for the rendering of apparent sound source width in 3d audio displays," in *Proc. of the 7th Int. Conference on Digital Audio Effects (DAFx'04), Naples, Italy*, Oct 2004.

[14] Heiko Purnhagen, "Low complexity parametric stereo coding in mpeg-4," *Proc. of the 7th Int. Conference on Digital Audio Effects (DAFx'04), Naples, Italy*, 2014.

[15] Sascha Disch and Achim Kuntz, *A Dedicated Decorrelator for Parametric Spatial Coding of Applause-Like Audio Signals*, pp. 355–363, Springer-Verlag, January 2012.

[16] R. Penniman, "A general-purpose decorrelation algorithm with transient fidelity," *137th Audio Engineering Society Convention 2014*, pp. 99–107, 01 2014.

[17] Florin Ghido, Sascha Disch, Jürgen Herre, Franz Reutelhuber, and Alexander Adami, "Coding of fine granular audio signals using high resolution envelope processing (HREP)," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 701–705.

[18] Manfred Schroeder, "Natural sounding artificial reverberation," *Journal of the Audio Engineering Society*, vol. 10, no. 3, pp. 219–223, 1962.

[19] S. G. Johnson and M. Frigo, "A modified split-radix FFT with fewer arithmetic operations," *IEEE Transactions on Signal Processing*, 2006.

[20] Carlotta Anemüller, Oliver Thiergart, and Emanuël A. P. Habets, "A Data-Driven Approach to Audio Decorrelation," *IEEE Signal Processing Letters*, vol. 29, pp. 2477–2481, 2022, Conference Name: IEEE Signal Processing Letters.

Poster Session

FULLY CONDITIONED AND LOW-LATENCY BLACK-BOX MODELING OF ANALOG COMPRESSION

Riccardo Simionato and Stefano Fasciani

Department of Musicology
University of Oslo
Oslo, Norway
riccardo.simionato@imv.uio.no | stefano.fasciani@imv.uio.no

ABSTRACT

Neural networks have been found suitable for virtual analog modeling applications. Several analog audio effects have been successfully modeled with deep learning techniques, using low-latency and conditioned architectures suitable for real-world applications. Challenges remain with effects presenting more complex responses, such as nonlinear and time-varying input-output relationships. This paper proposes a deep-learning model for the analog compression effect. The architecture we introduce is fully conditioned by the device control parameters and it works on small audio segments, allowing low-latency real-time implementations. The architecture is used to model the CL 1B analog optical compressor, showing an overall high accuracy and ability to capture the different attack and release compression profiles. The proposed architecture' ability to model audio compression behaviors is also verified using datasets from other compressors. Limitations remain with heavy compression scenarios determined by the conditioning parameters.

1. INTRODUCTION

The unique timbre and sound coloring provided by analog circuits and their nonlinearities are still appealing to musicians and sound engineers. The digital emulation of vintage analog musical instruments and audio effects, known as Virtual Analog (VA), has been an active field of research and development for several years [1]. To date, a variety of digital emulations of audio analog devices have been introduced [2]. Recently artificial neural networks have been successfully employed also for VA audio effects [3, 4]. In particular, nonlinear distortion circuits [5, 6, 7] have been accurately modeled using architectures based on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Models suitable for real-world applications (i.e., low-latency and real-time computation) have also been proposed [8, 9, 10]. Challenges remain for nonlinear time-varying audio effects, where the state-of-the-art employs models requiring long audio segments and large networks [11], which are detrimental for latency and real-time computation. This work further investigates time-varying effects, in particular dynamic range compression. Audio compression is a nonlinear effect that reduces the dynamic range of the input signal by a given amount when this exceeds a given threshold [12]. Compression is usually applied to reduce the dynamic range of the signal. The gain reduction is time-varying according to variable attack and release times. In an early attempt to model an analog compressor/leveling amplifier [13], an artificial neural network, based on fully-connected layers, predicts the STFT of the compressed signal. The model, in this case, is large and works with long segments of the input signal. State-of-the-art modeling of audio compression is based on Temporal Convolutional Networks (TCN), which have been adapted to model analog compression [14], allowing real-time computation. However, input-output latency is still high because the model requires a long input segment ($\sim$1.5 seconds) to compute a segment of output accordingly to the receptive field of the network. Specifically, the lower bound latency is equal to the size of the receptive field, which in this case is between 101 and 1008 ms. Both works aimed at modeling the *TELETRONIX LA-2A leveling amplifier* with only two conditioning parameters: a binary switch to choose between the limit and compression modes and the peak reduction. Lastly, a gray-box model for the same device is presented in [15], where RNN and Multi-Layer Perceptron (MLP) networks are used in combination with traditional signal processing techniques. In particular, MLP is used to predict the parameters of the static compression curve based on the conditioning information, while RNN predicts the time parameters for the filters controlling the attack and release times. This model emulates only the compression mode (i.e., the switch parameter is fixed) and includes the peak reduction parameter as conditioning.

In our previous work, we focused on dynamic range compression as well but modeling another analog compressor device [16]. We investigated the use of Encoder-Decoder (ED) Long Short Term Memory (LSTM) based architectures to learn a static temporal profile of the TUBE-TECH CL 1B opto compressor (i.e., fixed attack and release time) conditioned to two control parameters: compression ratio and threshold. The architectures we used are relatively small with respect to network size and length of input segment, allowing low-latency real-time implementations of the model. Here we present improvements to the previously proposed architecture, and we use it for fully-conditioned modeling of the CL 1B, including variable attack and release time. Generally, LSTMs, when taken alone, struggle to learn long temporal dependencies, while CNNs require long receptive fields to model them. We show how combining LSTM and CNN in an ED architecture improves the modeling accuracy, enabling learning various compression amounts and temporal profiles given by different combinations of threshold, ratio, attack, and release values. The ED architecture can provide similar accuracy using networks with a relatively small number of parameters and a small input size compared to TCN-based works. The output gain parameter of CL 1B, and of other compressors as well, is excluded from conditioning because it controls an independent amplification stage applied after the compression, which is not influencing the com-

pression process. In analog compressor devices, the output gain can add characteristic harmonic distortion after the compression stage, but modeling distortion circuits is beyond the scope of this work. Such circuits can be modeled separately from existing techniques [8]. In addition, [13] showed that a single architecture can model compressors with different characteristics; thus we evaluate the proposed ED models with a set of heterogeneous compressors. The rest of the paper is organized as follows. Sec. 2 presents the various compressors used in this study and the associated datasets. Section 3 details the proposed architecture. An overview of the experiments carried out to validate the architecture is in Sec. 4, while the results are presented in Sec. 5. Sec. 6 concludes the paper.

2. DATASETS

In this section, we describe the various compressor datasets we use to train and test our models. Using multiple datasets collected from devices with different characteristics allows for verifying whether the proposed deep learning architecture can model audio compressors in general rather than only a specific unit.

2.1. CL 1B Compressor

The CL 1B[1] is an analog optical compressor manufactured by TUBE-TECH. In optical compression, a lighting-emitting element is fed with the audio signal, and this illuminates a light-sensitive resistor, also known as a photocell. The amplitude of the input signal determines the brightness of the element that, in turn, changes the resistance in the gain attenuation circuit. This device presents five variable control parameters: *ratio*, *threshold*, *attack*, *release times*, and *output gain*. We have built the CL 1B dataset[2] by recording data directly from the device as described in [16]. For this study, we have extended the dataset to include variable ratio, threshold, attack, and release time, but for each combination of the control parameters, we have limited the audio recording to 210 seconds. The output gain is fixed to 0 dB. Input signals include frequency sweeps (ranging from 20 Hz to 20 kHz), white noises with increasing amplitude (linear and logarithmic ramp), guitar, bass, and drums recordings (loop and single notes), and vocals. The output signal was recorded for 5 different values of each of the four parameters (equally spaced within the selected range), resulting in 625 combinations, which corresponds to a dataset of $\sim$ 36 hours recorded at 48 kHz. Threshold values ranges from 0 to -40 dBu, ratio values from 2:1 to 10:1, attack time from 0.5 to 300 ms, and release time from 0.05 to 10 seconds. The exact values of attack and release time are not marked on the device; therefore, we assume a linear range between the maximum and the minimum and the maximum indicated in the manual. The recordings associated with each combination of the control parameters are split into 21 parts of 10 seconds. From these, 50% of them are randomly removed, ensuring that no parameter combination is either over- or under-represented in the training set (i.e., random selection with uniform distribution against parameter combinations). Therefore, the complete training set includes $\sim$ 18 hours of recording associated with 625 different parameter combinations, out of which 20% is used for validation.

[1] http://www.tube-tech.com/cl-1b-opto-compressor/

[2] https://doi.org/10.5281/zenodo.6497085

2.2. Software Compressors

To further verify the extent to which the proposed architecture models audio compression, we have built three additional datasets from software compressors. We have selected compressors implemented as VST plugins because data collection can be completely automated, adapting an existing tool to work with audio effects [17]. The selected plugins are: the Softube FET Compressor[3], the PSP MicroComp[4], and the u-he Presswerk[5]. The first is a pure digital audio compressor, while the other two are VA devices. The input signal is the same described in Sec.2.1. For each software compressor, we have selected 4 variable parameters, summarized in Tab. 1, with functionality and range close to those selected for the CL 1B dataset. Also, in these cases, the 210 seconds output signals were recorded at 48 kHz for 625 different combinations of the control parameters (i.e., 5 equally spaced values within the selected ranges). The training, validation, and test set are built identically to that of CL 1B. Since FET Compressor has a fixed threshold, we have included in the set of variable parameters its input gain, which allows us to vary the relative attack/release point with respect to the input signal.

Table 1: *Selected variable parameters and respective ranges for the software compressors.*

VST	Parameter	Values
MicroComp	ratio	[2:1, 12:1]
	threshold	[0, -30] dB
	attack time	[0.5, 300] ms
	release time	[0.05, 3.30] s
Fet Compressor	ratio	[2:1, 10:1]
	input	[-6, +6] dB
	attack time	[20, 800] μs
	release time	[0.05, 1.1] s
Presswerk	ratio	[2:1, 10:1]
	threshold	[-10, -40] dB
	attack time	[0.1, 150] ms
	release time	[0.015, 2.5] s

2.3. LA-2A Leveling Amplifier

Teletronix LA-2A Leveling Amplifier[6] is an analog limiter/compressor, whose optical gain reduction works similarly to the CL 1B. The LA-2A is used in all previous works on black box modeling of analog compression [14, 15]. We consider this device in our experiments to allow a performance comparison with the state-of-the-art, although the LA-2A has a fundamental difference from other compressors used in our study. The LA-2A does not present variable attack and release time that users can fix using dials. Instead, the LA-2A presents an average attack time of 10 ms and a multi-stage release. The duration of the first stage is 0.06 seconds, while the second stage of release is controlled by the photocell's memory, which depends on the brightness and time the light-emitting has been on. The duration of the second stage ranges from 0.5 to 5 seconds for the complete release. This implies

[3] https://www.softube.com/fet-compressor-mk-ii

[4] https://www.pspaudioware.com/products/psp-mastercomp#psp-microcomp

[5] https://u-he.com/products/presswerk/

[6] https://www.uaudio.com/hardware/la-2a.html

that if the compression is heavy and/or the signal has been above the threshold for a long time, the LA-2A's release will be slower. Therefore the attack and release times of the LA-2A are unknown a priori but depend on the past input signal. The device presents an *output gain*, a switch controlling if the device is operating in a *limit* or *compression* mode, and a *peak reduction* that controls the amount of compression to apply. The LA-2A dataset[7] [13] contains approximately 20 hours of recordings at 44.1 kHz of the device fed with various acoustic and synthetic instruments, percussive clips, excerpts of musical pieces, tones, and noise bursts. The dataset includes the variations of two control parameters (in total 20 combinations): the binary switch that sets the device in either compress or limit mode and the peak reduction parameter that controls the amount of compression as a function of the input level. We use the same training, validation, and test split included in the dataset, representing 80%, 15%, and 5%, respectively. Recordings taken for each combination of the control parameters are not identical in both duration and contents.

3. PROPOSED ARCHITECTURE

The architecture we propose was developed by experimenting primarily with the CL 1B dataset, starting from the ED model we investigated in our previous work [16]. The encoder processes n past sample of the input sequence together with the additional conditioning parameters and returns its final internal states and output. The output is discarded while the internal state is passed to the decoder as its initial internal state. The decoder learns to predict the target sample at each time step, given the input sample at the current time step and a number n of past input samples. The improved architecture we propose in this paper is illustrated in Fig. 1. The input signal and conditioning parameters are fed to the network separately. In order to reduce the model complexity of the encoder, we replaced the LSTM layer with a 1D convolutional layer, which provides a representation of the signal's past information using fewer trainable parameters. Control parameters are fed to the encoder through a simpler fully-connected layer. The decoder still includes an LSTM layer, but it works on a larger input-output audio segment rather than on single samples. The architecture uses as input a segment of $2w$ past sample of the input signal x, which we split into two halves: one labeled as "far" past ($[x_{-2w}, ..., x_{-w}]$) and one labeled "recent" past ($[x_{-w}, ..., x_0]$). The "far" past represents the input for the encoder, and the "recent" past is the input for the decoder, both of size w. The encoder output, representing the internal states of the LSTM layer in the decoder, must match the LSTM number of neurons u. The LSTM layer is followed by a fully-connected layer with *sigmoid* activation function. Finally, at the output, we have another fully-connected layer with, in our case, a number of neurons of the same decoder input segment, where each neuron predicts an audio sample. With this architecture, the size of the output layer, o, can be reduced to predict as little as one sample at a time, allowing a fine-grain accuracy at the expense of the computational cost. In our experiments, we used input and output signal segments of identical size to minimize the computational complexity of the model and to work similarly to most real-time audio stream processing applications, in which input and output buffer sizes are identical. Lastly, preliminary experiments showed that kernels of the convolutional layer with identical size to the encoder input size led to better accuracy. In addition, this choice simplifies the architecture as no transformation of the convolutional layer's output is needed to match the dimensionality of the LSTM internal state. Conditioning parameters compose the input vector for the fully-connected layer in the encoder, as illustrated in Fig. 1. Before feeding the networks, the conditioning values are normalized between $[0, 1]$. The output of the fully-connected layer is added to the output of the convolutional one in order to compute the states to give to the decoder. The internal state size has to match the number of units of the LSTM layer; for this reason, the number of units is the same for all the layers.

[7] https://zenodo.org/record/3824876

4. EXPERIMENTAL DESIGN

4.1. Models

As stated in previous sections, our objective is to achieve an accurate black-box modeling of audio compression using deep-learning models with low latency and low computational complexity. We limit our investigation to ED models with w set to $(16, 32, 64)$, representing the number of input samples for both the encoder and decoder. The size w is also the intrinsic audio latency of our architecture. Our models work with an overall input segment size of $2w$ samples. Therefore, in a hypothetical real-time audio stream processing application, the overall latency of the system is $5w$ samples ($4w$ of latency for the double input-output buffering). To contain the computational cost of the models' inference, we take two approaches. First, we limit the number of trainable parameters to $\sim$ 100k at most, and therefore we investigate only models with $(32, 64, 128)$ numbers of units u. Second, we minimize the frequency at which the inference has to be executed to predict the stream of output samples. In particular, we consider only output sizes o equal to w, which requires only two predictions to fill the output audio block with size $2w$. Execution time and memory requirements can be further reduced using existing techniques such as pruning [18], quantization [19], tensor decomposition [20], knowledge distillation [21], and skip-RNN [22]. Low-latency for digital audio effects is essential for live-audio applications, while in production settings, modern digital audio workstations can automatically compensate for the plugin's latency to avoid temporal misalignment in multi-track mixing. However, automatic compensation works only within a limited range. For example, AVID Pro Tools can automatically manage up to 4096 samples of latency.

In order to compare our architecture with the state-of-the-art, we implemented a TCN network that can be trained with our CL 1B dataset. We adapted the best architecture from [14] to work at 48 kHz, taking as input $72,000$ samples (1.5 s) of the audio signal and predicting an output segment of $58,668$ samples ($\sim$ 1.2 s) with a receptive field of $13,332$ samples ($\sim$ 278 ms). This, together with the use of the LA-2A dataset in our experiments, allows a comprehensive cross-comparison of our architecture against the state-of-the-art. Comparisons with baseline architectures, such as simple LSTM or dense layers, are detailed in our previous work [16].

4.2. Training & Testing

All models are trained with a batch size of 128 and employing Adam [23] optimizer with gradient norm scaling of 1 [24]. ED models use an initial learning rate of 10^{-4}, while TCN model uses $3\ 10^{-4}$, as reported in [14]. Models are trained for 50 epochs except for the LA-2A case, in which we train models for 60 epochs

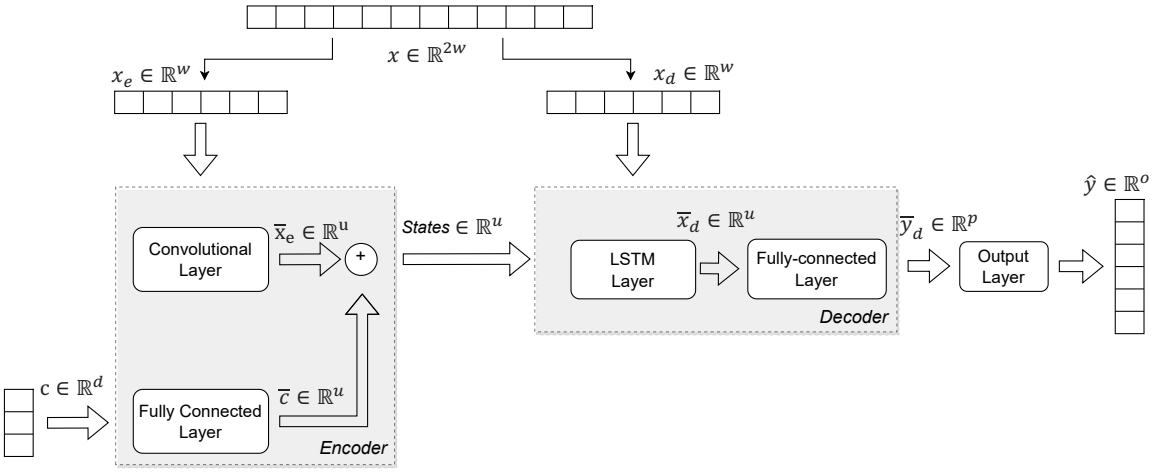


Figure 1: *Proposed architecture. Encoder consists of two layers: one taking the conditioning control parameters and the other one taking the past of the input signal. Control parameters are processed by a fully-connected layer, while the input is processed by the convolutional layer. The outputs of both layers are added to each other to compute the internal states that act as conditioning for the LSTM layer composing the decoder. The decoder takes the "current" input signal.* w *is the size in samples of the encoder and decoder input signals, and* u *is the number of units, representing as well dimension of the layer's outputs. The internal states dimension must match the number of units of the LSTM layer; for this reason, we selected the same number of units for all the layers. A fully-connected layer with* p *units and sigmoid activation function is placed after the LSTM layer, producing the decoder's output. This is fed to another fully-connected layer, including* o *neurons with linear activation function generating the* o *output samples of the model.*

to allow a comparison with [14]. Preliminary experiments varying input size $2w$ and the number of units u of the ED models use only half of the CL 1B dataset, still representing 625 combinations of conditioning parameters. The best-performing model, detailed in Sec. 4, is trained using the full dataset for 200 epochs in total. Results for the test loss are computed with the model's weights that minimize the validation loss throughout the training epochs. We have asserted the ability of the ED architecture to predict conditioning values never seen during the training in our previous investigation [16]. On the other hand, predicting abrupt and quick changes in the dynamic was challenging for the network. Similar behavior is also present in [15], where the prediction error increases with the increasing of the conditioning peak reduction value (i.e., in scenarios with larger dynamics change between input and output). Instead, conditioning values unseen during training do not determine a significant increase in the prediction error. For this reason, in this study, we test the generalization capability of the network using pairs of input signals and conditioning values unseen during the training phase, as detailed in the CL 1B dataset description.

4.3. Loss Function

As emerged in our previous work on CL 1B [16], error hikes during the attack phase of the compressor, triggered by fast changes in the input signal amplitude. To address this limitation, we investigate the use of different loss functions: we evaluate the Mean Absolute error (MAE), the Error-to-Signal Ratio (ESR), and a Short-Time Fourier Transform-based (STFT) loss function with different resolutions (window size of $[8, 16, 32]$ and hop size of $[2, 4, 8]$, respectively). As expressed in Eq. 1, the STFT-based loss function minimizes the spectral difference between the target and the predictions with a multi-resolution spectral loss. The component of the spectral loss with resolution m compares the two audio signals by summing the $L1$ differences between both their linear- and log-spectrograms. The models are trained for 50 epochs using different loss functions. Since absolute values returned by different loss functions are not comparable, results are qualitatively assessed by inspecting the waveforms of the predictions against the true outputs, focusing on the accuracy of the attack phase of the compression. Subsequently, we further explore combinations of the mentioned loss functions for training the model.

$$L_{STFT}(y, \hat{y}) = |||STFT_m(y)| - |STFT_m(\hat{y})|||_1 + ||\log(|STFT_m(y)|) - \log(|STFT_m(\hat{y})|)||_1 \quad \text{(Eq. 1)}$$

5. RESULTS & COMPARISONS

This section details the performance of the proposed architecture with the datasets detailed in Sec. 4. Comparisons with the state-of-the-art are included in Sec. 5.1, where the best TCN architecture from [14] is used to model the CL 1B; in Sec. 5.4 where our best ED architecture is used to model the LA2A used in all previous works on deep-learning modeling of dynamic range compression; and in Sec. 5.5, where latency and computational cost of our best ED model are compared with the best TCN architecture from [14]. Datasets, source code, trained models, audio examples, and additional figures available online[8].

[8] https://github.com/RiccardoVib/CONDITIONED-MODELING-OF-OPTICAL-COMPRESSOR

5.1. CL 1B

Tab. 2 details validation and test loss for models with different input segment size $2w$ and different numbers of units u. Accuracy is inversely proportional to the input size. Smaller sizes allow predicting output samples with a finer grain, resulting in smaller errors, although the network is fed with a smaller segment of audio samples. The accuracy is proportional to the number of units u, as larger networks appear to be beneficial to the model's accuracy. The most accurate model presents input segments $2w$ of 32 samples and 128 internal units u. Even if the best configurations resulted in 128 internal units, we used 64 hidden units for the final model. Therefore, we trained the model for 200 epochs, and we will refer to it for the figures and results detailed in the rest of this section. Associated validation and test loss are shown at the bottom of Tab 2. The model continues to learn at the increase of epochs, and conceivably losses could further decrease if training continues. The choice of 64 units is determined by our goal of minimizing computational complexity. A model with 128 units has almost four times the number of trainable parameters and brings improvements in the losses that do not justify the choice of having a bigger computational complexity. On the other hand, a bigger number of hidden units could lead to more remarkable improvements when increasing the number of training epochs. We noticed that the predicted signals include spurious tones generated by errors at the boundary of the output segments. We found that the frequency of such tones is equal to nF_r/o, where F_r is the sampling rate, o is the output size, and n is an integer representing the number of overtones, ranging from 1 to 6. The amplitude of the spurious tones does not exceed -70 dB; hence these are often masked by the compressed output audio signals within high-frequency contents. Cumulative figures on prediction error, such as averages across the entire dataset, are poorly informative with respect to the model's conditioned behavior. For this reason, we analyze the trend of prediction accuracy against the four conditioning parameters. The two colormaps in Fig. 2 display the errors for all combinations of attack and release time with fixed ratio and threshold, as well as for all combinations of ratio and threshold with fixed attack and release time. The fixed parameters were set at the middle of their range. To provide a fair and informative representation of the model's conditioned behavior, the MSEs are computed using the same audio signal for all parameter combinations, which includes 10 second of percussive and bass sounds taken from the test set. The various conditioning scenarios determine major changes in the dynamic range of the output signal, which should be taken into account when comparing the MSEs. To overcome this challenge and allow direct comparison of the errors, we compute the MSEs represented in Fig. 2 after normalizing target outputs to $[-1, +1]$ and applying the same normalization factor to the model's predictions. From the left image, it is evident that the prediction accuracy drops with the growth of attack and, in particular, release time. This reflects that with longer temporal dependencies, accurate prediction is more challenging. The right image shows that heavy compression scenarios (i.e., higher ratio and lower threshold) are also more challenging to be predicted accurately. Since the dataset was randomly split, for some parameter combinations, the overall percentage of signal above the threshold, which triggers the compressor, may not be identical between the training and test set. This, in turn, could be the reason why the error variations are not monotonic, in particular for the left colormap. However, we should also consider that the MSE ranges illustrated in Fig. 2 are extremely small, as visible from the associated color-bar values. Fig. 3 shows an example with three different settings of attack time, specifically for 0.5, 150, and 225 ms. Other parameters are fixed to ratio 6:1, threshold -30 dBu, and release time 0.05 s. It is visible how the model has learned different compression temporal profiles. The model handles the RMS envelope quite accurately, applying the gain reduction accordingly to the different attack time values. Lastly, Tab.3 compares the test losses (MSE and MAE) of the best TCN model from [14] and of our best ED model when trained with the CL 1B dataset. The losses after 50 epochs are reported, and the ED model shows better performance. In addition, from informal listening, it is evident that the TCN model introduces more audible artifacts than the ED model.

Table 2: *Validation and test loss (MSE) for ED model against different input segment sizes $2w$ and the number of units u. The test loss refers to unseen audio samples during training but seen conditioning values. The number of trainable parameters and epochs is also detailed. The models are trained for 50 epochs, using half dataset and MSE as the loss function. The bottom section of the table refers to the selected best model, which has been trained using the full dataset for up to 200 epochs.*

$2w$	u	Val Loss	Test Loss	Params	Ep.	Data
32	32	$1.45e^{-5}$	$1.12e^{-5}$	8,912	50	50%
-	64	$1.09e^{-5}$	$8.75e^{-6}$	24,912	-	-
-	128	$1.00e^{-5}$	$8.08e^{-6}$	81,488	-	-
64	32	$6.01e^{-5}$	$5.39e^{-5}$	10,976	-	-
-	64	$1.60e^{-5}$	$1.29e^{-5}$	28,000	-	-
-	128	$1.03e^{-5}$	$8.76e^{-6}$	86,624	-	-
128	32	$3.28e^{-4}$	$3.47e^{-4}$	15,104	-	-
-	64	$3.60e^{-4}$	$3.79e^{-4}$	34,176	-	-
-	128	$3.13e^{-4}$	$3.20e^{-4}$	96,896	-	-
32	64	$1.42e^{-5}$	$1.74e^{-5}$	24,912	50	100%
-	**-**	$\mathbf{6.92e^{-6}}$	$\mathbf{8.21e^{-6}}$	**-**	**200**	**-**

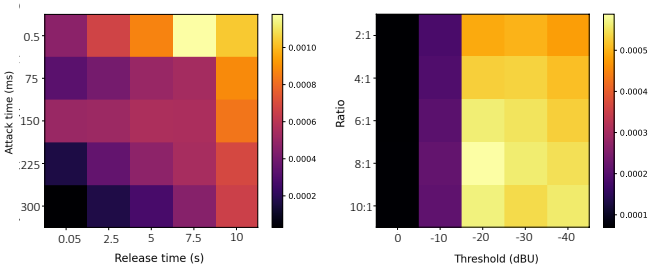


Figure 2: *ColorMaps of the test loss (MSE) for different combinations of conditioning parameters. Test errors for different values of attack and release times, with ratio and threshold fixed to -20 dBu and 6:1 (Left). Test errors for different values of ratio and threshold, with attack and release time fixed to the middle of their range (Right). Errors are computed after normalizing targets to $[-1, +1]$ and applying the same normalization factor to the predictions.*

5.2. Loss Functions

Fig. 4 shows prediction versus target output for a plucked bass example using models trained with four different loss functions. We

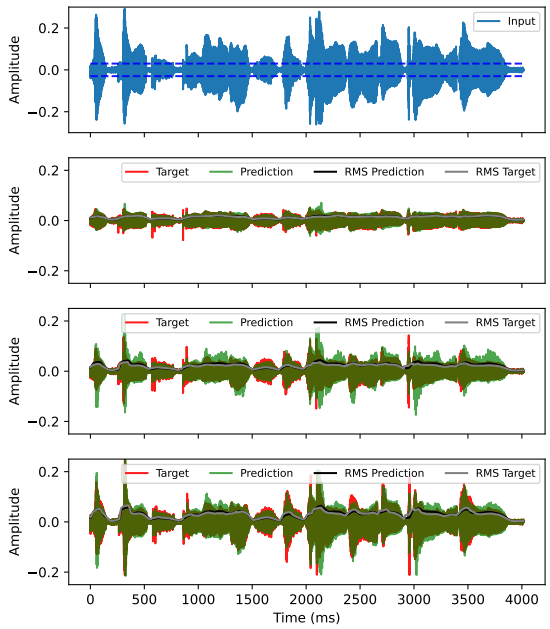


Figure 3: *Example of input waveform (first row) and associate output predicted versus target waveforms (second to fourth rows) for increasing values of attack time:* 0.5, 150, *and* 225 ms. *The ratio, threshold, and release time are set to* 6:1, −30 dBu, *and* 0.05 s, *respectively. The horizontal lines on the input waveform represent the threshold level.*

Table 3: *Test losses (MSE and MAE) for ED and TCN model. Losses refer to unseen audio samples during training but seen conditioning values. The number of trainable parameters and units is also detailed. The kernel size k, dilation factor d, and number of TCN blocks are reported for the case of TCN networks. The models are trained for* 50 *epochs with MSE as the loss function.*

Models	k	u	d	n	MSE	MAE	Params
TCN	13	32	10	4	$9.54e^{-4}$	$1.33e^{-2}$	51,464
ED-32	-	64	-	-	$1.74e^{-5}$	$1.93e^{-3}$	24,912

select a plucked bass sound because its amplitude envelope (sharp attack and decaying amplitude) is representative of scenarios that the model struggles to cope with, in particular during the initial phase of the compression. The plots refer to heavy-compressed scenarios (−40 dBu as threshold and 10:1 as ratio). In general, we found that models introduce more audible artifacts when these are accurate with onsets crossing the threshold and vice versa. When using MAE as the loss function, the models generate fewer artifacts but fail to learn the compression attack phase, applying the gain reduction instantaneously. On the other hand, when using MSE, the models learn the compressor attack and release phases more accurately but produce more audible artifacts. Using ESR or STFT-based loss functions provided poor performances, in particular, the latter one. The STFT-based's poor performance could be affected by the small output segment sizes that do not allow adequate frequency resolution. No combinations of loss functions led to advantages with respect to the attack phase. For this reason, we use the MSE only as the loss function for the training of our models, whose performance is detailed in previous sections.

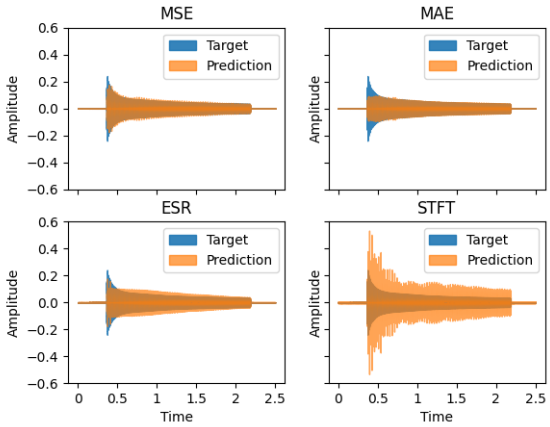


Figure 4: *Predicted waveform against the target for models trained using different loss functions: MSE, MAE, ESR, and STFT-based. The target example refers to a heavy-compressed plucked bass scenario (*−40 dBu *as threshold and 10:1 as ratio). These results are used to determine the influence of the loss functions on the predictions and are not representative of the accuracy of the final model.*

5.3. Software Compressors

Tab. 4 reports the validation and test loss associated with the software compressor datasets. These are obtained training for 50 epochs, the best model configuration derived from the CL 1B experiments, which uses 32 samples as the input segment $2w$, 64 hidden units u, and MSE as the loss function. The order of magnitude of the losses for all software compressors is similar to the CL 1B case, proving that the ED model is able to learn different compression profiles.

Table 4: *Validation and test losses (MSE) for ED model against software compressors. The models have* 64 *as the number of units* u*, and the input segment size* $2w$ *is equal to* 32 *samples. The models are trained for* 50 *epochs and use MSE as the loss function.*

Dataset	Val Loss	Test Loss
MicroComp	5.85^{-5}	$8.35e^{-6}$
Fet Compressor	$1.25e^{-4}$	$6.68e^{-5}$
Presswerk	$4.42e^{-5}$	$5.29e^{-5}$

5.4. LA-2A

Tab. 5 shows different test losses of our model trained on the LA-2A dataset for 60 epochs using different input segment sizes. The trend is similar to CL 1B, smaller input/output size turns in more accurate results. The MAE loss is also reported in the table in order to have a direct comparison with the TCN models in [14]. The number of hidden units is set to 64 to limit the number of trainable parameters, which are detailed in the table as well. Our proposed architecture trained on the LA-2A dataset presents an MSE two orders of magnitude greater than the one obtained with the CL 1B dataset. Although the ED model presents a higher loss,

it is still competitive with the TCN-based stat-or-the-art black-box model of the LA-2A, which presents a lower MAE equal to $7.66e^{-3}$ but uses almost twice as many trainable parameters and has a latency of 302 ms. Convolutional models generally converge quicker than LSTMs, which need a considerably larger number of training epochs to achieve similar losses. Hence this comparison based on an equal number of training epochs may be unfair to our ED model, which features an LSTM layer in the decoder. In addition, the LA-2A device presents faster attack and release time and fewer conditioning parameters than CL 1B. As stated before, the LA-2A has no tunable attack and release times, but these depend on the past state of the luminescent element. This can motivate the slight drop in accuracy of our architecture, which may not be able to infer the right compression temporal profile from the audio signal without explicit input data on attack and release time. The difference in the MSE can also be determined by differences between the datasets, such as type and levels of input signals, which have an impact on the overall amount and distribution of the audio compression.

Table 5: *Test loss (MAE, MSE, and ESR) for the ED model against different window input sizes. Both the encoder and decoder have* 64 *as the number of units. Input size refers to the number of total input samples used to compute the outputs; the encoder and decoder input sizes has to be considered half of this value. The number of trainable parameters is also reported. The models are trained for* 50 *epochs and use MSE as the loss function.*

$2w$	MAE	MSE	ESR	Params
32	$2.48e^{-2}$	$1.59e^{-3}$	$2.43e^{-1}$	24,656
64	$2.54e^{-2}$	$1.63e^{-3}$	$2.49e^{-1}$	27,744
128	$2.65e^{-2}$	$1.64e^{-3}$	$2.51e^{-1}$	33,920

5.5. Efficiency

The architecture we propose is designed to minimize latency and computational complexity, aiming at live audio applications. Considering the best model with 32 samples as input segment $2w$, the ED model has a latency of 16 samples, equivalent to 0.33 ms samples at 48 kHz sampling rate. The total latency in real-time audio stream processing application, including the double input-output buffering, is equal to 80 samples, equivalent to 1.66 ms. Tab. 7 reports the Floating Point Operations (FLOPs) required for each layer of the ED model. Breaking down the network, we have three Fully Connected (FC), one convolutional, and one LSTM layer. Each layer requires a different number of FLOPs, depending on the encoder and decoder input sizes w, number of conditioning parameters d, and output size o. The convolutional layer's FLOPs are influenced by the length of the kernel k and the number of filters f, which in our architecture are equal to w and u, respectively. In Tab. 7, we detail FLOPs for ED models with different input-output sizes and the respective intrinsic latencies, which are a key difference from the state-of-the-art TCN models. In [14], the most accurate TCN model has a latency of 302 ms, whereas experiments are detailed with other TCN models with latency as small as 101 ms. In our case, the best ED model presents a latency of only 0.33 ms. On the other hand, the most accurate TCN model is less computationally demanding since the inference predicts significantly longer segments of audio than our ED model. In particular, the inference requires $215,664$ FLOPs, equivalent to just 4 FLOPs per sample. Finally, we have also experimented using a TCN architecture that can deliver latency as low as our best ED model, using input and output segments of 32 and 16 samples, respectively. Results show that the TCN model trained on the CL 1B dataset is less accurate than the equivalent ED model. In particular, the MSE after 50 epochs is $4.20e^{-5}$ (TCN) versus $1.74e^{-5}$ (ED) as reported in Tab.2.

Table 6: *Number of Floating Point Operations (FLOPs) for the different layers of the ED model. d is the number of conditioning parameters, w is the input size, o is the output size, u is the number of hidden units, k is the kernel shape, and f is the number of filters.*

Layer	FLOPs
FC (conditioning)	2 x d x u
FC (decoder)	2 x u x u
FC (output)	2 x u x o
Convolutional	2 x f x k
LSTM	8 x (w + u) × u
Sigmoid function	4 x u

Table 7: *FLOPs for ED models with different input-output sizes (total FLOPs for inference, FLOPs per audio samples, and required GFLOPS for real-time implementation) and associated intrinsic latency (without accounting for the audio input-output buffering), considering* 48 kHz *sampling rate.*

$2w$	u	FLOPs	FLOPs/smp	GFLOPS	Latency
32	32	16,256	1,016	0,49	0.33 ms
32	64	52,480	3,280	1.57	-
32	128	186,368	11,648	5.59	-
64	32	22,912	716	0.34	0.66 ms
64	64	64,256	2,008	0.96	-
64	128	208,384	6,512	3.13	-
128	32	39,296	614	0.29	1.33 ms
128	64	90,880	1,420	0.68	-
128	128	255,488	3,992	1.92	-

6. CONCLUSION

We proposed a deep-learning architecture for black-box modeling of audio dynamic range compression. The model is in an Encoder-Decoder based on LSTM and convolutional layers. The encoder processes the near-past audio samples together with the values of the control parameters, which are encoded in a state helping the decoder to infer the output given the associated input samples. We demonstrated that the proposed architecture is able to model different types of compressors, three software, and two analog ones, with different compression characteristics. The model is designed to minimize latency and computational complexity and ease implementation in real-time audio stream processing applications. We conditioned the model against variations of the compressor's control parameters, including compression ratio, threshold, attack, and release time. The latter two change the system's temporal characteristics because the gain reduction is applied and removed in a different amount of time. Therefore, the architecture is able to learn nonlinear time-varying characteristics of the dynamic range compression, subject to attack and release time values, passed as

conditioning parameters to the network. The model is competitive with state-of-the-art works presenting similar accuracy while using shorter input signal segments. Prediction accuracy drops slightly with heavy compression settings. In particular drastic dynamic changes during the attack phase of the compression are the most challenging to predict and produce audible artifacts. We have also investigated the influence of different training loss functions, showing how the MSE is the most suitable to model the initial phase of the compression. Finally, we detailed the latency and computational complexity of the proposed architecture. Future work includes investigations to reduce the presence of audible artifacts, which are also clearly visible from spectrograms of the predicted output. Preliminary experiments suggest that ED models with smaller output sizes may significantly contribute to minimizing the presence of audible artifacts while slightly penalizing efficiency in terms of computational cost and intrinsic latency.

7. REFERENCES

[1] Jussi Pekonen and Vesa Välimäki, "The brief history of virtual analog synthesis," in *Proc. 6th Forum Acusticum. Aalborg, Denmark: European Acoustics Assoc.*, 2011, pp. 461–466.

[2] Jyri Pakarinen, Vesa Välimäki, Federico Fontana, Victor Lazzarini, and Jonathan S Abel, "Recent advances in real-time musical effects, synthesis, and virtual analog models," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, pp. 1–15, 2011.

[3] Marco A Martínez Ramírez, Emmanouil Benetos, and Joshua D. Reiss, "Modeling plate and spring reverberation using a dsp-informed deep neural network," in *ICASSP 2020-2020 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. IEEE, 2020, pp. 241–245.

[4] Marco A Martínez Ramírez, Emmanouil Benetos, and Joshua D. Reiss, "Deep learning for black-box modeling of audio effects," *Applied Sciences*, vol. 10, no. 2, pp. 638, 2020.

[5] Zhichen Zhang, Edward Olbrych, Joseph Bruchalski, Thomas J. McCormick, and David L. Livingston, "A vacuum-tube guitar amplifier model using long/short-term memory networks," in *SoutheastCon 2018*. IEEE, 2018, pp. 1–5.

[6] Eero-Pekka Damskägg, Lauri Juvela, Etienne Thuillier, and Vesa Välimäki, "Deep learning for tube amplifier emulation," in *ICASSP 2019-2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. IEEE, 2019, pp. 471–475.

[7] Thomas Schmitz and Jean-Jacques Embrechts, "Nonlinear real-time emulation of a tube amplifier with a long short time memory neural-network," in *Audio Engineering Society Convention 144*. Audio Engineering Society, 2018.

[8] Eero-Pekka Damskägg, Lauri Juvela, Vesa Välimäki, et al., "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Int. Sound and Music Computing Conf.(SMC-19)*, 2019, pp. 332–339.

[9] Alec Wright, Eero-Pekka Damskägg, Vesa Välimäki, et al., "Real-time black-box modelling with recurrent neural networks," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019.

[10] Julian D. Parker, Fabián Esqueda, and André Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019, pp. 2–6.

[11] Marco A Martínez Ramírez, Emmanouil Benetos, and Joshua D. Reiss, "A general-purpose deep learning approach to model time-varying audio effects," *Proc. Int. Conf. on Digital Audio Effects (DAFX-19)*, 2019.

[12] Udo Zölzer, Xavier Amatriain, Daniel Arfib, Jordi Bonada, Giovanni De Poli, Pierre Dutilleux, Gianpaolo Evangelista, Florian Keiler, Alex Loscos, Davide Rocchesso, et al., *DAFX-Digital audio effects, Second Edition*, chapter 5, Wiley & Sons, 2011.

[13] Scott Hawley, Nenjamin Colburn, and Stylianos Ioannis Mimilakis, "Profiling audio compressors with deep neural networks," *Journal of the Audio Engineering Society*, 2019.

[14] Christian J. Steinmetz and Joshua D. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *Audio Engineering Society Convention 151*. Audio Engineering Society, 2022.

[15] Alec Wright, Vesa Välimäki, et al., "Grey-box modelling of dynamic range compression," in *Proc. Int. Conf. on Digital Audio Effects (DAFX-22)*, 2022, pp. 304–311.

[16] Riccardo Simionato and Stefano Fasciani, "Deep learning conditioned modeling of optical compression," in *Proc. Int. Conf. on Digital Audio Effects (DAFX-22)*, 2022, pp. 288–295.

[17] Stefano Fasciani, "Tsam: a tool for analyzing, modeling, and mapping the timbre of sound synthesizers," pp. 129–136, 2016.

[18] Russell Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. of the IEEE Conf. on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[20] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "Tensor decomposition for compressing recurrent neural network," in *2018 Int. Joint Conf. on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *Conf. on Neural Information Processing Systems*, 2015.

[22] Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang, "Skip RNN: Learning to skip state updates in recurrent neural networks," *Int. Conf. on Learning Representations*, 2017.

[23] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *Int. Conf. on Learning Representations*, 2014.

[24] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *Int. Conf. on machine learning*. Pmlr, 2013, pp. 1310–1318.

A QUADRIC SURFACE MODEL OF VACUUM TUBES FOR VIRTUAL ANALOG APPLICATIONS

Riccardo Giampiccolo

DEIB, Politecnico di Milano
Milan, Italy
riccardo.giampiccolo@polimi.it

Stefano D'Angelo

Orastron Srl
Sessa Cilento, Italy
stefano.dangelo@orastron.com

Alberto Bernardini

DEIB, Politecnico di Milano
Milan, Italy
alberto.bernardini@polimi.it

Augusto Sarti

DEIB, Politecnico di Milano
Milan, Italy
augusto.sarti@polimi.it

ABSTRACT

Despite the prevalence of modern audio technology, vacuum tube amplifiers continue to play a vital role in the music industry. For this reason, over the years, many different digital techniques have been introduced for accomplishing their emulation. In this paper, we propose a novel quadric surface model for tube simulations able to overcome the Cardarilli model in terms of efficiency whilst retaining comparable accuracy when grid current is negligible. After showing the model capability to well outline tubes starting from measurement data, we perform an efficiency comparison by implementing the considered tube models as nonlinear 3-port elements in the Wave Digital domain. We do this by taking into account the typical common-cathode gain stage employed in vacuum tube guitar amplifiers. The proposed model turns out to be characterized by a speedup of $4.6\times$ with respect to the Cardarilli model, proving thus to be promising for real-time Virtual Analog applications.

1. INTRODUCTION

Vintage audio gear based on tube amplifiers is known for producing many of the distinctive sounds that characterize music genres such as blues, rock, and metal. Many professional electric guitar players prefer tube amplifiers due to the pleasing distortion they produce, resulting in a rich and warm sound. However, the high cost of reproducing tube-based analog audio gear and the decreasing supply of electrical components used in popular pieces of audio equipment have made these sounds inaccessible to many. As a result, Virtual Analog (VA) modeling has become a popular alternative as it seeks to create digital algorithms that accurately replicate the behavior of analog audio effects. This has resulted in many software implementations that serve as convenient digital counterparts for expensive, bulky, and fragile analog equipment, of which tube guitar amplifiers are a noteworthy example.

Two broad categories of models are used in VA modeling: "black-box" models that emulate the effect of a specific piece of gear by replicating its input-output behavior, using, for example, neural networks [1], Volterra series [2], or block-oriented Wiener-Hammerstein models [3]; "white-box" methods which emulate the reference circuit by solving the corresponding system of ordinary differential equations (or differential algebraic equations), using, for example, the Port-Hamiltonian approach [4], the State-space approach [5], or Wave Digital Filters (WDFs) [6, 7]. In this work, we are interested in deriving efficient "white-box" models of audio circuits with vacuum tubes, such that we can leverage the knowledge of schematics in order to address VA modeling.

Among vacuum tubes, triodes are the most widespread [8]. Before the advent of solid-state electronics, these devices were used in a variety of audio circuits, including preamplifiers, power amplifiers, equalizers, and compressors. In particular, the basic construction of a triode vacuum tube consists of three main components [9]: a cathode, a grid, and an anode (also known as plate). The cathode is a heated filament that emits electrons, while the anode is a positively charged electrode that attracts the electrons. The grid is a wire mesh located between the cathode and the anode that can control the flow of electrons between them. Over the years, many other designs of vacuum tubes have been introduced, mostly adding other grids between cathode and plate (e.g., pentodes).

Several models of vacuum tubes have been presented so far [8, 10]. In [11], tubes are modeled by combining linear filters and waveshapers, while the state-space approach and numerical methods are used in [12] and [13], respectively. Nonetheless, numerous implementations make use of WDFs. In fact, in the WD domain, circuits with up to one nonlinear element described by an explicit mapping can be solved with no use of iterative solvers, even using stable discretization methods (e.g., Backward Euler, trapezoidal rule, etc.), making WDFs interesting for real-time scenarios [14]. For instance, examples of WDF tube implementations can be found in [15], [16], [17], [18], [19], or [20], in which triodes are modeled combining Wave Digital (WD) principles to neural networks. However, most of the implementations available in the literature refer to two triode models: Koren [21] and Cardarilli models [22]. The first is based on a phenomenological model of the triode operation, while the second provides a simple yet accurate representation of triodes using a set of analytical expressions that describe the tube transfer function.

In this paper, we propose a new quadric surface model of vacuum tubes which is characterized, at the same time, by high efficiency and accuracy as long as grid current is negligible. Starting

from the generic equation describing quadric surfaces and considering a grid current equal to zero, we obtain a multivariate quadric equation able to outline the plate-to-cathode characteristics of triodes in active region employing just three parameters. These can be then fitted on datasheet or measurement data by applying a simple least squares method [23]. With respect to Koren model, the proposed approach is more accurate and almost as accurate as Cardarilli model if we assume a null grid current. Moreover, the model is characterized by a lower number of parameters, making the fitting operation more lightweight and simple. Although many different implementations of the proposed model can be pursued, we decided to realize it as a 3-port Wave Digital (WD) block, which we later employed for emulating the common-cathode gain stage typically found in tube guitar amplifiers.

The paper is organized as follows. In Section 2, we provide a background on triode modeling, by discussing both Koren and Caradilli models. We introduce the proposed quadric surface model in Section 3, and we draw a first comparison with the traditional models presented in the previous section focusing on their accuracy. Then, in Section 4, we address the emulation of a typical common-cathode gain stage considering the famous 12AX7 triode, and we run a comparison with Cardarilli model regarding the efficiency. Section 5 concludes this paper.

2. BACKGROUND ON TRIODE MODELING

Let us consider the vacuum tube depicted in Fig. 1(a). Such a device presents three terminals, and thus, it is typically known under the name of *triode*. The basic structure of a triode consists of an evacuated glass envelope containing three electrodes: a *cathode* "k" (or heated filament), a *grid* "g", and a *plate* "p" (also referred to as anode). Such a vacuum tube is the ancestor of modern transistors; in fact, it functions as an electronic amplifier by controlling the flow of electrons between the cathode and anode via the electric field generated by the grid [10]. By modulating the voltage applied to the grid, the current flow between the cathode and anode can be varied, allowing for amplification of signals. A triode operates as a "normally on" device, meaning that electrons flow to the positively biased plate even when the grid voltage is at its quiescent value. As the grid becomes more negative with respect to the cathode, the anode current is progressively reduced. Hence, in order to prevent the anode current to be turned off, a constant voltage is typically applied to the cathode. Moreover, the bias ensures that the positive peaks of the signal do not drive the grid voltage close to the cathode one, which would result in nonlinear behavior and the generation of grid current. The grid voltage that completely blocks electrons from reaching the anode is, instead, called *cutoff voltage*. Finally, in order to obtain linear amplification, the voltage on the grid must remain above the cutoff voltage without exceeding the cathode potential.

Different models of triodes are available in the literature. Usually, such models rely on a current source for the plate current i_p that is dependent on both the grid-to-cathode voltage V_gk and the plate-to-cathode voltage V_pk [9]. In the following, we present two traditional models widely used for VA applications, i.e., Koren [21] and Cardarilli [22] models.

2.1. Koren Model

In [21], Koren presents a phenomenological model able to describe the behavior of triodes making use of a small amount of parameters. The model equations are designed such that the plate current $i_\mathrm{p} > 0$ whenever the plate-to-cathode voltage $V_\mathrm{pk} > 0$. In particular, Koren model is defined as follows

$$i_\mathrm{p} = \frac{{V_1}^k}{k_\mathrm{g1}} \left(1 + \mathrm{sgn}(V_1)\right) , \tag{1}$$

$$i_\mathrm{k} = -\, i_\mathrm{p} - i_\mathrm{g} , \tag{2}$$

where i_k and i_g are the cathode and grid currents, respectively, and

$$V_1 = \frac{V_\mathrm{pk}}{k_\mathrm{k}} \log\left(1 + \exp\left(k_\mathrm{k}\left(\frac{1}{\mu_\mathrm{k}} + \frac{V_\mathrm{gk}}{\sqrt{k_\mathrm{vb} + V_\mathrm{pk}^2}}\right)\right)\right) , \tag{3}$$

$$V_\mathrm{gk} = V_\mathrm{g} - V_\mathrm{k} , \quad V_\mathrm{pk} = V_\mathrm{p} - V_\mathrm{k} ,$$

whereas $\mathrm{sgn}(\cdot)$ is the sign function, μ_k is the amplification factor, and k, k_g1, k_k, and k_vb are real parameters. Moreover, V_p, V_g, and V_k are the plate, grid, and cathode voltages, respectively. Koren already provides the values of such parameters for modeling the main triodes found in audio circuitry [21], as well as fitting methodologies for obtaining implementations of custom triodes. The positive direction of tube currents and voltages are assumed to be compliant with Fig 1(a). Furthermore, Koren model entails a diode between grid and cathode; as a matter of fact, i_g is typically computed employing common diode models or approximated with a time-varying resistor dependent on the grid-to-cathode voltage, as explained in [24]. Finally, it is worth pointing out that such a model is derived assuming that plate and grid are mutually insulated [21].

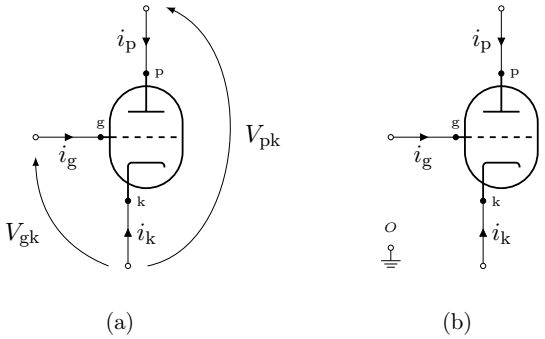


Figure 1: *(a) Circuit symbol of the triode reporting the current and voltage conventions employed in this paper. The plate is marked with "p", the grid is marked with "g", while the cathode is marked with "k." (b) Triode augmented with the reference node O.*

Table 1: *Parameters for modeling 12AX7 triode with Koren model [21].*

Parameter	Value	Parameter	Value
k	1.4	k_g1	1060 V^k/A
k_k	600	μ_k	100
k_vb	300 V^2	–	–

Table 2: *Parameters for modeling 12AX7 triode with Cardarilli model [22].*

Parameter	Value	Parameter	Value
G_0	$1.102\ \text{mA/V}^{2/3}$	μ_0	99.705
G_1	$15.12\ \mu\text{A/V}^{5/3}$	μ_1	$-22.98\ \text{kV}^{-1}$
G_2	$-31.56\ \mu\text{A/V}^{8/3}$	μ_2	$-0.4489\ \text{V}^{-2}$
G_3	$-3.286\ \mu\text{A/V}^{11/3}$	μ_3	$-22.27\ \text{kV}^{-3}$
h_0	0.6 V	V_{off}	-0.2 V
h_1	0	D	0.12
h_2	$0\ \text{V}^{-1}$	K	1.1
h_3	$0\ \text{V}^{-2}$	–	–

2.2. Cardarilli Model

The triode model developed by Cardarilli *et al.* [22] relies on a unique approach that combines physical and interpolative techniques. This model replaces the constant parameters typically used in classical formulations for plate current i_{p} and grid current i_{g} with splines that depend on the grid-to-cathode voltage V_{gk}. The resulting model, which is based on the current and voltage directions illustrated in Fig. 1(a), is defined as

$$i_{\text{p}} = G\sqrt{\left(V_{\text{gk}} + \frac{V_{\text{pk}}}{\mu} + h\right)^3}\,, \tag{4}$$

$$i_{\text{g}} = \begin{cases} \frac{i_{\text{p}}}{1+D\left(\frac{V_{\text{pk}}}{V_{\text{gk}}-V_{\text{off}}}\right)^K} & \text{if} \quad V_{\text{gk}} > V_{\text{off}} \\ 0 & \text{if} \quad V_{\text{gk}} \leq V_{\text{off}} \end{cases}\,, \tag{5}$$

$$i_{\text{k}} = -\,i_{\text{p}} - i_{\text{g}}\,, \tag{6}$$

where

$$G = \sum_{i=0}^{3} G_i V_{\text{gk}}^i\,, \quad \mu = \sum_{i=0}^{3} \mu_i V_{\text{gk}}^i\,, \quad h = \sum_{i=0}^{3} h_i V_{\text{gk}}^i\,, \tag{7}$$

and G is the perveance, μ is the amplification factor, h models the effects introduced by the initial electron velocity, while V_{off} is the threshold grid-to-cathode voltage above which grid current is present. The classical plate-current formulation shown in (4) plays a significant role in describing the behavior of triodes. In addition, for the case $V_{\text{gk}} > V_{\text{off}}$, the grid current expression of (5) is similar to the empirical relation presented in [9], where the numerical constants K and D help taking into account the contact potential effect.

Together, these formulas offer a valuable insight into the functioning of triodes and are useful tools for analyzing and modeling their behavior in electronic circuits. Moreover, in [22] a technique for calculating the values for all the parameters shown in (4), (5), and (7) based on curve fitting of datasheet data is presented. This approach helps to ensure the accuracy of the model, which has been shown to reproduce the transconductance behavior of triodes with high fidelity. Finally, unlike other triode models, Cardarilli model can provide highly accurate results even in saturation conditions.

Table 3: *Parameters for modeling 12AX7 triode with the proposed model.*

Parameter	Value	Parameter	Value
k_{p}	1.014×10^{-5}	k_{pg}	1.076×10^{-5}
k_{p2}	$5.498 \times 10^{-8}\ \text{V}^2$	–	–

3. PROPOSED QUADRIC SURFACE MODEL

In this section, we propose a new memoryless triode model based on a quadric surface approximation. The model is characterized by a lower number of parameters and by simpler mathematical operations with respect to those involved in the two traditional models of Section 2. On the other hand, for the sake of computational efficiency, it assumes null grid current, thus sacrificing accuracy when the grid voltage approaches and exceeds the cathode voltage.

The model is derived by exploiting the similarity between part of the typical triode characteristics $i_{\text{p}}(V_{\text{gk}}, V_{\text{pk}})$ and a quadric surface. In particular, we start by considering the general quadric surface equation [26]

$$\begin{aligned} &k_{\text{p2}}V_{\text{pk}}^2 + k_{\text{g2}}V_{\text{gk}}^2 + k_2 i_{\text{p}}^2 + k_{\text{pg}}V_{\text{gk}}V_{\text{pk}} + k_{\text{pk}}V_{\text{pk}}i_{\text{p}} + \\ &\quad + k_{\text{gk}}V_{\text{gk}}i_{\text{p}} + k_{\text{p}}V_{\text{pk}} + k_{\text{g}}V_{\text{gk}} + k_1 i_{\text{p}} + k_0 = 0\,, \end{aligned} \tag{8}$$

where k_{p2}, k_{g2}, k_2, k_{pg}, k_{pk}, k_{gk}, k_{p}, k_{g}, k_1, k_0 are real coefficients. In order to exploit such an equation for modeling the plate current i_{p}, we set $k_1 = -1$ and $k_2 = k_{\text{pk}} = k_{\text{gk}} = 0$ for avoiding dependences on i_{p} cross-products or second-order powers, yielding

$$i_{\text{p}} = k_{\text{p2}}V_{\text{pk}}^2 + k_{\text{g2}}V_{\text{gk}}^2 + k_{\text{pg}}V_{\text{gk}}V_{\text{pk}} + k_{\text{p}}V_{\text{pk}} + k_{\text{g}}V_{\text{gk}} + k_0\,. \tag{9}$$

It is worth pointing out that in (9), the terms V_{gk} and V_{pk} are present with all the powers up to the second order, allowing us to account for different contributions of both the plate-to-cathode voltage and the grid-to-cathode voltage whilst increasing the descriptive capability of the model.

Looking at the typical triode characteristics in datasheets, we recognize that the different curves obtained by varying V_{gk} are somehow tangent to the V_{pk}-axis in the $i_{\text{p}} - V_{\text{pk}}$ plane. Therefore, by imposing $i_{\text{p}} = 0$ and solving for V_{pk}, we obtain

$$V_{\text{pk}} = \frac{\sqrt{\Delta} - V_{\text{gk}}k_{\text{pg}} - k_{\text{p}}}{2k_{\text{p2}}} \tag{10}$$

with

$$\Delta = V_{\text{gk}}^2(k_{\text{pg}}^2 - 4k_{\text{g2}}k_{\text{p2}}) + V_{\text{gk}}\left(2k_{\text{p}}k_{\text{pg}} - 4k_{\text{g}}k_{\text{p2}}\right) - 4k_0k_{\text{p2}} + k_{\text{p}}^2\,, \tag{11}$$

and, by enforcing $\Delta = 0$, we constraint the vertex of the quadric to be on the V_{pk}-axis. If we then solve it for k_0, we obtain

$$k_0 = \frac{V_{\text{gk}}^2(k_{\text{pg}}^2 - 4k_{\text{g2}}k_{\text{p2}}) + V_{\text{gk}}(2k_{\text{p}}k_{\text{pg}} - 4k_{\text{g}}k_{\text{p2}}) + k_{\text{p}}^2}{4k_{\text{p2}}}, \tag{12}$$

which once substituted into (9) yields

$$\begin{aligned} i_{\text{p}} &= k_{\text{p2}}V_{\text{pk}}^2 + \frac{k_{\text{pg}}^2}{4k_{\text{p2}}}V_{\text{gk}}^2 + k_{\text{pg}}V_{\text{gk}}V_{\text{pk}} + k_{\text{p}}V_{\text{pk}} + \frac{k_{\text{p}}k_{\text{pg}}}{2k_{\text{p2}}}V_{\text{gk}} + \frac{k_{\text{p}}^2}{4k_{\text{p2}}} \\ &= i_{\text{p}}(V_{\text{pk}}) = i_{\text{pk}}\,. \end{aligned} \tag{13}$$

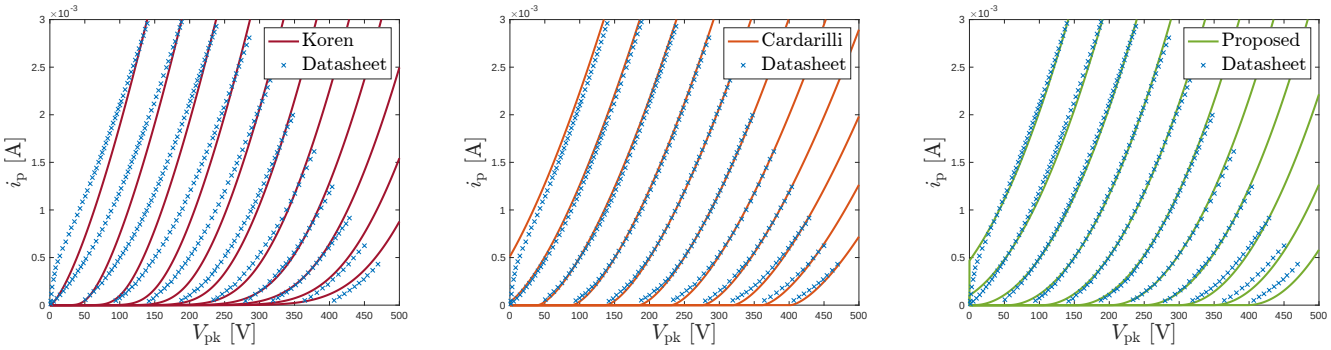


Figure 2: *Accuracy comparison between the $i_p - V_{pk}$ characteristics obtained with Koren model (a), Cardarilli model (b), and proposed model (c) and the one reported on the 12AX7 datasheet [25].*

For any given V_{gk}, (13) describes a parabola whose vertex lies on the V_{pk}-axis. In order to match datasheet curves [25], we only use the branch where the first derivative of i_p with respect to V_{pk}, i.e.,

$$i'_p = \frac{\mathrm{d}i_p(V_{pk})}{\mathrm{d}V_{pk}}\,, \tag{14}$$

is non-negative, and assume i_p to be equal to zero otherwise. Therefore, considering once again the conventions reported in Fig. 1(a) and the initial hypothesis on the grid current, we may write down the proposed quadric surface model as follows

$$i_p = \begin{cases} i_{pk} & i'_p \geq 0 \\ 0 & i'_p < 0 \end{cases}\,, \tag{15}$$

$$i_g = 0\,, \tag{16}$$

$$i_k = -i_p - i_g\,. \tag{17}$$

In any case, $V_{pk} \geq 0$ is also always assumed to hold true, as otherwise a region with negative static resistance would be introduced.

It is worth pointing out that the real coefficients of (13) can be computed starting from datasheets or measurements data, making use of common fitting techniques, such as a simple least squares method [23]. Moreover, as anticipated at the beginning of the section, the model presents just three parameters, which do increase the convergence performance of such fitting techniques, making it suitable to model with ease triodes available on the market. Then, the model mainly relies on a second-order formula which leads to closed-form solutions when equated to similar simple expressions (e.g., when coupled with linear systems), which makes it particularly suited for implementing fully-explicit white-box circuit models when such conditions are met. Finally, the quadric surface model is also able to describe pentodes in triode configuration.

In order to test the accuracy of the representation, we compare the $i_p - V_{pk}$ characteristics found on datasheets to that obtained by means of the proposed model. We do this also for the two traditional models presented in Section 2. Let us take into account the famous 12AX7 triode, which was employed in many different tube stages, as our reference vacuum tube. Tables 1, 2, 3 report the values of the coefficients to be used for modeling such a triode with Koren model, Cardarilli model, and proposed model, respectively. Fig. 2, instead, reports the results of such a comparison; the 11 curves present in each of the three plots are obtained by considering different values of V_{gk}. From the leftmost up to the rightmost, these curves are computed with V_{gk} equal to 0, -0.5, -1, -1.5, -2, -2.5, -3, -3.5, -4, -4.5, -5. In particular, in Fig. 2(a) the results of Koren model are reported, in Fig. 2(b) those of Cardarilli model, while in Fig. 2(c) those of the quadric surface model. In all the three plots, the data acquired from the 12AX7 datasheet [25] are marked with blue crosses. In Fig. 2, it is possible to observe that, in common operating regions [27], the proposed model is more accurate than Koren model and is almost as accurate as Cardarilli model, while featuring a lower number of parameters. In fact, the 15 parameters controlling Cardarilli model allows this to better grasp the nuances at higher V_{pk}. Nonetheless, we can state that the proposed model maintain, in general, a comparable accuracy since the of values of V_{pk} for which it starts to deviate from the measured characteristics are not often encountered in real applications.

3.1. 3-port Wave Digital Implementation

Although the proposed model can be implemented making use of different techniques, in this work we decided to use WDFs. The design procedure of WDFs entails a port-wise description of the reference circuit, whose elements and topological interconnections are realized as input-output blocks characterized by scattering relations. In order to accomplish such a description, WDFs make use of a change of variable, turning port voltage v and port current i (the so-called Kirchhoff variables) into incident wave a and reflected wave b with the addition of a free parameter per port Z called *port resistance* [6]. Such a free parameter constitutes a powerful mean for removing delay-free loops formed when Wave Digital (WD) blocks are interconnected [6, 28]. Among the different wave definitions present in the literature [6, 29, 30], the most widespread is that of voltage waves, i.e.,

$$\begin{aligned} a = v + Zi\,, \quad b = v - Zi\,, \\ v = \frac{a+b}{2}\,, \quad i = \frac{a-b}{2Z}\,, \end{aligned} \tag{18}$$

which is employed for deriving WD implementations of circuit elements [7, 31] and N-port topological junctions [32]. We aim at deriving a triode WD implementation of the quadric surface model

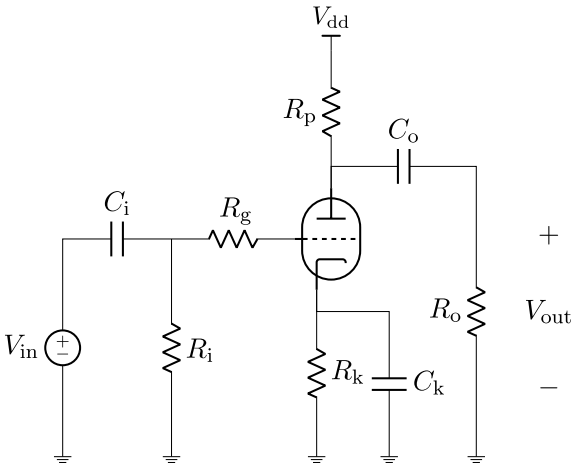


Figure 3: *Typical common-cathode gain stage found in tube amplifiers.*

such that it can be used in common WD structures. In particular, we here derive a 3-port triode model following the approach described in [18, 33]. Hence, we add a reference node O to the triode model, as shown in Fig. 1(b), and name the port between the p node and O as port 1, the port between the g node and O as port 2, and the port between the k node and O as port 3. Such a reference node will be coincident with the ground node when the model is exploited for circuit emulation. Then, each of these ports is characterized by the set of variables $\{a_x, b_x, Z_x\}$ with $x = 1, 2, 3$ being the port number.

If we apply the linear transformation in (18) to (16) and (17), and solving for b_2 and b_3 we readily obtain

$$b_2 = a_2\,, \tag{19}$$

$$b_3 = a_3 + \frac{Z_3}{Z_1}(a_1 - b_1)\,, \tag{20}$$

while doing the same with (15) and solving for b_1 gives us

$$b_1 = \pm\frac{\sqrt{\Delta_1}}{\sqrt{2Z_1 k_{\mathrm{p2}}\gamma}} - \frac{1}{4Z_1 k_{\mathrm{p2}}\gamma^2} - \eta\,, \tag{21}$$

where

$$\Delta_1 = \frac{1}{8Z_1 k_{\mathrm{p2}}\gamma^2} + a_1 + \eta\,, \tag{22}$$

$$\gamma = 1 + \frac{Z_3}{Z_1}\left(1 + \frac{k_{\mathrm{pg}}}{4k_{\mathrm{p2}}}\right)\,, \tag{23}$$

$$\eta = \frac{\beta + \frac{\alpha}{2}}{k_{\mathrm{p2}}\gamma}\,, \tag{24}$$

$$\alpha = k_{\mathrm{p}} + k_{\mathrm{pg}}\left(a_2 - a_3 - \frac{Z_3}{2Z_1}a_1\right)\,, \tag{25}$$

$$\beta = k_{\mathrm{p2}}\left(\frac{1 - \frac{Z_3}{Z_1}}{2}a_1 - a_3\right)\,. \tag{26}$$

Using the above equations, we can express

$$\begin{aligned} i'_{\mathrm{p}} &= k_{\mathrm{pg}}V_{\mathrm{gk}} + 2k_{\mathrm{p2}}V_{\mathrm{pk}} + k_{\mathrm{p}} = \\ &= k_{\mathrm{pg}}\left(\frac{Z_3}{2Z_1}(b_1 - a_1) + a_2 - a_3\right) + \\ &\quad + k_{\mathrm{p2}}\left(\frac{Z_3}{Z_1}(b_1 - a_1) + b_1 + a_1 - 2a_3\right) + k_{\mathrm{p}}\,, \end{aligned} \tag{27}$$

and hence

$$i'_{\mathrm{p}} \geq 0 \Leftrightarrow b_1 \geq -\eta\,, \tag{28}$$

assuming that all the involved model parameters and port resistances are positive. This should always be the case given how the model was constructed and that negative port resistances are related to incompatible sign conventions [34]. By comparing (28) with (21), it is evident that the first term in this last equation must be positive (necessary but not sufficient condition), hence the square root must be taken with the positive sign.

Since (21) is effectively the solution of the intersection of a line with a paraboloid pointing downwards in a 4-dimensional space, the cases in which $\Delta_1 < 0$, and thus in which such an intersection does not exist, can be interpreted as if V_{pk} or i_{pk} are too small or negative. In order to handle these occurrences, as well as the cases in which the resulting V_{pk}, i_{pk}, or i'_{p} are negative, we propose the following algorithm:

1. compute Δ_1 using (22);
2. if $\Delta_1 \geq 0$ then
   (a) compute b_1 using (21) taking the square root with positive sign;
   (b) if $i'_{\mathrm{p}} < 0$, computed according to (27), force $i_{\mathrm{p}} = 0$ (open circuit condition) by setting $b_1 = a_1$;

   otherwise, if $\Delta_1 < 0$ set $b_1 = a_1$ (i.e., $i_{\mathrm{p}} = 0$, open circuit condition);
3. if the resulting $V_{\mathrm{pk}} < 0$, force it to be $V_{\mathrm{pk}} = 0$ by setting

$$b_1 = \frac{(Z_3 - Z_1)a_1 + 2Z_1 a_3}{Z_1 + Z_3} \tag{29}$$

   according to (18) and (20).

Please notice that such a model is implemented in a fully explicit fashion, i.e., without the need for iterative solvers. Furthermore, it is worth pointing out that the quadric surface model can be also realized as a 2-port WD block, for example by employing the vector definition of waves in [30, 35], or, more generally, as a N-port with $N \leq 6$ following the approach discussed in [33]. Different WD implementations of the proposed model may lead to different advantages, potentially broadening the class of circuit models that can be implemented in an explicit fashion. For instance, the WD 2-port implementation by means of vector waves would allows us to explicitly realize triode stages with Miller capacitance or characterized by complicated topologies [9], which, instead, may not be realized by means of the 3-port implementation. However, when the number of multi-port nonlinear elements is ≥ 2, iterative methods must be employed for the solution of the WD structure [31].

4. EXAMPLE OF APPLICATION

Let us consider the common-cathode gain stage shown in Fig. 3, typically found in tube amplifiers. The same circuit has been taken into account in many other publications concerning the emulation

Table 4: *Values of the circuital components of Fig. 3.*

Parameter	Value	Parameter	Value
V_{dd}	250 V	R_p	100 kΩ
R_i	1 MΩ	R_o	1 MΩ
R_g	20 kΩ	R_k	1 kΩ
C_k	10 µF	C_i	100 nF
C_o	10 nF	–	–

of vacuum tubes, e.g., in [18]. With reference to the triode stage, the input signal is represented by the voltage generator V_{in}. Capacitor C_i and resistor R_i form a highpass filter which removes the DC (Direct-Current) component from the input if present. The cathode circuit, instead, is composed of a parallel connection between resistor R_k and capacitor C_k, and sets the signal-dependent biasing for the triode. The plate is connected to the DC voltage source V_{dd} via resistor R_p, which is responsible for the node "p" bias point. Then, the plate is also connected to the series between the DC decoupling capacitor C_o and load resistance R_o, whose voltage is taken as output variable. The values of the circuital parameters are reported in Table 4. The nonlinear amplification introduced by the triode together with the dynamic nature of the stage can generate a rather complex signal-dependent distortion on the output V_{out}.

We implement the common-cathode stage in the WD domain following the same approach and using the same WD structure employed in [18], i.e., by considering the 3-port WD block modeling the triode at the root of a connection tree. In order to test the accuracy of the WD implementation, we realize the same circuit in LTspice, which is a widespread freeware software for circuit simulations, together with the proposed triode model. We set $V_{in} = A \sin(2\pi k f_0/f_s)$, where $A = 2.5$ V is the amplitude, k is the sampling index, $f_0 = 1$ kHz is the fundamental frequency, and $f_s = 44.1$ kHz is the sampling frequency; in addition, we consider the duration of the sinusoidal input equal to 1 s. We select once again the 12AX7 as our reference triode, whose mdoel parameters are reported in Table 3. Fig 4 shows the output of the simulation, in particular the first three periods. The green dashed curve representing the LTspice simulation is overlapped with the continuous blue curve representing the WD simulation, pointing out the accuracy of the representation. Moreover, Fig. 4 also reports the results of the WD simulation obtained substituting the proposed triode model with the WD 3-port implementation of Cardarilli model (with no grid current) proposed in [18]. We select Cardarilli model over Koren model for drawing a comparison with state-of-the-art implementations of triodes since it is characterized by a higher accuracy, as shown in Fig. 2. Looking at Fig. 4, we can state that the two different realizations of the common-cathode stage are consistent, and that just a slightly different behavior can be spotted for the negative half-wave, which reflects the lack in restricting V_{pk} to non-negative values in the Cardarilli model.

As a final test, we perform an efficiency comparison simulating the same common-cathode stage in the WD domain employing both the proposed model and Cardarilli model. The two WD implementations, realized as GNU Octave scripts and available at `https://dangelo.audio/dafx2023-quadric.html`, differ only for the considered 3-port WD triode model. Under same operating conditions, we launch 20 runs of both algorithms on an AMD Ryzen 3 3200G processor, considering as input a mono guitar audio signal of duration 16 s and sampled at $f_s = 96$ kHz. We then measure the simulation time t_{sim}, and we average it over the number of runs for obtaining a fairer estimate. While the WD implementation with Cardarilli model takes on average 355.02 s, the WD implementation with the proposed model takes on average 76.55 s, showing thus a speedup of about $4.6\times$ over the traditional approach. It follows that the proposed quadric surface model is able to run much faster without compromising the accuracy of the representation, and, for this reason, can be a valuable triode model to be employed for real-time Virtual Analog applications.

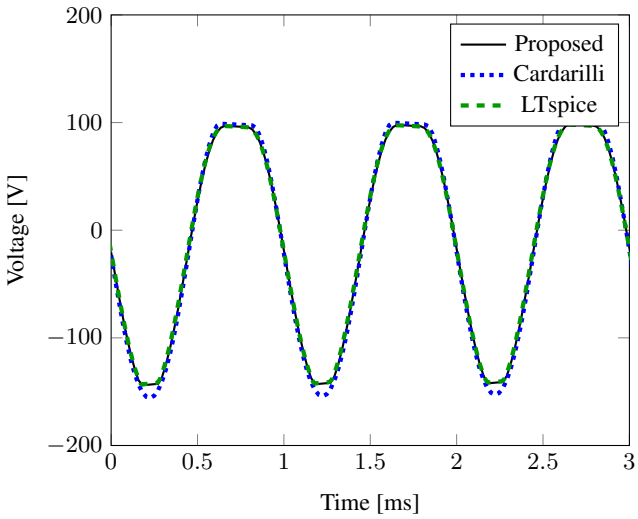


Figure 4: *Voltage V_{out} at the output of the triode stage shown in Fig 3. The continuous black curve represents the result of the WD simulation employing the proposed model, the dotted blue curve represents the result of the WD simulation employing Cardarilli model, while the dashed green curve represents the LTspice simulation employing the proposed model.*

5. CONCLUSIONS

In this paper, we proposed a novel triode model based on a quadric surface approximation. The model is characterized by only three parameters, involves simpler mathematical operations with respect to traditional models, such as Koren [21] and Cardarilli [22], and can lead to fully-explicit white box models. Considering the famous 12AX7 as a reference triode, we compared the plate current characteristics obtained with both the Cardarilli and Koren models and the proposed model to the one reported on its datasheet, showing that, when grid current is negligible, the new approach is more accurate than Koren's and almost as accurate as Cardarilli's even though the latter presents a number of parameters five times higher. Although the model can be realized in many circuit simulation methods, we decided to implement it as a 3-port WD block, which we later employed for the emulation of the typical common-cathode gain stage found in tube amplifiers. Under the same operating and modeling conditions, we drew an efficiency comparison between the quadric surface model and Cardarilli model, measur-

ing a $4.6\times$ speedup, which makes the proposed methodology very appealing for Virtual Analog applications.

Further work may concern the extension of the proposed model to other operating conditions by including, e.g., the tube diode between grid and cathode and the modeling of more complex vacuum tubes, such as tetrodes, pentodes, etc., by applying the same modeling methodology introduced in this paper but considering quadric hypersurfaces.

6. ACKNOWLEDGMENTS

S. D'Angelo would like to acknowledge Arturia for the time spent working on vacuum tube emulation back in 2015.

7. REFERENCES

[1] Eero-Pekka Damskagg, Lauri Juvela, Etienne Thuillier, and Vesa Välimäki, "Deep Learning for Tube Amplifier Emulation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, May 2019, pp. 471–475.

[2] Damien Bouvier, Thomas Hélie, and David Roze, "Phase-based Order Separation for Volterra Series Identification," *International Journal of Control*, vol. 94, pp. 2104–2114, Aug. 2021.

[3] Felix Eichas, Stephan Möller, and Udo Zölzer, "Block-Oriented Modeling of Distortion Audio Effects Using Iterative Minimization," in *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx-15)*, Trondheim, Norway, Nov. 2015.

[4] Antoine Falaize and Thomas Hélie, "Simulation of an Analog Circuit of a Wah Pedal: A Port-Hamiltonian Approach," in *Proceedings of the 135th Audio Engineering Society Convention*, Rome, Italy, May 2013, pp. 548–556.

[5] Gianpaolo Borin, Giovanni De Poli, and Davide Rocchesso, "Elimination of Delay-Free Loops in Discrete-Time Models of Nonlinear Acoustic Systems," *IEEE Transactions on Speech and Audio Processing*, vol. 8, pp. 597–604, 2000.

[6] Alfred Fettweis, "Wave Digital Filters: Theory and Practice," *Proceedings of the IEEE*, vol. 74, pp. 270–327, 1986.

[7] Alberto Bernardini, Paolo Maffezzoni, and Augusto Sarti, "Linear Multistep Discretization Methods with Variable Step-Size in Nonlinear Wave Digital Structures for Virtual Analog Modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, pp. 1763–1776, 2019.

[8] Jyri Pakarinen and David T. Yeh, "A Review of Digital Techniques for Modeling Vacuum-tube Guitar Amplifiers," *Computer Music Journal*, vol. 33, pp. 85–100, 2009.

[9] Karl Ralph Spangenberg, *Vacuum Tubes*, McGraw-Hill Book Company, 1948.

[10] Thomaz Chaves de A. Oliveira, Gilmar Barreto, and Alexander Mattioli Pasqual, "Review of Digital Emulation of Vacuum-Tube Audio Amplifiers and Recent Advances in Related Virtual Analog Models," *INFOCOMP Journal of Computer Science*, vol. 12, pp. 10–23, 2015.

[11] Udo Zölzer, *DAFX: Digital Audio Effects*, John Wiley and Sons, Ltd, Chichester, UK, Mar. 2011.

[12] Kristjan Dempwolf and Udo Zölzer, "A Physically-motivated Triode Model for Circuit Simulations," in *Proceedings of the International Conference on Digital Audio Effects, DAFx*, Paris, France, 2011.

[13] Ivan Cohen and Thomas Helie, "Measures and Parameter Estimation of Triodes, for the Real-Time Simulation of a Multi-Stage Guitar Preamplifier," in *129th Audio Engineering Society Convention 2010*, San Francisco, USA, 2010, vol. 1.

[14] Giovanni De Sanctis and Augusto Sarti, "Virtual Analog Modeling in the Wave-Digital Domain," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, pp. 715–727, 2010.

[15] Matti Karjalainen and Jyri Pakarinen, "Wave Digital Simulation of a Vacuum-Tube Amplifier," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toulouse, France, 2006, vol. 5.

[16] Jaromir Mačák and Jiri Schimmel, "Real-Time Guitar Tube Amplifier Simulation using an Approximation of Differential Equations," in *13th International Conference on Digital Audio Effects (DAFx)*, Graz, Austria, Sept. 2010.

[17] Jingjie Zhang and Julius O. Smith, "Real-time Wave Digital Simulation of Cascaded Vacuum Tube Amplifiers using Modified Blockwise Method," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Aveiro, Portugal, 2018, pp. 141–148.

[18] Stefano D'Angelo, Jyri Pakarinen, and Vesa Välimäki, "New Family of Wave-Digital Triode Models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, pp. 313–321, Feb. 2013.

[19] W. Ross Dunkel, Maximilian Rest, Kurt James Werner, Michael Jørgen Olsen, and Julius O. Smith III, "The Fender Bassman 5F6-A Family of Preamplifier Circuits-A Wave Digital Filter Case Study," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Brno, Czech Republic, Sept. 2016.

[20] Champ C. Darabundit, Dirk Roosenburg, and Julius O. Smith III, "Neural Net Tube Models for Wave Digital Filters," in *Proceedings of the 25th International Conference on Digital Audio Effects (DAFx)*, Vienna, Austria, Sept. 2022.

[21] Norman L. Koren, "Improved Vacuum Tube Models for SPICE Simulations," *Glass Audio*, vol. 8, pp. 18–27, 1996.

[22] Gian Carlo Cardarilli, Marco Re, and Leonardo Di Carlo, "Improved Large-Signal Model for Vacuum Triodes," in *Proceedings of the 2009 IEEE International Symposium on Circuits and Systems*, Taipei, Taiwan, May 2009.

[23] Kenneth Levenberg, "A Method for the Solution of Certain Non-linear Problems in Least Squares," *Quarterly of Applied Mathematics*, vol. 2, 1944.

[24] Jyri Pakarinen and Matti Karjalainen, "Enhanced Wave Digital Triode Model for Real-Time Tube Amplifier Emulation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, pp. 738–746, 2010.

[25] General Electric, "General Electric 12AX7 Electronic Tube - Datasheet," http://www.r-type.org/pdfs/12ax7.pdf.

[26] Michèle Audin, *Geometry*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[27] Ivan Cohen and Thomas Hélie, "Measures and Models of Real Triodes, for the Simulation of Guitar Amplifiers," *Acoustics 2012*, 2012.

[28] Augusto Sarti and Giovanni De Sanctis, "Systematic Methods for the Implementation of Nonlinear Wave-Digital Structures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, pp. 460–472, 2009.

[29] Kurt James Werner, Alberto Bernardini, Julius O. Smith, and Augusto Sarti, "Modeling Circuits with Arbitrary Topologies and Active Linear Multiports Using Wave Digital Filters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, pp. 4233–4246, 2018.

[30] Alberto Bernardini, Paolo Maffezzoni, and Augusto Sarti, "Vector Wave Digital Filters and Their Application to Circuits With Two-Port Elements," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, pp. 1269–1282, 2021.

[31] Alberto Bernardini, Enrico Bozzo, Federico Fontana, and Augusto Sarti, "A Wave Digital Newton-Raphson Method for Virtual Analog Modeling of Audio Circuits with Multiple One-Port Nonlinearities," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2162 – 2173, 2021.

[32] Alberto Bernardini, Kurt James Werner, Julius O. Smith, and Augusto Sarti, "Generalized Wave Digital Filter Realizations of Arbitrary Reciprocal Connection Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, pp. 694–707, 2019.

[33] Alberto Bernardini, Alessio E. Vergani, and Augusto Sarti, "Wave Digital Modeling of Nonlinear 3-terminal Devices for Virtual Analog Applications," *Circuits, Systems, and Signal Processing*, vol. 39, pp. 3289–3319, 2020.

[34] Stefano D'Angelo and Vesa Välimäki, "Wave-Digital Polarity and current Inverters and their Application to Virtual Analaog Audio Processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2012.

[35] Oliviero Massi, Riccardo Giampiccolo, Alberto Bernardini, and Augusto Sarti, "Explicit Vector Wave Digital Filter Modeling of Circuits with a Single Bipolar Junction Transistor," in *Proceedings of the 26th International Conference on Digital Audio Effects (DAFx23)*, Copenhagen, Denmark, Sept. 2023.

DESIGN OF FPGA-BASED HIGH-ORDER FDTD METHOD FOR ROOM ACOUSTICS

Yiyu Tan
Department of Systems Innovation Engineering
Iwate University
Morioka, Japan
tanyiyu@iwate-u.ac.jp

Guanghui Liu
Inflammatory Bowel and Immunobiology Research Institute
Cedars-Sinai Medical Center
Los Angeles, US
guanghui.liu@cshs.org

Xin Lu
Department of Systems Innovation Engineering
Iwate University
Morioka, Japan
luxin@iwate-u.ac.jp

Peng Chen, Yusuke Tanimu
Digital Architecture Research Center
National Institute of Advanced Industrial Science and Technology
Tokyo, Japan
{chin.hou,yusuke.tanimura}@aist.go.jp

ABSTRACT

Sound field rendering with finite difference time domain (FDTD) method is computation-intensive and memory-intensive. This research investigates an FPGA-based acceleration system for sound field rendering with the high-order FDTD method, in which spatial and temporal blockings are applied to alleviate external memory bandwidth bottleneck and reuse data, respectively. After implemented by using the FPGA card DE10-Pro, the FPGA-based sound field rendering systems outperform the software simulations conducted on a desktop machine with 512 GB DRAMs and a Xeon Gold 6212U processor (24 cores) running at 2.4 GHz by 11 times, 13 times, and 18 times in computing performance in the case of the 2nd-order, 4th-order, and 6th-order FDTD schemes, respectively, even though the FPGA-based sound field rendering systems run at much lower clock frequency and have much smaller on-chip and external memory.

1. INTRODUCTION

Room acoustic simulation exhibit numerical methods to model sound propagation phenomena in spatial and time domain, and are applied widely in many engineering and scientific applications, such as sound source localization [1-3], virtual reality [4-5], artificial reverberation [6], boundary impedance estimation [7], and so on. Many analysis algorithms have been proposed for sound field rendering in room acoustics, in particular, FDTD method, which has already become one of essential methods in room acoustics since it was introduced to analyse acoustical behaviour by O. Chiba et al., D. Botteldooren et al., and L. Savioja et al. [8-11]. FDTD method solves wave equation with a finite number of stencil points in a discretized sound space using numerical method, and provides much higher accuracy over other methods like geometric methods. The inherent problem of FDTD method is dispersion error, and oversampling in spatial grids is usually required to suppress the numerical dispersion. As a result, computation and memory demand are increased significantly. Although many works were done at algorithmic level to solve this problem, such as digital waveguide mesh topologies [12-15], explicit second-order accurate schemes [16], high-order explicit "large-star" schemes [17], and two-step explicit FDTD schemes with high-order accuracy [18-20], these approaches still suffer from high computational cost. In general, to solve wave equations using FDTD method, computing capability is increased as the fourth power of frequency and is proportional with the volume of a sound space [6], and the size of the required memory is third power of frequency. Given the auditory range of humans (20 Hz-20 kHz), analyzing sound wave propagation in a space corresponding to a concert hall or a cathedral (e.g. volume of 10000-15000 m^3) for the maximum simulation frequency of 20 kHz requires petaflops of computing capability and terabytes of memory. This requires computing systems to have huge computational capability and large memory bandwidth.

In recent years, graphic processing units (GPUs) and field programmable gate arrays (FPGAs) have been applied to speed up computation in sound field rendering because of their much higher parallel computational capability over traditional general-purpose processors [21-36]. In particular, latest FPGAs contain thousands of hardened floating-point arithmetic units, several Megabytes of on-chip block memories to cache data, and millions of reconfigurable logic blocks. These on-chip hardware resources may be applied to directly implement sound wave equations to accelerate computation in contrast with software simulations in GPUs and general-purpose processors. Furthermore, system data paths can be customized in accordance with the data flow of a sound field rendering system to improve computing performance. On the other hand, the high-order FDTD method provides more accurate approximation on the second-order partial derivative and reduces dispersion. In this research, an FPGA-based accelerator is developed to speed up computation in sound field rendering with the high-order FDTD method. The main contributions of this work are summarized as follows.

(1) A high-order FDTD method. The related formula is derived, including approximation of the second partial derivative using Lagrange polynomial interpolation, the updated equation of 4th-order and 6th-order FDTD schemes.
(2) Design and implementation of an FPGA-based sound field rendering system with the high-order FDTD method. Spatial and temporal blockings are adopted to reduce memory bandwidth requirement and reuse data.
(3) Performance evaluation and analysis based on the prototype machine. The proposed rendering system is designed using OpenCL and implemented using the FPGA card DE10-Pro.

* This work was supported by the JSPS KAKENHI Grant Number JP22K12123

Its performance is evaluated through analyzing sound propagation in a three-dimensional shoebox with dimensions being 16m×8m×8m, incidence being an impulse, and sampling rate of sound being 44.1 kHz. Compared with the software simulations performed on a desktop machine with 512 GB DDR4 RAMs and an Intel's Xeon Gold 6212U processor running at 2.4 GHz, the proposed rendering systems speed up computation by 11 times, 13 times, and 18 times in the 2nd-order, 4th-order, and 6th-order FDTD schemes, respectively.

The rest of this paper is organized as follows. The high-order FDTD schemes are introduced in Section 2. In Section 3, system design is described, including spatial blocking, temporal blocking, and system architecture. System performance of the FPGA-based prototype machine is presented in Section 4, followed by the conclusions drawn in Section 5.

2. HIGH-ORDER FDTD SCHEME

A high-order approximation in FDTD method gives more accurate approximation, reduces dispersion, and increases valid bandwidth [37]. In general, Lagrange interpolation [38] and Taylor series expansion [39] are applied for such approximation. In this research, the Lagrange polynomial method is used to approximate the second-order partial derivative in spatial domain.

2.1. Approximation of second-order partial derivative

In a 4th-order scheme, the Lagrange polynomial is assumed as equation (1) and pass through five adjacent points $(0, f_0),(\Delta, f_1),(2\Delta, f_2),(3\Delta, f_3),(4\Delta, f_4)$ along x axis. The Δ is the unit of x axis.

$$f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \tag{1}$$

Then we have

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ \Delta^4 & \Delta^3 & \Delta^2 & \Delta & 1 \\ 16\Delta^4 & 8\Delta^3 & 4\Delta^2 & 2\Delta & 1 \\ 81\Delta^4 & 27\Delta^3 & 9\Delta^2 & 3\Delta & 1 \\ 256\Delta^4 & 64\Delta^3 & 16\Delta^2 & 4\Delta & 1 \end{bmatrix} \begin{bmatrix} a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{2}$$

Equation (2) can be solved through matrix inversion, and the parameters a_0, a_1, a_2, a_3, a_4 are obtained as shown in Equation (3) [17][40].

$$\begin{cases} a_0 = f_0 \\ a_1 = \dfrac{-25f_0 + 48f_1 - 36f_2 + 16f_3 - 3f_4}{12\Delta} \\ a_2 = \dfrac{35f_0 - 104f_1 + 114f_2 - 56f_3 + 11f_4}{24\Delta^2} \\ a_3 = \dfrac{-5f_0 + 18f_1 - 24f_2 + 14f_3 - 3f_4}{12\Delta^3} \\ a_4 = \dfrac{f_0 - 4f_1 + 6f_2 - 4f_3 + f_4}{24\Delta^4} \end{cases} \tag{3}$$

Then the second derivative of $f(x)$ equals to Equation (4). In order to get a centered difference approximation, the middle point $(2\Delta, f_2)$ of the five adjacent points are chosen to approximate the second derivative, which is shown in equation (5).

$$f''(x) = \frac{f_0 - 4f_1 + 6f_2 - 4f_3 + f_4}{2\Delta^4}x^2 + \frac{-5f_0 + 18f_1 - 24f_2 + 14f_3 - 3f_4}{2\Delta^3}x + \frac{35f_0 - 104f_1 + 114f_2 - 56f_3 + 11f_4}{12\Delta^2} \tag{4}$$

$$f''(2\Delta) = \frac{-f_0 + 16f_1 - 30f_2 + 16f_3 - f_4}{12\Delta^2} \tag{5}$$

Thus, the approximated parameters for the second derivative is $(-\frac{1}{12}, \frac{4}{3}, -\frac{5}{2}, \frac{4}{3}, -\frac{1}{12})$, and $(f_0, f_1, f_2, f_3, f_4)$ corresponds to the values of the points $(i-2, j, k), (i-1, j, k), (i, j, k), (i+1, j, k), (i+2, j, k)$ along x axis in a three dimensional Cartesian space, respectively. The similar derivation can be conducted for the 6th-order approximation, in which $f(x)$ is assumed to be $f(x) = a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ and seven adjacent points are required to solve the equation.

2.2. High-order FDTD scheme

Sound wave propagation in a cubic space is governed by the equation.

$$\frac{\partial^2 P}{\partial t^2} = c^2\left(\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} + \frac{\partial^2 P}{\partial z^2}\right) \tag{6}$$

where P denotes sound pressure, c is the speed in air, t is time, x, y and z are Cartesian coordinates in a three-dimensional space. To solve Equation (6), high-order approximation, such as equation (5) for the 4th-order approximation, is applied to approximate the second-order partial derivative instead of the second-order center difference method. In general, the high-order approximation in time domain increases memory requirement because more data at previous time steps are involved in computation while the high-order approximation in spatial domain introduces additional computations due to more neighbor grids are needed to update value of a grid. In order not to increase memory requirement but just increase computations of updating sound pressure of a grid, the second-order approximation in time domain and high-order approximation in spatial domain are applied on Equation (6), which are shown as follows.

$$\begin{aligned} \frac{\partial^2 P}{\partial t^2} &= \frac{P^{n-1}_{i,j,k} - 2P^{n}_{i,j,k} + P^{n+1}_{i,j,k}}{\Delta t^2} \\ \frac{\partial^2 P}{\partial x^2} &= \frac{-\frac{1}{12}(P^{n}_{i-2,j,k} + P^{n}_{i+2,j,k}) + \frac{4}{3}(P^{n}_{i-1,j,k} + P^{n}_{i+1,j,k}) - \frac{5}{2}P^{n}_{i,j,k}}{\Delta x^2} \\ \frac{\partial^2 P}{\partial y^2} &= \frac{-\frac{1}{12}(P^{n}_{i,j-2,k} + P^{n}_{i,j+2,k}) + \frac{4}{3}(P^{n}_{i,j-1,k} + P^{n}_{i,j+1,k}) - \frac{5}{2}P^{n}_{i,j,k}}{\Delta y^2} \\ \frac{\partial^2 P}{\partial z^2} &= \frac{-\frac{1}{12}(P^{n}_{i,j,k-2} + P^{n}_{i,j,k+2}) + \frac{4}{3}(P^{n}_{i,j,k-1} + P^{n}_{i,j,k+1}) - \frac{5}{2}P^{n}_{i,j,k}}{\Delta z^2} \end{aligned} \tag{7}$$

Letting $\Delta x = \Delta y = \Delta z = \Delta l$ and inserting Equation (7) into Equation (6), the updated equation for the 4th-order scheme is obtained and shown in Equation (8) [40], in which $\chi = c\Delta t/\Delta l$ is

the Courant number. A similar derivation can be conducted on the 6th-order scheme and Equation (9) is yielded to update sound pressure of a grid.

$$\begin{aligned} P^{n+1}_{i,j,k} = \chi^2[-\frac{1}{12}(&P^n_{i-2,j,k} + P^n_{i+2,j,k} + P^n_{i,j-2,k} + P^n_{i,j+2,k} \\ &+ P^n_{i,j,k-2} + P^n_{i,j,k+2}\ ) + \frac{4}{3}\ (P^n_{i-1,j,k} + P^n_{i+1,j,k} + P^n_{i,j-1,k} \\ &+ P^n_{i,j+1,k} + P^n_{i,j,k-1} + P^n_{i,j,k+1})] + (2 - \frac{15}{2}\chi^2)P^n_{i,j,k} - P^{n-1}_{i,j,k} \end{aligned} \tag{8}$$

$$\begin{aligned} P^{n+1}_{i,j,k} = \chi^2[\frac{1}{90}(&P^n_{i-3,j,k} + P^n_{i+3,j,k} + P^n_{i,j-3,k} + P^n_{i,j+3,k} \\ &+ P^n_{i,j,k-3} + P^n_{i,j,k+3}\ ) - \frac{3}{20}(P^n_{i-2,j,k} + P^n_{i+2,j,k} + P^n_{i,j-2,k} \\ &+ P^n_{i,j+2,k} + P^n_{i,j,k-2} + P^n_{i,j,k+2}\ ) + \frac{3}{2}(P^n_{i-1,j,k} + P^n_{i+1,j,k} \\ &+ P^n_{i,j-1,k} + P^n_{i,j+1,k} + P^n_{i,j,k-1} + P^n_{i,j,k+1})]\ -\ P^{n-1}_{i,j,k} \\ &+ (2 - \frac{49}{6}\chi^2)P^n_{i,j,k} \end{aligned} \tag{9}$$

Equations (8) and (9) show that sound pressures of the neighbor grids along axes at previous time steps are needed to update sound pressure of a grid. The neighboring grids are in six axial directions. Two and three neighbor grids are in each direction in the 4th-order and 6th-order schemes, respectively. Computing sound pressure of a grid requires 11 additions, 2 subtractions, 3 multiplications, and 14 memory accesses in the 4th-order FDTD scheme while it needs 17 additions, 2 subtractions, 4 multiplications, and 20 memory accesses in the 6th-order FDTD scheme. In addition, since the high-order and 2nd-order approximation are applied on the space domain and time domain, respectively, the proposed FDTD scheme has high-order accuracy in space domain while it remains 2nd accuracy in time domain. For example, the 4th-order FDTD scheme provides 4th-order accuracy in space and 2nd-order accuracy in time.

Stability condition and dispersion are important in the FDTD method. J. Mourik discussed the stability condition and dispersion of high-order FDTD method and claimed that the 4th-order scheme was the best in terms of valid bandwidth up to 16th-order scheme [17][41]. The valid bandwidth of the 4th-order scheme was about 1.5 times and 1.1 times of those of the 2nd-order and 6th-order schemes, respectively, and the valid bandwidth dropped a little bit along with every increase of the order after the 4th-order. The stability condition for the 4th-order and 6th-order FDTD schemes are $\chi \le 0.5$ and $\chi \le \sqrt{15/68}$, respectively. In addition, high-order FDTD boundary conditions were also investigated by J. Mourik [41]. To simplify system design and evaluation, boundary conditions are not discussed in this paper.

3. SYSTEM DESIGN

From Equations (8) and (9), sound pressures of grids at previous two continuous time steps (time steps n and n-1) are required to compute sound pressures of grids at time step n+1, and huge amounts of data are read from and written back to memory as the grid dimensions are increased. Therefore, it is impossible to store all data in the on-chip block RAMs of FPGA, which are about several Megabytes in size, to reduce data access overhead in the case of large sound spaces even though the size of on-chip block memories inside current FPGAs has been increased significantly. Instead, external on-board DDR4 DRAMs on the FPGA card, which are several Gigabytes in size are needed to store data during computing. Another challenge is how to reuse data and reduce memory bandwidth requirement. In this research, spatial blocking is introduced to reduce the required memory bandwidth between the computing engine and on-board memory, and temporal blocking is employed to reuse data and reduce data accesses to external memory.

3.1. Spatial Blocking

Spatial blocking is applied to reduce the required on-chip memory, and it is employed in many deep-pipeline implementations of stencil computation on FPGA [42-43]. As shown in Fig. 1(a), a large sound space with $Nx \times Ny \times Nz$ grids is decomposed into small spatial blocks and each spatial block has $Cx \times Cy \times Nz$ grids. A small spatial block is further partitioned into *x-y* planes along the *z* dimension (Fig. 1(b)). Computations are performed plane by plane in a spatial block while they are carried out along the *x* dimension in a plane. Equations (8) and (9) indicate that data values of three adjacent planes are required to calculate new results. In the current design, shift registers are introduced as on-chip buffers to stream in data. As illustrated in Fig. 1, *n* values of the plane *i+1*, all values of the planes *i-1* and *i* are firstly streamed into a shift register from external memory to compute sound pressures of grids on the plane *i*, the computing unit then fetches data from the shift register and computes sound pressures of *n* grids on the plane *i* concurrently. Then, the shift register is shifted right by *n* data, and another *n* new data are written into the head of the shift register while *n* old data are evicted from the tail at each clock cycle. When computations in a plane are completed, data in a new plane are streamed in the shift register and computation is moved to the next plane. This procedure is repeated until sound pressures of all grids in a spatial block are computed, and then computation is switched to the next spatial block. The shift-register-based buffer can be efficiently implemented by the on-chip block RAMs inside an FPGA.

Using shift register minimizes the size of on-chip buffer by only storing sound pressures of the needed grids in a spatial block. Furthermore, current FPGAs provide abundance of block RAMs, therefore, much larger on-chip buffers can be implemented to store sound pressures of grids in a large spatial block to speed up data access. In addition, to parallelize computation spatially and improve utilization efficiency of the external on-board memory bandwidth, data are coalesced, and computations are vectorized to calculate *n* grids concurrently through loop unrolling in each spatial block. If the dimension of a spatial block is $Cx \times Cy$ and *n* grids are computed in parallel, the depth of the shift register is calculated through Equation (10).

$$depth = 2 \times rad \times Cx \times Cy + n \tag{10}$$

where *rad* is the stencil radius and it is 1, 2, and 3 for the 2nd-order, 4th-order, and 6th-order FDTD schemes, respectively. In contrast, the depth of the shift register is $2 \times rad \times N_x \times N_y + n$ if the spatial blocking is not applied. During implementation on an FPGA, parts of the shift register will be replicated to support parallel accesses because of the limited number of ports in each block RAM unit. Such replication will require more block RAMs inside an FPGA.

Computing sound pressures of grids on boundary planes (front, real, right, and left) of a spatial block needs data from its neighbor spatial blocks. But these data are not read into the shift

register during the computations of current spatial block. To avoid data exchange between adjacent spatial blocks, overlapped blocking is applied and such grids on boundary planes of a spatial block are treated as internal grids of the related neighbor spatial blocks and computed later. The size of the overlapped parts of neighbor spatial blocks are linearized to the stencil radius *rad* and the dimension of a spatial block (*Cx* × *Cy*).

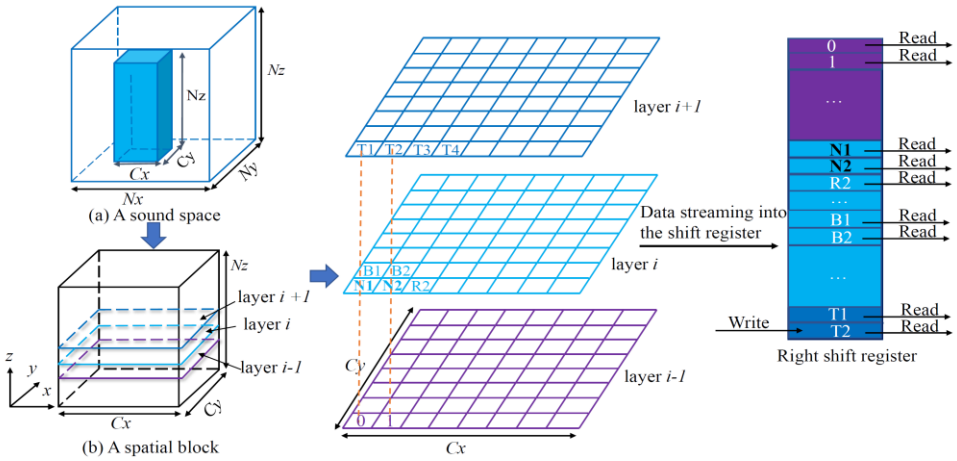


Figure 1: *Spatial blocking*

3.2. Temporal Blocking

Temporal blocking allows system to continuously compute sound pressures of grids of a spatial block at different time steps. Hence, data access to external memory is reduced. To implement temporal blocking, a computing kernel consisting of several replicated processing elements (PEs) is designed, and each PE computes sound pressures of grids in the same spatial block at different time steps. As shown in Fig. 2, several PEs are cascaded to compute sound pressures of grids in a same spatial block at continuous time steps. For example, PE_0 calculates sound pressures of grids at time step *n*. The computed results are sent to PE_1 and then PE_1 computes sound pressures of grids in the same spatial block at time step *n+1*. Such computation procedure is repeated until the final PE computes sound pressures of grids at time step *n+k-1*. Thus, access to external memory is reduced, and computation is sped up because sound pressures of a spatial block at several time steps are computed concurrently. Since computation of a given PE starts only after the outputs of the previous PE are available, computation in a PE is always behind its previous PE.

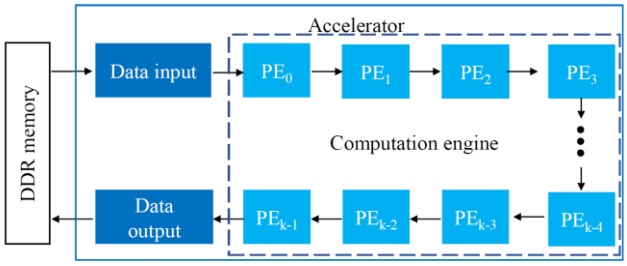


Figure 2: *System diagram*

3.3. System Design

The system diagram of the FPGA-based sound field rendering system is presented in Fig. 2, which consists of the Data input module, Computation engine, and Data output module. The Data input module streams data of a spatial block from the external DDR DRAMs on the FPGA card plane by plane, and feeds data to the computation engine. The computation engine consists of 16 PEs. Each PE computes sound pressures of grids in a spatial block at a time step, and all PEs are applied to compute sound pressures of grids in the same spatial block at continuous 16 time steps. The Data output module writes the computation results back to the external memory.

A PE computes sound pressure of a grid according to its position, incidence, and sound pressures of its neighbor grids at previous time steps. The computed results are sent to the neighbor PE except for the final PE, in which they are written back to the external memory through the Data output module. As shown in Fig. 3, a PE includes system controller, four buffers (shift_register_p1, shift_register_p2, shift_register_posi, and shift_register_incidence), and computing units. The functions of each module are described as follows.

- System controller. Each grid has an associated position flag, which will be applied to choose the updated equation in the computing unit. The system controller reads position flag and data values at previous time steps from the Data input module or a neighbor PE according to the computation flow, and writes them into the related shift registers like shift_register_p1, shift_register_p2, shift_register_posi, respectively. Then the computing unit computes sound pressures and sends the computation results to the neighbor PE except for the final PE, in which the computed results are written back to the external memory directly through the Data output module.
- Shift_register_p1, shift_register_p2, shift_register_posi, and shift_register_incidence. To compute sound pressures of grids at time step *n*, data values at time step *n-1* and *n-2* are streamed in the shift_register_p1 and shift_register_p2, respectively. In a PE, the input data data_p1 is directly passed to the next neighbor PE as the data values at time step *n-2* while the computed results are output to the next neighbor PE as the data values at time step *n-1*. The data values are exchanged through high bandwidth channels between neighbor PEs. The

position flags of grids are kept in the shift_register_posi. Since all PEs compute sound pressures of grids in a same spatial block, the position flag of a grid is same in all PEs, and a PE just passes the position_flag to its next neighbor PE. The incident data are stored in the shift_register_incidence.

- Computing unit. The computing unit fetches data from four buffers and compute sound pressures. It is designed based on the sound field rendering algorithm, namely Equations (8) and (9) for the 4th-order and 6th-order FDTD schemes.

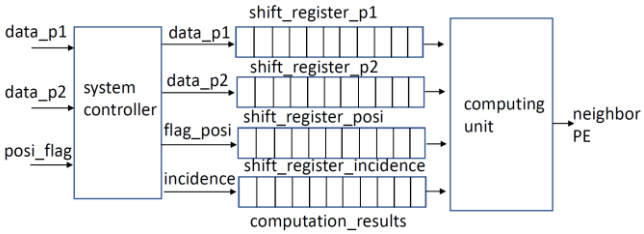


Figure 3: *PE structure*

4. PERFORMANCE EVALUATION

The proposed sound field rendering systems based on the 2nd-order, 4th-order, and 6th-order FDTD schemes were designed using the OpenCL programming language and implemented using the FPGA card DE10-Pro from Terasic Company [44]. The FPGA card contained a Stratix 10 SX FPGA (1SX280HU2F50E1VG) and 8 GB on-board external DDR4 DRAMs. To verify and estimate the performance of the developed sound field rendering systems, sound propagation in a three-dimensional shoebox with dimension being 16m×8m×8m was analyzed. The incidence was an impulse, and the number of the computed time steps was 32. As a comparison, relative counterpart systems were developed using the C++ programming language, and executed on a desktop machine with 512 GB DDR4 DRAMs and an Intel Xeon Gold 6212U processor (24 cores) running at 2.4 GHz. The OpenCL codes were compiled using the Intel FPGA SDK for OpenCL 19.1 while the reference C++ codes were compiled using the GNU compiler (version: 4.8.5) with the option -O3 and -fopenmp to use all 24 processor cores. During analysis, the sound speed was 340 m/s, sampling rate is 44.1 kHz, the Courant number χ was $\sqrt{3}/3$, $1/2$, $\sqrt{15/68}$ in the 2nd-order, 4th-order, 6th-order FDTD schemes, respectively, and all boundaries were clamped to 0, i.e. phase-reversing fully reflective boundaries. Data were single-precision floating point in both the FPGA-based rendering systems and software simulations. The development environment in the FPGA-based sound field rendering system and software simulation is shown in Table 1. As presented in Table 1, the memory size of external and on-chip memories in the FPGA-based system is much smaller than that of the desktop machine in the software simulation, and the FPGA system runs at much lower clock frequency over the desktop machine.

4.1. Hardware resource utilization

Table 2 presents the hardware resource utilization of the FPGA-based sound field rendering systems with the 2nd-order, 4th-order, and 6th-order FDTD schemes when the size of a spatial block is 128 × 128, the number of PEs is 16 in the computation engine, and the number of grids computed concurrently is 16. Equations (8) and (9) indicate that as the order of the FDTD scheme is increased, the number of operations are increased, more data are streamed in the shift registers, more DSP blocks, which are utilized to implement multipliers, are involved in computation, and more RAM blocks are required to implement the shift registers to store data during computing. From Equation (10), the utilized RAM blocks are significantly affected by the size of a spatial block. If the size of a spatial block is changed from 128 × 128 to 256 × 256 in the 2nd-order FDTD scheme, the number of utilized RAM blocks will be increased from 1785 to 5129. In addition, since the control of shifting and reading out data from the shift registers at a clock cycle is complicated in the sound field rendering system with the higher-order FDTD scheme, the system data path becomes more complex, and the clock frequency is decreased.

Table 1: *Development environment*

	FPGA	software simulation
computing unit	Stratix 10 SX	Intel Xeon Gold 6212U
# of cores	5760 DSP blocks	24 cores
frequency	about 350 MHz	2.4 GHz
on-chip memory	28.6 MB block RAMs	L1 cache: 1.5 MB
		L2 cache: 24 MB
		L3 cache: 35.75 MB
external memory	8 GB DDR4-2400	512 GB DDR4-2933
operating system	CentOS 7.2	CentOS 7.2
programming language	OpenCL	C++
compiler	Intel FPGA SDK for OpenCL 19.1	GNU compiler (version: 4.8.5)
fabrication	14 nm	14 nm

Table 2: *Hardware resource utilization*

orders	logic utilization	DSP blocks	RAM blocks	clock frequency (MHz)
2nd	269,159 (29%)	342 (6%)	1,785 (15%)	357
4th	293,001 (31%)	630 (11%)	3,764 (32%)	355
6th	335,237 (36%)	918 (16%)	4,309 (37%)	337

From Table 2, the hardware resources are not utilized efficiently in the current design. The logic blocks, DSP blocks, and RAM blocks are used by 29%, 6%, and 15% of the relative valid resources inside the FPGA, respectively. Thus, the size of the spatial block and the number of grids computed in parallel can be further increased in the current design. On the other hand, as the size of the spatial block and the number of grids computed in parallel are increased, the data path in the hardware system may become complicated, and clock frequency may be decreased, which will result in the degradation of computing performance. Therefore,

the size of the spatial block and the number of grids computed concurrently cannot be increased unlimitedly.

4.2. Computation time

When the size of the spatial block is 128 × 128, the number of PEs is 16, and the number of grids computed concurrently is 16, Table 3 presents the average rendering time at each time step in the FPGA-based sound field rendering systems and software simulations in the case of the FDTD schemes with different orders. Although the desktop machine in the software simulations runs at much higher clock frequency and has much larger external and on-chip memories than the FPGA-based sound field rendering systems, the FPGA-based sound field rendering systems speed up computation by 11 times, 13 times, and 18 times in the 2nd-order, 4th-order, and 6th-order FDTD schemes, respectively, over software simulations performed on the desktop machine. In the FPGA-based rendering system, sound pressures of grids at time steps *n* and *n-1* are stored into two independent DDR4 DRAMs, and they are fetched through two independent channels and streamed into the on-chip shift registers inside FPGA. The overhead to access data from the shift registers is usually one clock cycle. In contrast, all sound pressures are stored in external memory in the software simulations, and external memory is accessed frequently to fetch or write back data during computation. The data access is constraint by the memory bandwidth and the access overhead is very large. Although on-chip caches inside the processor may reduce the overhead of accessing data, their benefits to the computing performance improvement are limited as the grid dimension is increased. Moreover, data are reused through temporal blocking in the FPGA-based system, and sound pressures of a spatial block at 16 continuous time steps are computed in parallel. This further reduces data access to the external memory. All these lead to the performance improvement of computation in the FPGA-based sound field rendering system.

Table 3: *Rendering time per time step (s)*

orders	FPGA	software simulation
2nd	0.0486	0.5363
4th	0.0333	0.4458
6th	0.0238	0.4437

In the current performance evaluation, the Courant number is $\sqrt{3}/3$, $1/2$, $\sqrt{15/68}$ in the 2nd-order, 4th-order, 6th-order FDTD schemes, respectively. As the order of the FDTD scheme is increased, although the clock frequency of the FPGA-based sound field rendering system decreased a little bit, the computing time at each time step is decreased significantly because the grid dimension becomes smaller and the number of grids is reduced. But it is worth noting that different Courant numbers will impact upon the valid bandwidth of the outputs in each FDTD scheme.

4.3. Computational Throughput

The computational throughput stands for the number of grids updated per second at each time step and is calculated by using the following formula.

$$SP_{updated} = \frac{N_{grid}}{t_{time\_step}} \tag{11}$$

where N_{grid} is the number of grids, and t_{time_step} is the average computing time at a time step. Table 4 shows the computational throughput in the FPGA-based sound field rendering systems and software simulations in the case of the FDTD schemes with different orders. As shown in Table 4, the FPGA-based system updates grids at much higher speed over the software simulations because it achieves much better computing performance at each time step. On the other hand, as the order of the FDTD scheme is increased from the 2nd to 6th, the computational throughput is improved about 9.5%.

Table 4: *Computational throughput (Ggrids/s)*

orders	FPGA	software simulation
2nd	8.8457	0.8015
4th	8.3604	0.6235
6th	9.6882	0.5207

4.4. Discussion

In the current evaluation, sound propagation in a simple three-dimensional shoebox was analyzed using the developed FPGA-based sound field rendering system with the high-order FDTD method. For a sound space with complex geometries, decomposition methods to discretize a sound space into a grid mesh are required. In the hardware system, the data flow to stream data from the external on-board memory will be changed, and the system data path may become complicated. Furthermore, all boundaries were clamped to 0 in current evaluations. If complex boundary conditions are adopted, the updated equations for the grids on the boundaries are needed to be derived from the high-order FDTD method, and the computing unit inside a PE will be changed in the FPGA-based sound field rendering systems because the updated equation is different in accordance with the position of a grid.

On the other hand, the computing pattern in sound field rendering with FDTD methods is stencil computation in principle, in which the bottleneck of computing performance is memory bandwidth. In the current design, the spatial blocking is applied to alleviate the required memory bandwidth, and the temporal blocking is adopted to reuse data and reduce memory access to external memory. Although FPGA provides on-chip block memories with large bandwidth, the size of on-chip block memories is limited, such as several Mega bytes. And the FPGA card DE10-Pro provides large size on-board external memory, which is 8GB DDR4-2400 DRAMs. In the FPGA-based acceleration system, computation is sped up through customization of data path according to the data flow during computing and parallelism of PEs. In contrast, current GPUs provide several Giga bytes high speed and high memory bandwidth (HBM) memories, and data access overhead will be reduced significantly. Moreover, development of an FPGA-based system needs much hardware knowledge even though high level synthesis is widely applied in recent years. Development of a GPU-based system is relatively easier than that of FPGA-based system. All these results in that GPUs are more popular in computing than FPGAs. At next step, a sound field rendering system will be developed using GPU and compared with the

proposed FPGA-based sound field rendering system to validate which platform is better for sound field rendering.

5. CONCLUSIONS

High-order FDTD method provides more accurate approximation and smaller dispersion. The sound field rendering with FDTD method is computationally intensive and memory intensive. In this research, an FPGA-based sound field rendering system based on the high-order FDTD method is developed to speed up computation. The spatial blocking is applied to reduce the size of the required on-chip buffer and memory bandwidth, and the temporal blocking is adopted to reuse data and compute sound pressures of grids in the same spatial block at 16 continuous time steps in parallel. In the sound field rendering system with the 2nd-order, 4th-order, and 6th-order FDTD schemes, the FPGA-based system achieves much higher performance in computing and computational throughput over the software simulations carried out in a desktop machine even though the FPGA-based rendering systems run at much lower clock frequency and has smaller on-chip and external on-board memories. The evaluation results demonstrate that FPGAs are promising for sound field rendering. In future work, the decomposition methods to discretize a sound space with complex geometries into a grid mesh and the high-order FDTD schemes with complicated boundary conditions will be studied, and a real-time sound field rendering system based on the proposed architecture and high-order FDTD methods with complicated boundary conditions will be investigated, in which input incidence, computation, and computed results are all handled at real time. As a comparison, a counterpart system based on GPUs will be developed to compared with the FPGA-based sound field rendering system and explore the suitable platform for sound field rendering.

6. ACKNOWLEDGMENTS

Thanks for Intel's donation of the FPGA card DE10-Pro and EDA tools through its University Program. This work was supported by the JSPS KAKENHI Grant Number JP22K12123.

7. REFERENCES

[1] S. Kitic, N. Bertin, and R. Gribonval, "Hearing behind walls: localizing sources in the room next door with cosparsity," in Proc. *IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 3087 - 3091.

[2] N. Antonello, T. Waterschoot, M. Moonen, and P. Naylor, "Source localization and signal reconstruction in a reverberant field using the FDTD method," in *Proc. Eur. Signal Process. Conf.*, 2014, pp. 301 - 305.

[3] N. Bertin, S. Kiti, and R. Gribonval, "Joint estimation of sound source location and boundary impedance with physics-driven cosparse regularization", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 6340 - 6344.

[4] R. Mehra and D. Manocha, "Wave-based sound propagation for VR applications", in *Proc. IEEE Virtual Real.*, Minnesota, USA, 2014.

[5] N. Raghuvanshi, R. Narain, and M. C. Lin, "Efficient and Accurate Sound Propagation Using Adaptive Rectangular Decomposition," *IEEE Trans. Visualization and Computer Graphics*, vo. 15, no. 5, pp. 789 - 801, 2009.

[6] V. Valimaki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty Years of Artificial Reverberation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 5, pp. 1421 - 1448, 2012.

[7] N. Antonello, T. Waterschoot, M. Moonen, and P. Naylor, "Identification of surface acoustic impedances in a reverberant room using the FDTD method," in Proc. *IEEE Int. Workshop Acoust. Signal Enhancement*, pp. 114 - 118, 2014.

[8] Botteldooren, "Acoustical Finite-Difference Time-Domain Simulation in a quasi-cartesian node," *J. Acoust. Soc. Am.*, vol. 95, pp. 2313 - 2319, 1994.

[9] D. Botteldooren, "Finite-Difference Time-Domain Simulation of Low-Frequency Room Acoustic Problems," *J. Acoust. Soc. Am.*, vol. 98, pp. 3302 - 3308, 1995.

[10] O. Chiba, T. Kashiwa, H. Shimoda, S. Kagami, and I. Fukai, "Analysis of Sound Fields in Three-Dimensional Space by the Time-Dependent Finite-Difference Method based on the Leap Frog Algorithm," *J. Acoust. Soc. Jpn.*, vol. 49, pp. 551 - 562, 1993.

[11] L. Savioja, T. Rinne, and T. Takala, "Simulation of room acoustics with a 3-D finite difference mesh," in *Proc. Int. Computer Music Conf. (ICMC)*, Aarhus, Denmark, Sept. 1994, pp. 463-466.

[12] J. O. Smith III, "Physical Modeling Using Digital Waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74 - 91, 1992.

[13] L. Savioja, and V. Valimaki. "Interpolated Rectangular 3-D Digital Waveguide Mesh Algorithms with Frequency Warping," *IEEE Trans. Speech Audio Process.*, vol. 11, pp. 783 - 790, 2003.

[14] G. R. Campos and D. M. Howard, "On the Computational Efficiency of Different Waveguide Mesh Topologies for Room Acoustic Simulation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 13, no. 5, pp. 1063 - 1072, 2005.

[15] D. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley, "Acoustic Modeling Using the Digital Waveguide Mesh," *IEEE Signal Process Magazine*, vol. 24, no. 2, pp. 55 - 66, 2007.

[16] K. Kowalczyk and M. Walstijn, "Room Acoustics Simulation Using 3-D Compact Explicit FDTD Schemes," *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, no. 1, pp. 34 - 46, 2011.

[17] J. Mourik and D. Murphy, "Explicit Higher-Order FDTD Schemes for 3D Room Acoustic Simulation," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 22, no. 12, pp. 2003 - 2011, 2014.

[18] B. Hamilton and S. Bilbao, "Fourth-order and optimised finite difference schemes for the 2-D wave equation," in *Proc. Digital Audio Effects (DAFx'13)*, Maynooth, Ireland, Sept. 2013, pp. 2 - 6.

[19] B. Hamilton, S. Bilbao, and C. J. Webb, "Revisiting implicit finite difference schemes for 3D room acoustics simulations on GPU," in *Proc. Digital Audio Effects (DAFx'14)*, Erlangen, Germany, Sept. 2014, pp. 41 - 48.

[20] B. Hamilton and S. Bilbao, "FDTD Methods for 3-D Room Acoustics Simulation with High-Order Accuracy in Space and Time," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, pp. 2112 - 2124, 2017.

[21] T. Ishii, T. Tsuchiya, K. Okubo, "Three-Dimensional Sound Field Analysis Using Compact Explicit Finite Difference Time Domain Method with Graphics Processing Unit Cluster System," *Jpn. J. Appl. Phys.* vol. 52, pp. 07HC11, 2013.

[22] T. Tsuchiya and N. Maruta, "Three-Dimensional Compact Explicit Finite Difference Time Domain Scheme with Density Variation," *Jpn J. Appl. Phys.* vol. 57, pp. 07LC01, 2018.

[23] C. Spa, A. Rey, and E. Hernandez, "A GPU Implementation of an Explicit Compact FDTD Algorithm with a Digital Impedance Filter for Room Acoustics Applications," *IEEE/ACM Trans. Audio, Speech, Lang. Process*., vol. 23, no. 8, pp. 1368 - 1380, 2015.

[24] M. Tanaka, T. Tsuchiya, and K. Okubo, "Two-Dimensional Numerical Analysis of Nonlinear Sound Wave Propagation Using Constrained Interpolation Profile Method Including Nonlinear Effect in Advection Equation," *Jpn. J. Appl. Phys.*, vol. 50, pp. 07HE17, 2011.

[25] L. Savioja, "Real-time 3D finite-difference time-domain simulation of low-and mid-frequency room acoustics," in *Proc. Digital Audio Effects (DAFx'10)*, Graz, Austria, Sept. 2010.

[26] A. Southern, D. Murphy, G. Campos, and P. Dias, "Finite difference room acoustic modelling on a general purpose graphics processing unit," in *Proc. 128th Audio Engineering Society Convention*, London, UK, May 2010, pp. 1393 - 1403.

[27] L. Savioja, D. Manocha, and M. Lin, "Use of GPUs in room acoustic modeling and auralization," in *Proc. Int. Symposium on Room Acoustics (ISRA)*, Melbourne, Australia, August 2010.

[28] C. Webb and S. Bilbao, "Virtual room acoustics: a comparison of techniques for computing 3D-FDTD schemes using CUDA," in *Proc. 130th Audio Engineering Society Convention*, London, UK, May 2011, pp. 1163–1169.

[29] C. Webb and S. Bilbao, "Computing room acoustics with CUDA-3D FDTD schemes with boundary losses and viscosity," in *Proc. Int. Conf. on Acoustics Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 317–320.

[30] Y. Y. Tan, Y. Inoguchi, E. Sugawara, M. Otani, Y. Iwaya, Y. Sato, H. Matsuoka, and T. Tsuchiya, "A Real-Time Sound Field Renderer based on Digital Huygens' Model," *J. Sound Vib.*, vol. 330, pp. 4302 - 4312, 2011.

[31] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "A Hardware-Oriented Finite-Difference Time-Domain Algorithm for Sound Field Rendering," *Jpn. J. Appl. Phys.*, vol. 52, pp. 07HC03, 2013.

[32] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "A Real-Time Sound Rendering System based on the Finite-Difference Time-Domain Algorithm," *Jpn. J. Appl. Phys.*, vol. 53, pp. 07KC14, 2014.

[33] Y.Y. Tan, Y. Inoguchi, M. Otani, Y. Iwaya, and T. Tsuchiya, "A Real-Time Sound Field Rendering Processor," *Appl. Sci.*, vol. 8, no. 35, 2018.

[34] Y. Y. Tan and T. Imamura, "A FPGA-based accelerator for sound field rendering", Proc. Digital Audio Effect, 2019.

[35] J. Saarelma, J. Califa, and R. Mehra, "Challenges of distributed real-time finite-difference time-domain room acoustic simulation for auralization," in *Proc. AES Int. Conf. Spatial Reproduction -Aesthetics and Science*, Tokyo, Japan, July 2018, PP-1.

[36] Y. Y. Tan and T. Imamura, "An FPGA-based sound field rendering system", in *Proc. IEEE Cluster*, online, 2020.

[37] M. Ciment and S. Leventhal, "Higher Order Compact Implicit Schemes for the Wave Equation," *Math. Comput.*, vol. 29, no. 132, pp. 985-994, 1975.

[38] P. Deuflhard and A. Hohmann, *Numerical Analysis in Modern Scientific Computing*, Springer-Verlag, New York, 2nd edition, 2003.

[39] S. Sakamoto, "Phase-Error Analysis of High-Order Finite Difference Time Domain Scheme and its Influence on Calculation Results of Impulse Response in Closed Sound Field," *Acoust. Sci. Technol.*, vol. 28, no. 5, pp. 295-309, 2007.

[40] Y. Y. Tan and Toshiyuki Imamura, "Design and implementation of high-order FDTD method for room acoustics," in *Proc. 41th USE*, Tokyo, Japan, 2020.

[41] J. Mourik, *Higher-order Finite Difference Time Domain Algorithms for Room Acoustic Modelling*, PhD Thesis, University of York, 2016.

[42] H. Zohouri, A. Podobas, and S. Matsuoka, "Combined spatial and temporal blocking for high-performance stencil computation on FPGAs using OpenCL", in *Proc. FPGA*, 2018.

[43] Yiyu Tan, Toshiyuki Imamura, Masaaki Kondo, "FPGA-based Acceleration of FDTD Sound Field Rendering", *J. Audio Eng. Soc.*, vol. 69, no. 7/8, pp. 542–556, 2021.

[44] https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=13&No=1144&PartNo=1

HOW SMOOTH DO YOU THINK I AM: AN ANALYSIS ON THE FREQUENCY-DEPENDENT TEMPORAL ROUGHNESS OF VELVET-NOISE

Jade Roberts[1,2]*, Jon Fagerström*[1]*, Sebastian J. Schlecht*[1,2] *and Vesa Välimäki*[1]

[1]Acoustics Lab, Department of Information and Communications Engineering
[2]Media Lab, Department of Art and Media
Aalto University
Espoo, Finland
jade.roberts@aalto.fi

ABSTRACT

Velvet noise is a sparse pseudo-random signal, with applications in late reverberation modeling, decorrelation, speech generation, and extending signals. The temporal roughness of broadband velvet noise has been studied earlier. However, the frequency-dependency of the temporal roughness has little previous research. This paper explores which combinative qualities such as pulse density, filter type, and filter shape contribute to frequency-dependent temporal roughness. An adaptive perceptual test was conducted to find minimal densities of smooth noise at octave bands as well as corresponding lowpass bands. The results showed that the cutoff frequency of a lowpass filter as well as the center frequency of an octave filter is correlated with the perceived minimal density of smooth noise. When the lowpass filter with the lowest cutoff frequency, 125 Hz, was applied, the filtered velvet noise sounded smooth at an average of 725 pulses/s and an average of 401 pulses/s for octave filtered noise at a center frequency of 125 Hz. For the broadband velvet noise, the minimal density of smoothness was found to be at an average of 1554 pulses/s. The results of this paper are applicable in designing velvet-noise-based artificial reverberation with minimal pulse density.

1. INTRODUCTION

Velvet noise is a sparse pseudo-random noise sequence, which consists of ternary values (-1, 0, and 1) [1] and has a constant power spectrum [2]. Velvet noise was originally proposed by Karjalainen and Järveläinen to model room reverberation [1]. It is known that late reverberation resembles exponentially-decaying, filtered white noise [3, 4, 5]. Broadband velvet noise has been shown to retain its perceived smoothness with lower pulse densities in comparison to other types of sparse noise sequences [6]. At 2000 impulses per second, velvet noise has been shown to sound smoother than Gaussian white noise (GWN) [6].

The perceived temporal smoothness of velvet noise has been investigated mostly on broadband noise sequences [6]. Karjalainen and Järveläinen [1] made an initial study on the frequency-dependency of the temporal roughness, where lowpass filtered velvet-noise, with a cutoff frequency of $1.5\,\mathrm{kHz}$, was shown to sound smoother than GWN with a pulse density of 600 pulses/s. This paper investigates further the frequency-dependent psychoacoustic temporal roughness of velvet-noise sequences. Having a clear understanding of the frequency-dependency of the temporal roughness can help in the design and optimization of reverberation models based on sparse noise sequences.

The earliest sparse-noise-based reverberation algorithm was proposed by Rubak and Johansen [7, 8]. Their algorithm is based on totally random noise (TRN), which is a type of sparse pseudo-random noise with an equal probability of any sample having a non-zero value. The proposed minimal pulse density for producing high-quality noise with the TRN was reported to be between $2000 - 4200$ pulses/s, however for a lowpass filtered noise with cutoff at 8 kHz.

Rubak and Johansen also proposed a recursive structure for computational efficiency [7], which was further improved by Karjalainen and Järveläinen [1] by replacing the TRN with velvet noise and by introducing time-variation. The time variation was introduced to reduce the periodicity of repeating the same short velvet-noise sequence inside the recursive structure. A further problem arises from the time-variability which creates warbling especially on stationary input sounds [1, 9, 10]. An alternative solution to mitigate the periodicity problem was proposed in [10], where interleaved velvet-noise sequences hide the repetitiveness.

A different approach to reverberation modeling was taken in [4, 5], where filtered velvet noise segments were concatenated to model target late-reverberation. It was shown empirically that lower pulse density could be used towards the end of the model response, where the bandwidth was reduced. Recently, it was also shown that colored velvet noise can be generated directly by controlling the pulse location distribution [11, 2]. A practical algorithm for generating velvet noise with a lowpass spectrum, called dark velvet noise (DVN), was later proposed in [12]. The cutoff of DVN can be varied in time to generate characteristic late-reverberation, where the low frequencies decay slower than the high frequencies.

Another application for velvet-noise is to implement an efficient decorrelator [13, 14]. An optimization scheme for minimizing the spectral coloration introduced by the velvet-noise decorrelator was proposed in [14]. Velvet noise has been also used in hybrid reverb structures combining it with feedback delay networks (FDN) [15, 16]. Short velvet-noise filters are applied either within the feedback matrix [16] or at the inputs and outputs of the FDN [15]. Additionally, velvet-noise has been used in vocoder-based speech generation by serving as excitation signals [17].

In this paper, the perceptual temporal roughness of velvet noise at octave bands as well as at lowpass bandwidths with the octave band center frequencies as the cutoffs are studied in a perceptual test. Additionally, the time-domain smearing of various filter orders is investigated objectively to narrow down the filter

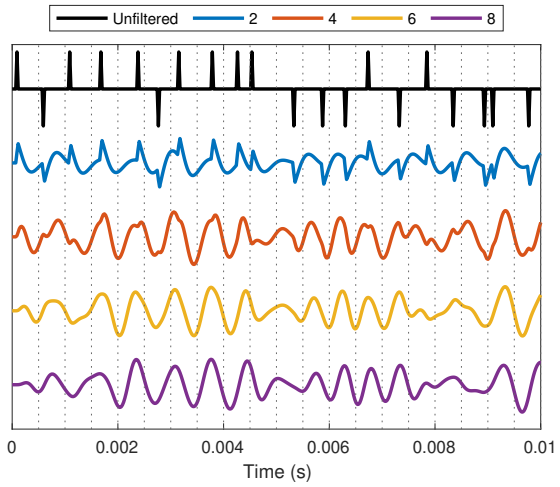


Figure 1: *Velvet-noise sequence, filtered with an octave-band filter at the center frequency $F_c = 1.5$ kHz, using different filter orders $N \in \{2, 4, 6, 8\}$ from top to bottom. The consecutive plots are offset for better visualization. The grid size $T_d = 0.5$ ms is shown with a dotted line.*

selection for the listening test. An adaptive perceptual test was implemented in Matlab, where the subjects control the density of test velvet noise signals. The test aims to find perceptual density thresholds where each test signal still sounds as smooth as a reference velvet noise signal with the pulse density of 2000 pulses/s.

The rest of this paper is organized as follows. Section 2 gives relevant background on velvet noise and temporal roughness. Section 3 describes the design of the filters for the listening test. Section 4 introduces the listening test setup. The results of the listening test are analyzed in Section 5. Section 6 concludes the paper and comments the future work.

2. BACKGROUND

This section explains the concepts behind velvet noise and temporal roughness.

2.1. Velvet Noise

Velvet noise is a sparse random noise sequence comprised of only sample values of -1, 0, and 1. Each frame contains a single randomly placed impulse with randomized sign; the rest are zeros, leading to a sparser noise than GWN. The number of nonzero impulses per second is defined as the pulse density ρ, i.e.,

$$\rho = \frac{f_s}{T_d}, \tag{1}$$

where T_d refers to the average distance between impulses measured in samples and f_s refers to the sampling rate. In this work, a sampling rate of $f_s = 44100$ Hz is used for all generated signals.

Karjalainen and Järveläinen [1] found that a pulse density of ρ = 1500 pulses/s satisfied the aim of minimal pulse density and maximal smoothness. This was the sweet spot for a perceived smoother noise than GWN. To prevent gaps and clusters of sample values which contribute to the perception of temporal roughness, the impulse locations are determined as [6]

$$k(m) = \lfloor mT_d + r_1(m)(T_d - 1) \rceil, \tag{2}$$

where m refers to the pulse counter while $r_1(m)$ refers to the sequence of uniformly distributed random values between 0 and 1, and $\lfloor \cdot \rceil$ is the rounding operation. The velvet-noise sequence is computed with

$$s(n) = \begin{cases} 2\lfloor r_2(m) \rceil - 1, & \text{when } n = k(m) \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where n is the sample index and $r_2(m)$ is another uniform random number sequence to decide when an impulse will be 1 or -1. In Fig. 1, the black line at the top shows a broadband velvet noise sequence. Here, the impulses of 1 or -1 appear only once in each frame.

Additionally, velvet noise is featureless with a flat power spectrum [6]. The computing time of a convolution with velvet noise is much shorter than that of GWN because it mainly consists of zeros and where there are impulses in velvet noise, the ones and minus ones are easy to multiply by [4, 5].

2.2. Temporal Roughness of Sparse Noise

Temporal roughness is a psychoacoustic quality of sparse noise sequences where the lower the pulse density the rougher the signal is perceived [1, 6]. Temporal roughness has not been fully defined in the literature but it might be directly related to the roughness which is defined as the sensation caused by amplitude-modulated sine waves with modulation frequencies between $15 - 300$ Hz range. The sensation reaches its maximum around modulation frequency of 70 Hz [18].

Rubak and Johansen [7] observed that when the delay of a comb filter is increased above 25 ms one starts to perceive roughness in the sound and the perception changes from the coloration to the time-domain character. The value 25 ms corresponds to a frequency of 40 Hz for the pulses of the comb filter, which falls close to the modulation frequency causing maximal roughness sensation. Furthermore, it is reported that the modulation signal does not have to be periodic to cause the perception of roughness [18] The random assignment of pulses in sparse noise sequence can be interpreted as pseudo-random amplitude modulation [19].

Velvet noise has the ability to sound smoother than GWN, leading to the question of at which pulse densities is velvet-noise perceived as smooth versus perceived as temporally rough. In previous studies on sparse noise, optimal pulse densities for still maintaining perceived smoothness were researched in multiple ways such as first-order lowpass filtering reverberation tails and using totally random noise [7, 8] as well as using velvet-noise [1] which gave results of $2000 - 4200$ pulses/s [7, 8] and 1500 pulses/s [1] as optimal smoothness, respectively.

3. FILTER DESIGN

In order to investigate the frequency-dependent temporal roughness of velvet noise, filtering is to be applied to the white velvet-noise sequences. In this research, both lowpass and octave-band filtering is applied. In this section, the properties of various filter orders are investigated, to narrow down the filter parameters for the perceptual test.

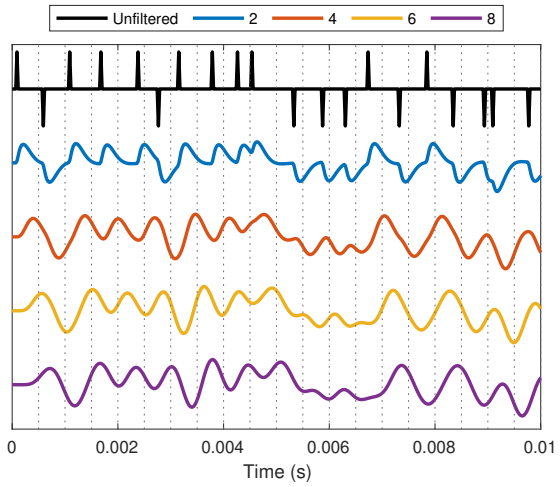


Figure 2: *Velvet-noise sequence, filtered with a Butterworth lowpass filter with the cutoff frequency $F_c = 1.5$ kHz, using different filter orders $N \in \{2, 4, 6, 8\}$, cf. Fig. 1.*

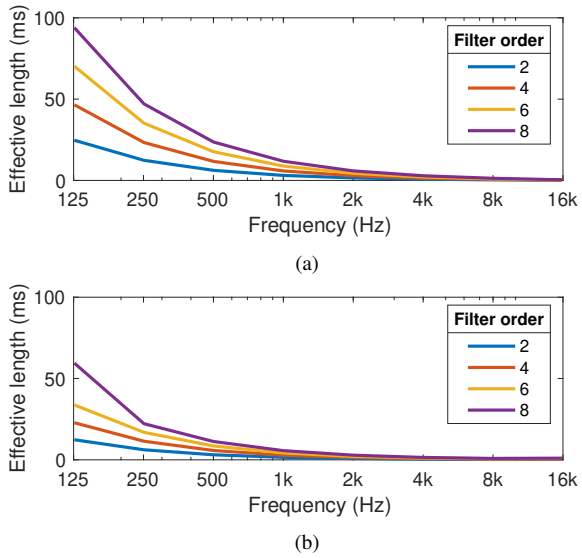


Figure 3: *Effective length of the (a) octave-band filters and (b) lowpass filters at the center frequencies $F_c \in \{125, 250, 500, 1k, 2k, 4k, 8k, 16k\}$ Hz, using different filter orders $N \in \{2, 4, 6, 8\}$.*

3.1. Time-domain Smearing

Time-domain smearing refers to the energy of a signal being spread across a longer period of time during playback which also means a loss of detail in the signal itself. Fig. 1 shows an example velvet-noise sequence (black) and its filtered versions (colored) with various filter orders of an octave band filter centered at $F_c = 1.5$ kHz. As the order of the filter increases the time response gets more and more smeared in time. With the second-order octave filter (blue) in Fig. 1, the pulse locations are still visible. A similar trend is shown in Fig. 2, which shows the velvet-noise sequence filtered with a lowpass Butterworth filter with various filter orders. The second-order lowpass filter shows more smearing than the second-order octave filter. This is why the fourth-order octave filters were used to compare against the lowpass filters.

Fig. 3a and Fig. 3b show the effective length of the octave-band filter and the Butterworth lowpass filter, respectively. Again, various filter orders are compared. The effective length is computed with Matlab function `impzlength` with a tolerance of -60 dB. Longer effective lengths of the filter will result in more time-domain smearing. The overall trend in Fig. 3 is that higher filter order and lower center frequency or cutoff frequency result in longer effective length. Furthermore, the difference in effective length between the lower and higher center frequencies grows with the filter order. Thus, for the listening test design we opted to use the second-order lowpass filters which introduce minimal smearing. For the octave filters order four was used, since the steepness of the second-order lowpass is most similar to that of a fourth-order octave filter.

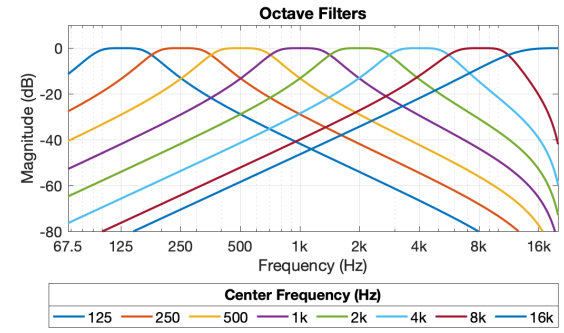
3.2. Listening Test Filters

Two types of filters were used in the final listening test: second-order Butterworth lowpass filters and fourth-order octave-band filters. A lowpass filter attenuates frequencies above a specified cutoff frequency and the frequencies below the cutoff are retained. The Butterworth lowpass filter [20] is often used in audio because of its maximally flat magnitude response in the passband and monotonic roll-off in the stopband. The magnitude responses of the used octave filters and lowpass filters are shown in Fig. 4a and Fig. 4b, respectively. Note: There is a slight shift of the lowpass passband versus the passband of the octave filters.

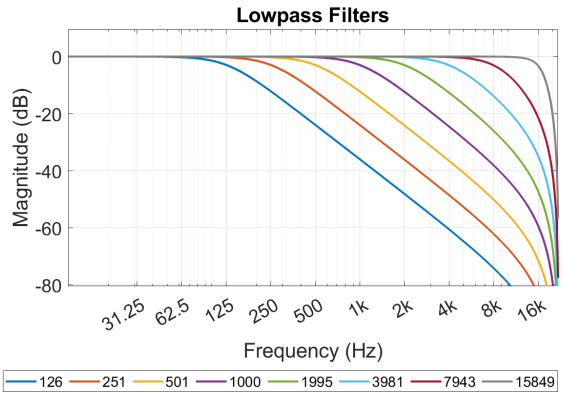
As for octave filters, an octave means an interval where there is a frequency ratio of 2:1, the upper frequency is twice the lower frequency. This is important when using the filter which consists of bandpass filters, the bandwidth of each filter will always be a 2:1 ratio. There are ten octave bands within the human hearing range. Octave filters are used because they can be utilized for measuring noise power at certain frequency ranges. They also give insight into the human hearing of temporal roughness which was the aim of this study. The magnitude response is shown in Fig. 4a. The center and cutoff frequencies were calculated using base 10. The nominal center frequencies of the octave filters used in the test are shown in Table 1 and is also shown by the peaks/centers of the octaves filter magnitude responses in Fig. 4a.

4. LISTENING TEST DESIGN

To test the perception of smoothness in velvet noise, a listening test was employed using Matlab App. As shown in Fig. 5 of the test user interface, there is a reference signal and a test signal. The reference signal was broadband velvet noise of 2000 pulses/s which has been shown to be perceived as smooth [1]. The participant was asked to find the lowest possible pulse density in which the test signal sounded as smooth as the reference. There was not a specific practice page, but testers were given a brief introduction to the sliders and signals where they could ask questions about the task. In this opportunity, they were able to familiarize themselves with the loudness level of the signals, the buttons and the slider



(a)



(b)

Figure 4: *(a) Fourth-order octave and (b) second-order Butterworth lowpass filter magnitude responses with center/cutoff frequencies one octave apart between 125 Hz and 16,000 Hz.*

function. The test required the participant to move the slider accordingly to match their perception of where the velvet noise signal is still smooth, but on the edge of being rough. The movement of the slider changed the pulse density of the velvet noise signal

Table 1: *Nominal center frequencies used to test the effect of octave and lowpass filters. For the lowpass filters, the center frequency was used as the cutoff frequency. Note: The lowest 2 octaves of the audio range are omitted, only octaves 3 to 10 were used in testing.*

Octave band	Nominal center frequency (Hz)
3	125
4	250
5	500
6	1000
7	2000
8	4000
9	8000
10	16000

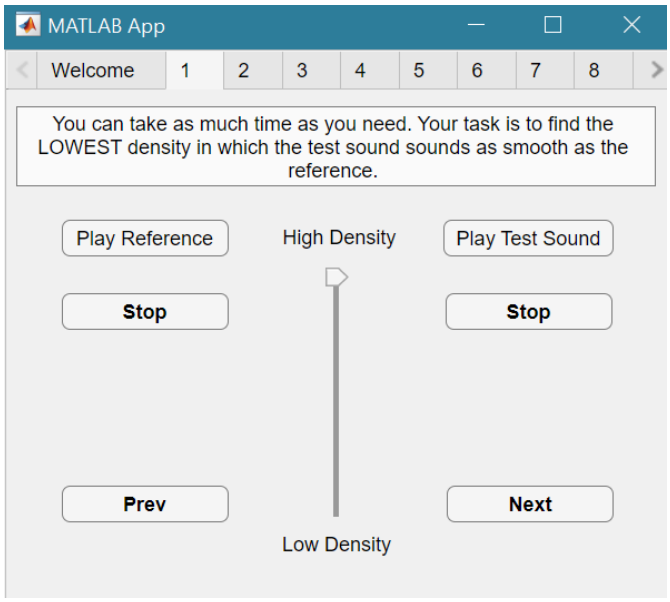


Figure 5: *User interface of the listening test. The participant chooses the lowest density where the test sound is still as smooth as the reference. The user can choose to play and stop the reference sound and the test sound. The participant can also go back to the previous or go to the next page using the respective buttons.*

and automatically played the test sound set to the chosen pulse density. Each page's velvet noise condition contained all possible pulse densities between 50 and 2000 and the reference remained at 2000 pulses per second for each test page. A pulse density of 2000 was chosen for the reference as previous papers on sparse and velvet noise respectively found already 1500 and 2000 as acceptably smooth pulse densities [1, 6].

The test signals presented were either unfiltered, filtered through an octave filter or through a lowpass Butterworth filter. Sound examples of the test signals are available on the companion web page of this paper[1] The filters used center frequencies or cutoff frequencies calculated from octave bands 3-10, shown in Table 1. The listening test was composed of two linked MATLAB apps, the first assessed octave filtering, while the second assessed lowpass filtered and broadband velvet noise. The center frequency change in the filters between pages was randomized. The lowest two center frequency bands were omitted due to being too low for users to perceive, and, for this reason, are not included in Table 1 either.

The loudness level for each signal was set based on the EBU R 128 standard at -23 LUFS (Loudness Unit Full Scale). LUFS is a loudness measurement based on the human perception of loudness. This was used over the counterpart of RMS (root-mean-square) power which is the average power of a signal without any weighting. Additionally, the use of LUFS was due to the frequency-dependent nature of loudness LUFS can account for. Normalizing the loudness between test signals was to ensure that rating was not influenced by how loud the signal sounded such as the perception that high frequencies are louder compared to low frequencies.

[1] `http://research.spa.aalto.fi/publications/papers/dafx23-vn-roughness`

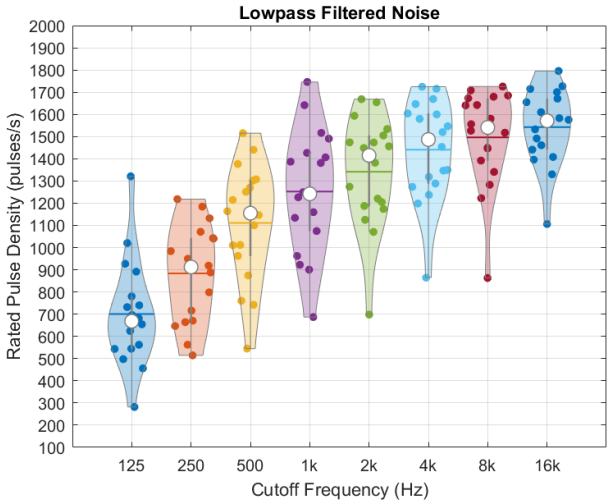


Figure 6: *Violin plot of minimal pulse density for smooth-sounding lowpass filtered velvet noise.*

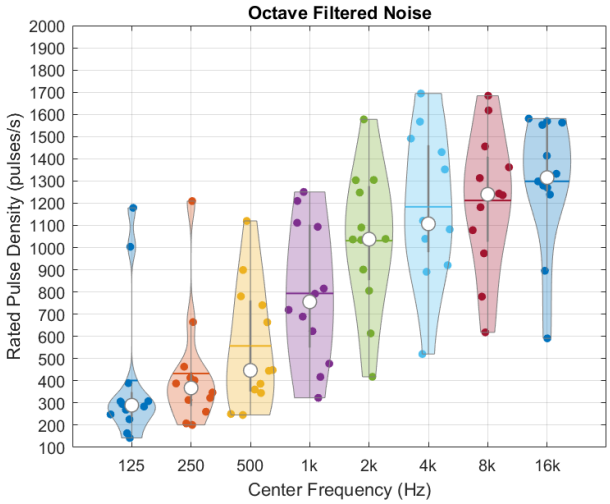


Figure 7: *Minimal pulse density for smooth-sounding octave filtered velvet noise.*

The final listening test sound pressure level of the isolated listening booths used for testing was calibrated to 60 dB [6] using a RA0045 G.R.A.S Ear Simulator. The simulator followed IEC 60318-4 regulations. Participants listened to the signals using Sennheiser HD-650 headphones.

There was a total of 34 listening test pages with 16 pages testing octave-filtered velvet noise, 16 testing pages for lowpass-filtered velvet noise, and two pages testing unfiltered velvet noise. This was done so that each signal of same conditions: filter type and center frequency or cutoff frequency, occurred twice during the test to compare individual differences and assess the reliability of the participants' ratings.

In total, there were 12 participants between ages 19 and 42 who completed the octave filtered velvet noise tests and 18 participants between ages 19 and 42 who completed the lowpass filtered velvet noise test all with previous experience in a formal listening test. For each test signal, there were two pages, and a correlation was calculated for each participant between the two pages for each center frequency/cutoff frequency. This was done using `corrcoef` in MATLAB. If the participant's mean correlation coefficient was below 0.5, meaning their answers were too different for the same stimuli to be considered, their data was discarded. Fortunately, no participants had a correlation coefficient under 0.8 and therefore, all participant data was assessed.

5. RESULTS

The results in Fig. 6 and Fig. 7 show the rated pulse density in the y-axis against the center frequencies of the 8 octave bands used in this study on the x-axis. The rated pulse density values come from the participant's task of setting the slider to where they perceive the lowest possible density where it still sounds as smooth as the reference signal. In both violin plots [21], the central white dot refers to the median of the data within the specific octave band center frequency. The means are indicated by a horizontal line in each violin. The median and mean values can be found in Tables 2 and 3. These values were calculated using an average of each participant's answers for each filter band and filter type respectively.The top and bottom of the thick grey line in the center of each plot refers to the first and third quartiles. Both plots seem to follow somewhat of a curve showing that there is some frequency-dependence on the perception of smoothness. Especially the octave-filtered velvet noise was rated with a consistently

Table 2: *Nominal cutoff frequencies used for lowpass cutoffs and their corresponding median and mean results.*

Cutoff frequency (Hz)	Median (pulses/s)	Mean (pulses/s)
125	668	725
250	912	896
500	1155	1120
1000	1242	1260
2000	1415	1350
4000	1487	1455
8000	1541	1509
16000	1571	1548
Broadband	1587	1554

Table 3: *Nominal center frequencies used for octave filters and their corresponding median and mean results.*

Center frequency (Hz)	Median (pulses/s)	Mean (pulses/s)
125	290	401
250	368	433
500	447	557
1000	756	794
2000	1038	1035
4000	1107	1183
8000	1240	1212
16000	1315	1299
Broadband	1587	1554

lower pulse density than for the low pass filtered velvet noise.

Fig. 6 demonstrates that lower cutoff frequencies allowed the pulse density of the velvet noise to be lower than that of broadband and octave-filtered signals. The signals were rated at nearly half the pulse density of broadband in the lowest octave band center frequency cutoff of 125 Hz, whereas in the highest human hearing octave band frequency cutoff, there was little difference.

We found the broadband velvet noise had a median of 1587 pulses/s and a mean of 1554 pulses/s where the signal still sounded as smooth as the reference signal, as shown in Fig. 6, which is similar to results in the original velvet noise study [1], where 1500 pulses/s was found to be an optimal smoothness. The filter shapes of the second-order Butterworth lowpass filters and the fourth-order octave filters were similar, however, the octave-filtered velvet noise had much lower ratings.

A repeated measures two-way analysis of variance (ANOVA) test with factors of center or cutoff frequency and repetition was used to assess the statistical significance of the octave-filtered velvet noise and the lowpass-filtered velvet noise separately. the second factor of repetition was tested to determine if there were significant differences in how a participant rated the same stimuli. For both filtered velvet noises, the dependent variable was pulse density. ANOVA preconditions such that the data is normally distributed, the dependent variable of rated pulse density is on a continuous scale and the sphericity of the two trials per participant showing equal amounts of variance were met. More specifically, a Kolmogorov–Smirnov test was conducted on the rated pulse densities for each center and cutoff frequency at a 1% significance level which found the data to be normally distributed for both the lowpass filtered data and the octave filtered data separately. A Mauchly sphericity test was also conducted which found that for each participant, the density ratings across the center or cutoff frequencies were normally distributed.

For the octave-filtered noise, the repeated measures two-way analysis was chosen to show the effect of center-frequencies and repeating trials on participant pulse density rating. The two-way ANOVA test showed that center-frequencies was significant and repeated trials were not significant on pulse density with F-statistics of $F(9, 99) = 79.801, p < .001$ and $F(1, 11) = .995, p > .001$, respectively. The interaction between frequency and repetition was also not significant with an F0-statistic of $F(9, 99) = 1.609, p > .001$.

For the lowpass-filtered noise, the two-way analysis was chosen to show the effect of cutoff frequency and repetition of conditions on pulse density ratings. There was no significant effect from repetition, $F(1, 16) = .278, p > .001$. or from the interaction between repetition and frequency, $F(9, 144) = 1.475, p > .001$. The two-way repeated measures ANOVA testing found only significant effect from cutoff frequencies on pulse density rating such that F(9, 144) = 354.15, $p < .001$.

Additionally, a post-hoc paired t-test was conducted with a Bonferroni-Holm correction on both the lowpass and octave filtered noise tests which showed that between neighboring center or cutoff frequencies, the pulse density ratings were statistically significant ($p < .01$) except between 4 kHz and 8 kHz as well as between 8 kHz and 16 kHz. This means that pulse densities above 4 kHz show no important differences in pulse density ratings.

6. CONCLUSIONS

We studied the frequency-dependency of temporal roughness of velvet noise in human noise perception. The results showed that second-order lowpass filtering and fourth-order octave filtering, the shape of the filter, allows for temporal smearing meaning that the resulting signal can still seem smooth at lower densities than that of broadband velvet noise. Octave-filtered noise at the lowest frequencies showed even lower pulse density ratings than lowpass filtered velvet noise. This study showed that artificial reverberation modeling which utilizes filtering and velvet noise can employ lower pulse densities than what is currently used which allows for more efficient computation.

7. ACKNOWLEDGMENTS

This work has been funded in part by the Nordic Sound and Music Computing Network, NordForsk project number 86892. The authors would like to thank Dr. Hanna Järveläinen for her helpful comments and would also like to thank Nils Meyer-Kahlen in his help with ANOVA testing and clarifications.

8. REFERENCES

[1] M. Karjalainen and H. Järveläinen, "Reverberation modeling using velvet noise," in *Proc. Audio Eng. Soc. 30th Int. Conf. Intell. Audio Environ.*, Saariselkä, Finland, Mar. 2007.

[2] N. Meyer-Kahlen, S. J. Schlecht, and V. Välimäki, "Colours of velvet noise," *Electronics Letters*, vol. 58, no. 12, pp. 495–497, Jun. 2022.

[3] J. A. Moorer, "About this reverberation business," *Computer Music J.*, vol. 3, no. 2, pp. 13–28, Jun. 1979.

[4] B. Holm-Rasmussen, H.-M. Lehtonen, and V. Välimäki, "A new reverberator based on variable sparsity convolution," in *Proc. Int. Conf. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sep. 2013, pp. 344–350.

[5] V. Välimäki, B. Holm-Rasmussen, B. Alary, and H-M. Lehtonen, "Late reverberation synthesis using filtered velvet noise," *Applied Sciences*, vol. 7, no. 5, 2017.

[6] V. Välimäki, H.-M. Lehtonen, and M. Takanen, "A perceptual study on velvet noise and its variants at different pulse densities," *IEEE Trans. Audio Speech Lang. Process.*, vol. 21, no. 7, pp. 1481–1488, Jul. 2013.

[7] P. Rubak and L. G. Johansen, "Artificial reverberation based on a pseudo-random impulse response: Part I," in *Proc. 104th Conv. Audio Eng. Soc.*, Amsterdam, The Netherlands, May 1998.

[8] P. Rubak and L. G. Johansen, "Artificial reverberation based on a pseudo-random impulse response: Part II," in *Proc. 106th Conv. Audio Eng. Soc.*, Munich, Germany, May 1999.

[9] K.-S. Lee, J. S. Abel, V. Välimäki, T. Stilson, and D. P. Berners, "The switched convolution reverberator," *J. Audio Eng. Soc.*, vol. 60, no. 4, pp. 227–236, Apr. 2012.

[10] V. Välimäki and K. Prawda, "Late-reverberation synthesis using interleaved velvet-noise sequences," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 29, pp. 1149–1160, Feb. 2021.

[11] K. J. Werner, "Generalizations of velvet noise and their use in 1-bit music," in *Proc. Int. Conf. Digital Audio Effects (DAFx-19)*, Birmingham, UK, Sep. 2019.

[12] J. Fagerström, N. Meyer-Kahlen, S. J. Schlecht, and V. Välimäki, "Dark velvet noise," in *Proc. Int. Conf. Digital Audio Effects (DAFx-22)*, Vienna, Austria, Sep. 2022, pp. 192–199.

[13] B. Alary, A. Politis, and V. Välimäki, "Velvet-noise decorrelator," in *Proc. Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, Sep. 2017, pp. 405–411.

[14] S. J. Schlecht, B. Alary, V. Välimäki, and E. A. P. Habets, "Optimized velvet-noise decorrelator," in *Proc. Int. Conf. Digital Audio Effects (DAFx-18)*, Aveiro, Portugal, Sep. 2018, pp. 87–94.

[15] J. Fagerström, B. Alary, S. J. Schlecht, and V. Välimäki, "Velvet-noise feedback delay network," in *Proc. Int. Conf. Digital Audio Effects (DAFx-20)*, Vienna, Austria (remote), Sep. 2020, pp. 219–226.

[16] S. J. Schlecht and E. A. P. Habets, "Scattering in feedback delay networks," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 28, pp. 1915–1924, Jun. 2020.

[17] H. Kawahara, K.-I. Sakakibara, M. Morise, H. Banno, T. Toda, and T. Irino, "Frequency domain variants of velvet noise and their application to speech processing and synthesis," in *Proc. InterSpeech*, Hyderabad, India, Sep. 2018.

[18] H. Fastl and E. Zwicker, *Psychoacoustics – Facts and Models*, Springer, 2007.

[19] N. Meyer-Kahlen, S. J. Schlecht, and T. Lokki, "Perceptual roughness of spatially assigned sparse noise for rendering reverberation," *The Journal of the Acoustical Society of America*, vol. 150, no. 5, pp. 3521–3531, Nov. 2021.

[20] T. W. Parks and C. S. Burrus, *Digital Filter Design*, Wiley-Interscience, New York, NY, 1987.

[21] B. Bechtold, "Violin Plots for Matlab, Github Project," Available at https://github.com/bastibe/Violinplot-Matlab, accessed Jun. 3, 2021.

A FREQUENCY TRACKER BASED ON A KALMAN FILTER UPDATE OF A SINGLE PARAMETER ADAPTIVE NOTCH FILTER

Randall Ali and Toon van Waterschoot *

Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, Belgium
randall.ali@esat.kuleuven.be | toon.vanwaterschoot@esat.kuleuven.be

ABSTRACT

In designing a frequency tracker, the goal is to follow the continual time variation of the frequency from a particular sinusoidal component in a noisy signal with a high accuracy and a low sample delay. Although there exists a plethora of frequency trackers in the literature, in this paper, we focus on the particular class of frequency trackers that are built upon an adaptive notch filter (ANF), i.e. a constrained bi-quadratic infinite impulse response filter, where only a single parameter needs to be estimated. As opposed to using the conventional least-mean-square (LMS) algorithm, we present an alternative approach for the estimation of this parameter, which ultimately corresponds to the frequency to be tracked. Specifically, we reformulate the ANF in terms of a state-space model, where the state contains the unknown parameter and can be subsequently updated using a Kalman filter. We also demonstrate that such an approach is equivalent to doing a normalized LMS filter update, where the regularization parameter can be expressed as the ratio of the variance of the measurement noise to the variance of the prediction error. Through an evaluation with both simulated and realistic data, it is shown that in comparison to the LMS-updated frequency tracker, the proposed Kalman-updated alternative, results in a more accurate performance, with a faster convergence rate, while maintaining a low computational complexity and the ability to be updated on a sample-by-sample basis.

1. INTRODUCTION

Frequency estimation is a well-known problem in signal processing with a long history [1, 2] and continues to be relevant for a number of audio-related applications[1] including acoustic feedback detection [3, 4], automatic music transcription [5], tuning of musical instruments, and audio effects such as pitch-shifting just to name a few. Specifically, it refers to the problem of estimating the frequency of a sinusoidal component from a set of noisy observations. In cases where the periodic component of the noisy observations consist of harmonically-related sinusoidal components and it is the lowest frequency component that is being estimated, the problem is often referred to as fundamental frequency (f_0) estimation or pitch estimation [1, 2].

In this paper we are concerned with following the continual time variation of the frequency pertaining to a particular sinusoidal component of an audio signal, and hence we refer to the type of frequency estimation as frequency tracking. The main consideration for a frequency tracker is that it needs to have a low sample delay, with the ideal case being zero delay, and can be updated on a sample-by-sample basis. More concretely, the problem of frequency tracking can be summarized as finding an updated estimate of the frequency given a new set of samples or simply just one (in the real-time scenario) and a prior estimate of the frequency [6]. Several approaches for this have been proposed in the literature [6, 7, 8, 9], where the problem is referred to as pitch tracking.

The frequency tracker investigated in this work is one based on adapting the coefficients of a constrained bi-quadratic (biquad) infinite impulse response (IIR) filter [3, 4, 10, 11, 12], which functions as an adaptive notch filter (ANF). In a nutshell, the centre frequency of the ANF is continually updated so as to cancel a high-energy sinusoidal component in an attempt to minimize the mean-square of the output signal power. One main advantage of using the constrained biquad filter is that only one parameter needs to be adapted in order to obtain a frequency estimate. Furthermore, by expressing the filter in its direct form II, this single parameter can be updated very efficiently [12] such as with a least-mean-square (LMS) algorithm [3, 4], resulting in a frequency tracker that can be updated on a sample by sample basis. Moreover, the computational complexity is very low, making the algorithm suitable for real-time applications.

Our contribution in this paper is a subtle but powerful extension of the aforementioned approach, whereby we reformulate the ANF in terms of a state-space model, with the state containing the unknown parameter to be estimated. In such a formulation, a Kalman filter [13] can then be used for updating the state and hence for frequency estimation and tracking. We subsequently refer to this frequency tracker[2] as a Kalman ANF (KalmANF). We will demonstrate that the KalmANF is equivalent to a normalized LMS (NLMS) filter update [14, 15], where the regularization parameter can be expressed as the ratio of the variance of the measurement noise to the variance of the prediction error [14], both of which can be tuned accordingly. This results in a more accurate frequency tracker as compared to one that uses an ANF updated with an LMS algorithm, while maintaining a low computational

* This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 956369. The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program / ERC Consolidator Grant: SONORA (no. 773268). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information.

[1] Although certainly not limited to audio as it is also relevant in biomedical signal processing, power-line monitoring, and seismology for instance.

[2] As a consequence of using the ANF approach, we stick to the term of frequency tracking as opposed to pitch estimation since an estimation of the frequency will not necessarily correspond to a fundamental frequency, but to the frequency of the sinusoidal component contributing to most of the energy in the audio signal.

complexity and the ability to be updated on a sample-by-sample basis.

In relation to prior work, a state-space approach of the ANF has also been considered in [16], however an alternative formulation was used, and a relation with the normalized LMS was not established. In [17], an extended Kalman filter was used to update a single parameter adaptive comb filter (i.e. multiple ANFs), whereas we consider a single ANF in this paper, which allows us to have a linear state-space model. Several other Kalman filtering-based approaches to frequency tracking also exist [18, 19, 20, 21, 22], but they are not built upon an ANF.

The remainder of this paper is organized as follows. In Section 2, we review the ANF, i.e. the constrained biquad IIR filter, and how the LMS algorithm is used to update the filter coefficients. In section 3, we reformulate the problem in terms of a state-space model where the state contains the relevant filter coefficients to be updated. By applying a Kalman filter, it is then shown how the KalmANF is equivalent to using an NLMS algorithm with a well-defined time-varying regularization parameter. In section 4, we evaluate the KalmANF in comparison to its LMS-based counterpart on both simulated and realistic data, where it is demonstrated that the KalmANF outperforms the ANF frequency tracker that uses an LMS algorithm in terms of its accuracy and convergence speed.

2. LEAST-MEAN-SQUARE ADAPTIVE NOTCH FILTER

Let us consider the following signal model in the discrete-time domain, with n being the discrete-time index:

$$y(n) = A_o(n)\sin\left[n\omega_o(n) + \phi_o(n)\right] + g(n) \tag{1}$$

where $y(n)$ is a measured signal consisting of a sinusoidal component, $A_o(n)\sin\left[n\omega_o(n) + \phi_o(n)\right]$, with time-varying parameters: amplitude $A_o(n)$, phase $\phi_o(n)$, digital angular frequency $\omega_o(n) = 2\pi f_o(n)/f_s$, where $f_o(n)$ is the frequency (Hz), and f_s is the sampling frequency (Hz). This model is very broad in the sense that the remaining component, $g(n)$ can be representative of a number of signals such as a broadband desired signal, additional harmonics, or simply noise depending on the application. Given the measurement, $y(n)$, our goal is to track the time-variation of $f_o(n)$. The approach that we follow is to design an ANF that can be applied to $y(n)$ to effectively suppress the sinusoidal component and by consequence will also result in a frequency tracker.

A well-known technique of designing an ANF is to adaptively compute the parameters of a constrained IIR filter [10, 11, 12], which can be done quite efficiently using an LMS algorithm [3]. In this paper, since our signal model only consists of one sinusoidal component, we will simply consider a constrained biquad IIR filter, i.e. with two-zeros and two-poles. Firstly, let us recall the biquad filter in the z-domain without any constraints:

$$H(z^{-1}) = \frac{b_o + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2}$$

which for $b_o = 1$, can be expressed in polar coordinates (in the complex plane) in terms of a zero radius, ζ, and zero angle ω_z, and a pole radius, ρ, and pole angle, ω_p as follows

$$H(z^{-1}) = \frac{(1 - \zeta e^{j\omega_z} z^{-1})(1 - \zeta e^{-j\omega_z} z^{-1})}{(1 - \rho e^{j\omega_p} z^{-1})(1 - \rho e^{-j\omega_p} z^{-1})} \tag{3}$$

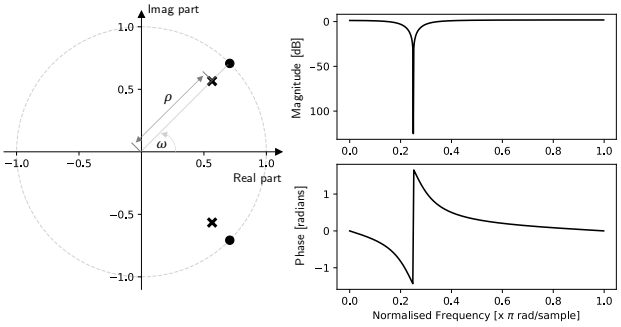


Figure 1: *(Left) Pole-zero plot of a constrained biquad IIR filter configured as a notch filter. The poles and zeros lie on the same radial line defined by $\omega = \pi/4$, where the zeros are placed on the unit circle and the poles at a distance $\rho = 0.8$. (Right) The corresponding magnitude and phase response. A notch is clearly visible at $\omega = \pi/4$ with a very narrow bandwidth due to $\rho = 0.8$.*

where $b_1 = -(\zeta e^{j\omega_z} + \zeta e^{-j\omega_z}) = -2\zeta\cos(\omega_z)$, $b_2 = \zeta^2$, $a_1 = -2\rho\cos(\omega_p)$, and $a_2 = \rho^2$. In order to convert this filter into a more suitable form where its coefficients can be adapted, two constraints need to be subsequently introduced.

The first of these constraints as proposed in [10] is to make the poles and zeros lie on the same radial line, defined by angle ω in the complex plane (see Fig. 1), i.e. $\omega_z = \omega_p = \omega$. These poles and zeros must also lie completely within the unit circle, where the zeros would be in between the poles and the unit circle in order to define a notch filter. The intuition behind this is that placing a zero near to the unit circle would attenuate all the frequency components in the neighbourhood of the angular frequency, ω, defining that particular radial line. Placing a pole on the same radial line then creates a resonance at ω, with the bandwidth of the notch filter becoming narrower as $\rho \to \zeta$.

The second constraint on the biquad filter is to let the zeros all lie on the unit circle [11] so that $\zeta = 1$. In this case the frequency component at ω would be completely attenuated and the pole at the same radial line would once again create a resonance at ω, with the bandwidth of the notch filter becoming narrower as $\rho \to 1$.

Imposing these constraints on the biquad filter of (3), results in the constrained biquad filter:

$$\begin{aligned} H(z^{-1}) &= \frac{(1 - e^{j\omega} z^{-1})(1 - e^{-j\omega} z^{-1})}{(1 - \rho e^{j\omega} z^{-1})(1 - \rho e^{-j\omega} z^{-1})} \\ &= \frac{1 - 2\cos(\omega) z^{-1} + z^{-2}}{1 - 2\rho\cos(\omega) z^{-1} + \rho^2 z^{-2}} \\ &= \frac{1 - a z^{-1} + z^{-2}}{1 - \rho a z^{-1} + \rho^2 z^{-2}} \end{aligned} \tag{4}$$

where $a \triangleq 2\cos(\omega) = 2\cos(2\pi f/f_s)$ is the only parameter we need to estimate (since it appears in both the numerator and denominator) and is directly related to the centre frequency, f, of the notch filter. Consequently, by adapting the a coefficient, the centre frequency of the notch filter also changes resulting in an ANF. The pole-zero plot and corresponding magnitude and phase response of an example constrained biquad filter is shown in Fig. 1.

In order to estimate the frequency, f_o, of the sinusoid in (1), we need to find the parameter, a, such that when the ANF is applied to the input (or measured) signal, $y(n)$, the output signal power of

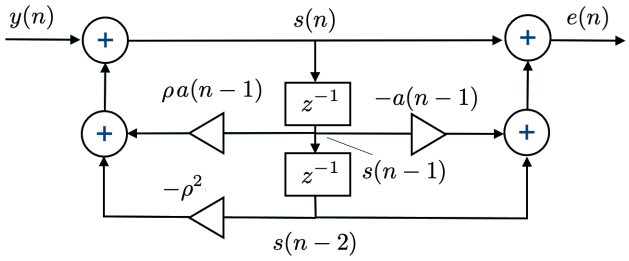


Figure 2: *Direct form II of the constrained biquad filter.*

the filter is minimal in the mean-squared sense. This would imply that the centre frequency of the ANF would have been updated to be f_o, thereby cancelling the high-energy sinusoid, resulting in a minimum mean-square output signal power.

An efficient method to estimate a can be derived by considering the direct form II of the constrained biquad filter [12] as illustrated in Fig. 2. The implementation equations are given as

$$s(n) = y(n) + \rho a(n-1)s(n-1) - \rho^2 s(n-2) \quad (5)$$
$$e(n) = s(n) - a(n-1)s(n-1) + s(n-2) \quad (6)$$

where $y(n)$ is the input to the constrained biquad filter (the measured signal from (1)), $e(n)$ is the output, and $s(n)$ is introduced as an auxiliary variable. In this form, the biquad filter is explicitly split into two sections. The first is a two-pole resonance IIR filter illustrated on the left side of Fig. 2 corresponding to the denominator of (4) and whose difference equation is given by (5). The second section is a finite impulse response (FIR) two-zero notch filter illustrated on the right side of Fig. 2, corresponding to the numerator of (4) and whose difference equation is given by (6).

We can now proceed to estimate a by minimizing the mean-squared output signal power of the filter, i.e. minimizing the mean-square of $e(n)$. In [12], it was proposed to only update the FIR section of the biquad filter, i.e. estimate the coefficient a in the FIR section, and since this a occurs in both the numerator and denominator in (4), this estimate can be simply copied to the IIR section of the biquad filter. An LMS algorithm can then be used to estimate a by making use of (6) as follows [3, 4]

$$\begin{aligned}\hat{a}(n) &= \hat{a}(n-1) + \mu\left(-\frac{\partial e^2(n)}{\partial a(n-1)}\right) \\ &= \hat{a}(n-1) + 2\mu\, s(n-1)e(n) \end{aligned} \quad (7)$$

where μ is the step size parameter. As opposed to estimating a, we could alternatively attempt to directly estimate $\omega_o = 2\pi f_o$, however this would result in a nonlinear update equation that will not have the properties of an LMS algorithm, and hence we stick to estimating a from which we can then obtain an estimate for f_o.

The algorithm for computing frequency estimates using such an LMS update is given in Algorithm 1 and we subsequently refer to the resulting frequency tracker as LMS-ANF. In Algorithm 1, N is the length of the signal, $y(n)$, and due to the $s(n-2)$ term, we simply start the for loop from $n = 2$ and initialize $s(0)$ and $\hat{a}(1), s(1)$. Since $\arccos(\hat{a}(n)/2)$ does not exist for $|\hat{a}(n)| > 2$, we have additionally imposed a constraint on the values of $\hat{a}(n)$ such that $\hat{a}(n)$ is re-initialized to zero when $|\hat{a}(n)| > 2$, i.e. we restart the algorithm[3] with an initial frequency estimate at half of the Nyquist frequency. By defining the computational complexity as the number of multiplications per recursion, from Algorithm 1, we can deduce that the LMS-ANF has a computational complexity of 11 multiplications per recursion. It should also be noted that in addition to obtaining a sample-by-sample update of the estimated frequency, we also obtain a sample-by-sample update of the output (i.e. adaptive notch-filtered input signal), $e(n)$, however we are only concerned with the former as it pertains to the frequency tracker.

As previously mentioned, by thinking of the biquad filter as consisting of a two-pole IIR resonance filter, followed by a two-zero FIR notch filter as depicted in Fig. 2, we can give the following interpretation to the algorithm. The two-pole IIR resonance filter amplifies the frequency component in $y(n)$ corresponding to the initial value of $\hat{a}(n-1)$ according to (5) so that the signal $s(n)$ would have a fairly dominant component[4] at $\hat{f}_o(n-1)$. The two-zero FIR notch filter then attempts to reduce the error, $e(n)$ by cancelling this same frequency component that was amplified in $s(n)$ as evident by (6). If the true sinusoidal component in $y(n)$ was not amplified in $s(n)$, then the two-zero FIR notch filter would yield a mean-square of $e(n)$ that remains sufficiently large. Consequently a step size according to (7) is taken to update $\hat{a}$, which corresponds to "trying" another frequency to be amplified and notched. This procedure repeats until the frequency corresponding to the true sinusoidal component in $y(n)$ is found, since applying a notch to this component will minimize the mean-square of $e(n)$.

Algorithm 1 *LMS Update of the ANF (LMS-ANF)*

Initialize $\hat{a}(1), s(0), s(1) = 0$
Set μ, ρ
for $n = 2$ to $N-1$ **do**
  $s(n) = y(n) + \rho\hat{a}(n-1)s(n-1) - \rho^2 s(n-2)$
  $e(n) = s(n) - \hat{a}(n-1)s(n-1) + s(n-2)$
  $\hat{a}(n) = \hat{a}(n-1) + 2\mu s(n-1)e(n)$
  if $|\hat{a}(n)| > 2$ **then**
    $\hat{a}(n) = 0$
  end if
  $\hat{f}_o(n) = (f_s/2\pi)\arccos(\hat{a}(n)/2)$
end for

3. KALMAN-BASED ADAPTIVE NOTCH FILTER (KALMANF)

In this section, we derive an alternative algorithm for the estimation of a in (4) using a Kalman filter. We will refer to this algorithm as KalmANF and demonstrate that it is an example within the family of normalized LMS algorithms that are based on the Kalman filter [14]. Let us firstly recall the vector form of the state-space model [23, 24]:

$$\mathbf{x}(n) = \mathbf{C}\mathbf{x}(n-1) + \mathbf{w}(n) \quad (8)$$
$$\mathbf{z}(n) = \mathbf{H}\mathbf{x}(n) + \mathbf{v}(n) \quad (9)$$

where $\mathbf{x}(n) \in \mathbb{R}^L$ is the state vector at time n, $\mathbf{C} \in \mathbb{R}^{L\times L}$ is the state-transition matrix, $\mathbf{w}(n) \in \mathbb{R}^L$ is the process noise vector,

[3] This is certainly not the only strategy to deal with out of range values for $\hat{a}(n)$, but an investigation into this aspect of the ANF is out of the scope of this work.

[4] This of course depends on the value of ρ chosen as well as the signal-to-noise ratio of $y(n)$.

which is modelled as a zero-mean, Gaussian process with covariance matrix $\mathbf{Q} \in \mathbb{R}^{L\times L}$, $\mathbf{z}(n) \in \mathbb{R}^M$ is the measurement vector, $\mathbf{H} \in \mathbb{R}^{M\times L}$ is the measurement matrix, and $\mathbf{v}(n) \in \mathbb{R}^M$ is the measurement noise vector, also modelled as a zero-mean, Gaussian process but with covariance matrix $\mathbf{R} \in \mathbb{R}^{M\times M}$.

For dynamical systems which can be described in the state-space form of (8) and (9), the state-vector at time n can be estimated using a Kalman filter. The Kalman filter consists of two steps: (i) a prediction (or update) stage and, (ii) an estimation (or measurement) stage, which are performed in a recursive manner, and are given by the following equations [23, 24]

$$\hat{\mathbf{x}}(n|n-1) = \mathbf{C}\hat{\mathbf{x}}(n-1) \tag{10}$$

$$\hat{\mathbf{P}}(n|n-1) = \mathbf{C}\hat{\mathbf{P}}(n-1)\mathbf{C}^T + \mathbf{Q} \tag{11}$$

$$\mathbf{K}(n) = \hat{\mathbf{P}}(n|n-1)\mathbf{H}^T\left(\mathbf{H}\hat{\mathbf{P}}(n|n-1)\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{12}$$

$$\mathbf{v}(n) = \mathbf{z}(n) - \mathbf{H}\hat{\mathbf{x}}(n|n-1) \tag{13}$$

$$\hat{\mathbf{x}}(n) = \hat{\mathbf{x}}(n|n-1) + \mathbf{K}(n)\mathbf{v}(n) \tag{14}$$

$$\hat{\mathbf{P}}(n) = [\mathbf{I} - \mathbf{K}(n)\mathbf{H}]\,\hat{\mathbf{P}}(n|n-1) \tag{15}$$

where $\mathbf{K}(n)$ is the Kalman gain, the notation $\hat{\mathbf{x}}(n|n-1)$ denotes a prediction of $\mathbf{x}(n)$ based on measurement samples up to time $n-1$, and the prediction error is defined as $\mathbf{x}(n) - \hat{\mathbf{x}}(n|n-1)$ with a covariance matrix, $\mathbf{P}(n)$, whose estimate is denoted as $\hat{\mathbf{P}}(n)$. The first two equations, (10) and (11), are the prediction equations, which update the state and the covariance matrix of the prediction error from measurement samples up to time $n-1$. The subsequent equations are the estimation equations. $\mathbf{v}(n)$ in (13) is also referred to as the innovation signal, which is the error between the new measurement at time n and the prediction based on measurement samples up to time $n-1$, and is used to update the state-vector estimate at time n in (14) along with the Kalman gain, $\mathbf{K}(n)$, computed in (12). The prediction error covariance matrix at time n is finally updated in (15), and the entire sequence of equations is repeated for the next time index.

By following the strategy of estimating a in the FIR section of the constrained biquad filter from Fig. 2 and copying the estimate to the IIR section, we can use (5) and (6) to define a state-space model corresponding to the form of (8) and (9) as follows:

$$\underbrace{\begin{bmatrix} a(n) \\ 1 \end{bmatrix}}_{\mathbf{x}(n)} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix} a(n-1) \\ 1 \end{bmatrix}}_{\mathbf{x}(n-1)} + \underbrace{\begin{bmatrix} w(n) \\ 0 \end{bmatrix}}_{\mathbf{w}(n)} \tag{16}$$

$$\underbrace{s(n)}_{\mathbf{z}(n)} = \underbrace{\begin{bmatrix} s(n-1) & -s(n-2) \end{bmatrix}}_{\mathbf{H}(n)} \underbrace{\begin{bmatrix} a(n) \\ 1 \end{bmatrix}}_{\mathbf{x}(n)} + \underbrace{e(n)}_{\mathbf{v}(n)} \tag{17}$$

Focusing firstly on (17) and comparing with (9), it is evident that we have defined $s(n)$ as our measurement, which is a scalar. Although we do not explicitly measure $s(n)$, it is a function of the input signal, $y(n)$, and therefore we can use (5) with $a(n-1) = \hat{a}(n-1)$ to obtain a value for $s(n)$. We can also observe that the measurement matrix, $\mathbf{H}(n)$, is now time-varying and depends on two previous measurement samples. The measurement noise vector is also simply a scalar and is the error, $e(n)$, we want to minimize in the ANF context.

Finally we can observe that the state vector, $\mathbf{x}(n)$ is a function of $a(n)$, which is the parameter that we want to estimate. With the measurement equation defined, the state equation of (16) follows directly from (8), where $\mathbf{C}$ is simply an identity matrix and $\mathbf{w}(n)$ has one non-zero value, $w(n)$, since it is only $a(n)$ that needs to be updated.

We can simply proceed to use the equations (10) - (15) to obtain an estimate for $a(n)$. However, because of the low dimensionality of the state-space equations defined in (16) and (17), we can also substitute them into (10) - (15) to obtain simpler and more intuitive expressions to understand how $a(n)$ is being estimated.

Since $\mathbf{C}$ is an identity matrix, (10) is simply

$$\hat{\mathbf{x}}(n|n-1) = \begin{bmatrix} \hat{a}(n|n-1) \\ 1 \end{bmatrix} = \begin{bmatrix} \hat{a}(n-1) \\ 1 \end{bmatrix} \tag{18}$$

We initialize the estimate of the covariance matrix of the prediction error, $\hat{\mathbf{P}}(n)$, and the covariance matrix of the process noise, $\mathbf{Q}$ with only one non-zero entry so that (11) reduces to

$$\hat{\mathbf{P}}(n|n-1) = \begin{bmatrix} \hat{p}(n|n-1) & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \hat{p}(n-1) + q & 0 \\ 0 & 0 \end{bmatrix} \tag{19}$$

where q is the variance of $w(n)$, which is a hyperparameter of the proposed algorithm.

Since the measurement equation of (17) is scalar, the covariance of the measurement noise boils down to the variance of $e(n)$, which we denote as r, another hyperparameter. Using the time-varying measurement matrix, $\mathbf{H}(n)$ from (17) and $\hat{\mathbf{P}}(n|n-1)$ from (19), the Kalman gain follows from (12) as

$$\mathbf{K}(n) = \frac{s(n-1)}{s^2(n-1) + \frac{r}{\hat{p}(n|n-1)}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{20}$$

From (14) we then obtain the update equation for the state vector. Since the second element in the state vector is always 1 and the Kalman gain is zero for this entry, we will in fact just have a scalar update equation as follows:

$$\hat{a}(n) = \hat{a}(n-1) + \frac{s(n-1)}{s^2(n-1) + \frac{r}{\hat{p}(n|n-1)}} e(n) \tag{21}$$

where using (13), $e(n)$ is given by

$$e(n) = s(n) - s(n-1)\hat{a}(n-1) + s(n-2) \tag{22}$$

which is identical to (6) but with $a(n-1) = \hat{a}(n-1)$.

Finally from (15), the update of the first and only non-zero element of the covariance matrix of the prediction error is

$$\hat{p}(n) = \left(1 - \frac{s^2(n-1)}{s^2(n-1) + \frac{r}{\hat{p}(n|n-1)}}\right) \hat{p}(n|n-1) \tag{23}$$

It can now be seen that (21) is indeed in the form of a normalized LMS (NLMS) filter update [14, 15], with a time-varying step size of $1/\left[s^2(n-1) + r/\hat{p}(n|n-1)\right]$, where the term $s^2(n-1)$ provides the normalization and $r/\hat{p}(n|n-1)$ acts as a time-varying regularization parameter. As opposed to having to choose this regularization parameter in a heuristic manner [25], an optimal value is now defined in the Kalman filter context as the ratio of the variance of the measurement noise to the variance of the prediction error obtained from measurements samples up to time $n-1$ [14]. We also note that a similar, yet time-invariant expression for the NLMS parameter was obtained in a Bayseian framework in [26, 27].

A summary of the KalmANF frequency tracker is given in Algorithm 2. As with the LMS-ANF, due to the $s(n-2)$ term, we simply start the for loop from $n = 2$ and initialize $s(0), \hat{a}(1), s(1)$, and $\hat{p}(1)$. The constraints are also imposed on $\hat{a}(n)$ to ensure $\arccos(\hat{a}(n)/2)$ exists. We can also deduce that the KalmANF has a computational complexity of 14 multiplications per recursion (in line 7, only two multiplications are counted since $k(n)$ would have been computed in line 4), which is the same order of magnitude as the LMS-ANF.

Algorithm 2 *Kalman-based/NLMS update of the ANF (KalmANF)*

Initialize $s(0), s(1), \hat{a}(1), \hat{p}(1) = 0$
Set r, q, ρ
1: **for** $n = 2$ to $N-1$ **do**
2: $\quad \hat{p}(n|n-1) = \hat{p}(n-1) + q$
3: $\quad s(n) = y(n) + \rho\hat{a}(n-1)s(n-1) - \rho^2 s(n-2)$
4: $\quad k(n) = \frac{s(n-1)}{s^2(n-1)+\frac{r}{\hat{p}(n|n-1)}}$
5: $\quad e(n) = s(n) - \hat{a}(n-1)s(n-1) + s(n-2)$
6: $\quad \hat{a}(n) = \hat{a}(n-1) + k(n)e(n)$
7: $\quad \hat{p}(n) = \left(1 - \frac{s^2(n-1)}{s^2(n-1)+\frac{r}{\hat{p}(n|n-1)}}\right)\hat{p}(n|n-1)$
8: $\quad$ **if** $|\hat{a}(n)| > 2$ **then**
9: $\quad\quad \hat{a}(n) = 0$
10: $\quad$ **end if**
11: $\quad \hat{f}_o(n) = (f_s/2\pi)\arccos(\hat{a}(n)/2)$
12: **end for**

4. EVALUATION

We evaluate the performance of the KalmANF in relation to the LMS-ANF using both simulated and realistic data. We firstly use simulated data so that we can compare frequency estimates to a ground truth, and consequently make observations on the general performance of the KalmANF in relation to the LMS-ANF. We then apply the algorithms to realistic acoustic data and compare how well a dominant frequency component is tracked. For both the LMS-ANF and the KalmANF, it was always the case that $-2 \leq \hat{a}(n) \leq 2$ so that the constraint of $\hat{a}(n) = 0$ when $|\hat{a}(n)| > 2$ was never executed. We do not consider an evaluation of the KalmANF against other types of pitch/frequency trackers that are not based upon the ANF as this is beyond the scope of this work. The code used to generate all of the results that follow is available at [28].

4.1. Simulated Data

4.1.1. Instantaneous change in frequency

In this first simulation, we observe the performance of the KalmANF and LMS-ANF for the situation where there is an instantaneous change in frequency of the sinusoidal component of the input signal, so as to initially gauge how well the algorithms are suited for rapidly changing sinusoidal components. We synthesized an input signal of duration 4 s consisting of a sinusoid embedded in white Gaussian noise at a sampling frequency of 8 kHz and with a signal to noise ratio of 2 dB. For the first 2 s, the frequency of the sinusoid was 1500 Hz, after which the frequency was instantaneously changed to 500 Hz for the remainder of the signal. The amplitude of the sinusoid was 0.5, and its initial phase was set to zero.

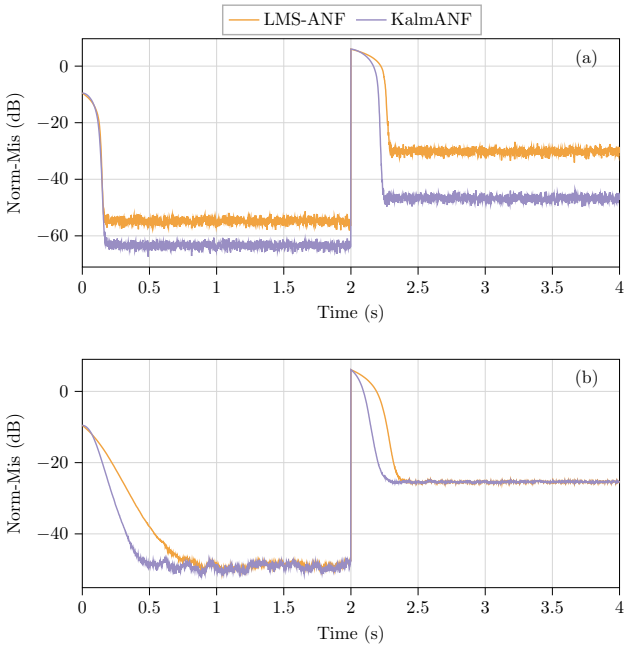


Figure 3: *Averaged Norm-Mis from the LMS-ANF and KalmANF across 100 different signal realizations consisting of a sinusoid whose frequency instantaneously changes in frequency at 2* s *and white Gaussian noise at an SNR of* 2 dB. *(a) Using an ANF with* $\rho = 0.95$*, (b) Using an ANF with* $\rho = 0.7$*. In both cases,* $\mu = 1 \cdot 10^{-3}$ *for the LMS-ANF and* $q = 8 \cdot 10^{-5}$*, and* $r = 10$ *for the KalmANF.*

Therefore, at any point in time, this signal corresponded to the signal model of (1), where $A_o(n) = 0.5$, $\phi_o(n) = 0$, and $f_o(n)$ varied according to the aforementioned frequency of the sinusoid.

We applied both the LMS-ANF and KalmANF algorithms to this input signal to estimate the frequency, $\hat{f}(n)$, of the sinusoid over time. In order to quantify the performance of both algorithms, we computed the normalized misalignment (error) between the estimated frequency and the true frequency as follows:

$$\text{Norm-Mis}(n) = 20\log_{10}\frac{|f_o(n) - \hat{f}_o(n)|}{f_o(n)} \tag{24}$$

We repeated this procedure for 100 different realizations of white Gaussian noise and averaged the normalized misalignment across the different realizations. Fig. 3 shows this averaged Norm-Mis for two values of ρ. In Fig. 3 (a), $\rho = 0.95$ for a narrow bandwidth notch filter, and in Fig. 3 (b), $\rho = 0.7$ for a wider bandwidth notch filter. In both simulations, $\mu = 1 \cdot 10^{-3}$ for the LMS-ANF and $q = 8 \cdot 10^{-5}$, and $r = 10$ for the KalmANF. These parameters were chosen such that the initial convergence rates for $\rho = 0.95$ of both algorithms were approximately similar as shown in Fig. 3 (a).

Despite the similar initial convergences rates, however, in Fig. 3 (a) we can firstly observe that the KalmANF converges to a lower steady state Norm-Mis than the LMS-ANF both during the first 2 s and after the instantaneous change in the frequency of the input signal. We can also observe that the KalmANF converges faster than the LMS-ANF after this instantaneous change in frequency, without any change to the aforementioned algorithm parameters.

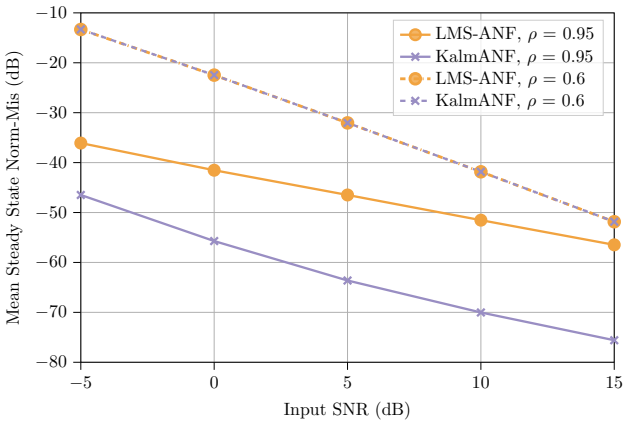


Figure 4: *Mean of the steady state Norm-Mis for the LMS-ANF and KalmANF (after averaging over 100 different realizations) as a function of the input SNR. The signal was 4* s *and consisted of a sinusoid embedded in white Gaussian noise.*

In Fig. 3 (b) where $\rho = 0.7$, the KalmANF now converges to a similar steady state Norm-Mis as the LMS-ANF, but however at a faster rate both during the first 2s and after the instantaneous change in the frequency.

It should be noted that since we have initialized $\hat{a} = 0$, the initial frequency estimate is half of the Nyquist frequency ($f_s/4$). Hence the tracking of low frequencies at higher sampling frequencies can result in longer convergence times, which would be particularly problematic for the LMS-ANF due to its fixed step size. If there is some prior knowledge of the dominant frequency content of the signal however, the sampling frequency could be chosen so that the initial frequency estimate is close to this dominant frequency.

4.1.2. *Influence of ρ and input SNR*

In order to observe the influence of ρ and the input SNR on the performance of the algorithms, in this simulation, we used a 4 s input signal consisting of a single sinusoidal component with $A_o(n) = 0.5$, $\phi_o(n) = 0$, and $f_o(n) = 868$Hz embedded in white Gaussian noise at $f_s = 8$kHz. We ran the KalmANF and the LMS-ANF for $\rho = 0.6$ and $\rho = 0.95$ for a range input SNRs = $\{-5, 0, 5, 10, 15\}$ dB. For each input SNR and ρ, we ran the algorithms using 100 different realizations of white Gaussian noise and averaged the Norm-Mis across the different realizations. We then computed the mean of the steady state Norm-Mis using the last 2s of the averaged Norm-Mis across the different realizations. In other words, we computed the mean of a converged region of the Norm-Mis such as that between 3 s and 4 s in Fig. 3 (a). For the LMS-ANF, $\mu = 1 \cdot 10^{-3}$, and for the KalmANF, $r = 10$, but q was varied such that for the different values of ρ and all input SNRs, the convergence rates of both the LMS-ANF and KalmANF were approximately the same. For $\rho = 0.95$, $q = \{10, 4.5, 2.5, 1.9, 1.7\} \cdot 10^{-5}$, and for $\rho = 0.6$, $q = \{4, 2.5, 2, 2, 2\} \cdot 10^{-5}$, where each value of q corresponds to the particular input SNR in the range $\{-5, 0, 5, 10, 15\}$ dB. As can be seen from (21), the ratio $r/\hat{p}(n|n-1)$ directly affects the time-varying step size, and hence can be tuned to obtain a desired convergence speed by varying r and/or q. In general, when there is more uncertainty in the system model, such as in low SNR conditions, q should be set to a larger value (as was done in this simulation), which would consequently yield larger step sizes, allowing the algorithm to accommodate for larger deviations from the system model.

Fig. 4 shows the mean of the steady state Norm-Mis plotted against the input SNR for the different algorithms and different values of ρ. We can observe that for the larger value $\rho = 0.95$, the KalmANF outperforms the LMS-ANF by achieving a lower steady state Norm-Mis, and hence a more accurate frequency estimate, whereas for the smaller value of $\rho = 0.6$, both algorithms achieve the same the steady state Norm-Mis. In general however, for both algorithms, the results suggest that larger values of ρ are a preferable choice as they result in a lower steady state Norm-Mis. As expected, we can also observe that the performance of all algorithms degrades as the SNR decreases, and is particularly poor for the smaller values of ρ at very low input SNRs.

4.2. Real data

In this section, we evaluate the KalmANF using two realistic acoustic signals, both of which conform to the model in (1), but where the signal represented by $g(n)$ is different for each case.

4.2.1. *Musician Wren*

The first signal we consider is that of a musician wren (Cyphorhinus arada), a bird in the family of brown passerine birds, known for its melodious birdsong that spans a considerable frequency range. Fig. 5 (a) shows the spectrogram of an excerpt of musician wren recorded in Uiramutã, Brazil taken from www.xeno-canto.org[5]. As can be observed, the birdsong is fairly sinusoidal and spans a range of about 2 kHz, with substantial and almost instantaneous jumps in frequency. The remaining component of the signal, $g(n)$, in this case is the noise of the outdoor environment, which is not fully white or Gaussian, and in fact consists of impulsive-type sounds presumably due to rainfall.

In Fig. 5 (b), the estimated frequency tracks at each sample using the LMS-ANF and the KalmANF, both with $\rho = 0.95$ are overlaid on the same spectrogram of 5 (a), so as to visualize how well the frequency of the birdsong is being tracked. For the LMS-ANF, $\mu = 0.3$ and for the KalmANF, $q = 8 \cdot 10^{-3}$ and $r = 1$. The LMS-ANF could not be tuned to match the convergence rate of the KalmANF since larger values of μ resulted in an unstable filter. This demonstrates that firstly, the KalmANF is able to converge faster than the LMS-ANF. Secondly, it can also be observed that the KalmANF is able to quickly adapt to the rapid changes in frequency of the birdsong. In most cases, the KalmANF appears to yield a more accurate frequency estimate as compared to the LMS-ANF. In particular, around 2.3 s, it can be seen that the impulsive background noise negatively impacts the frequency estimation of the LMS-ANF, whereas the KalmANF estimate remains fairly stable by comparison. It is also noted that there were no sinusoidal components to be tracked in the first 1.2s and between approximately 3.5s and 4.7s, and hence frequency estimates during these times are meaningless. It nevertheless does provide some insight into the convergence of the filters when there is no dominant sinusoidal component.

[5]This recording has a catalogue number XC 513058 and was recorded by Gabriel Leite.

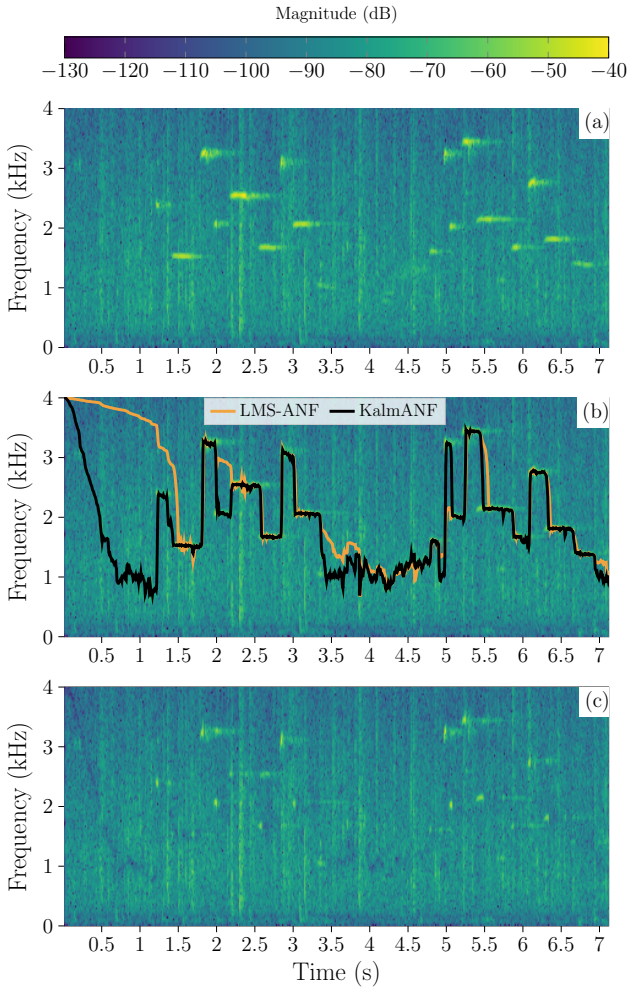


Figure 5: *(a) Spectrogram from an excerpt of a musician wren (Cyphorhinus arada). (b) Overlaid frequency tracks, i.e. sample-by-sample frequency estimates for the LMS-ANF and KalmANF with* $\rho = 0.95, \mu = 0.3, q = 8 \cdot 10^{-3}, r = 1$. *(c) Output signal from the KalmANF.*

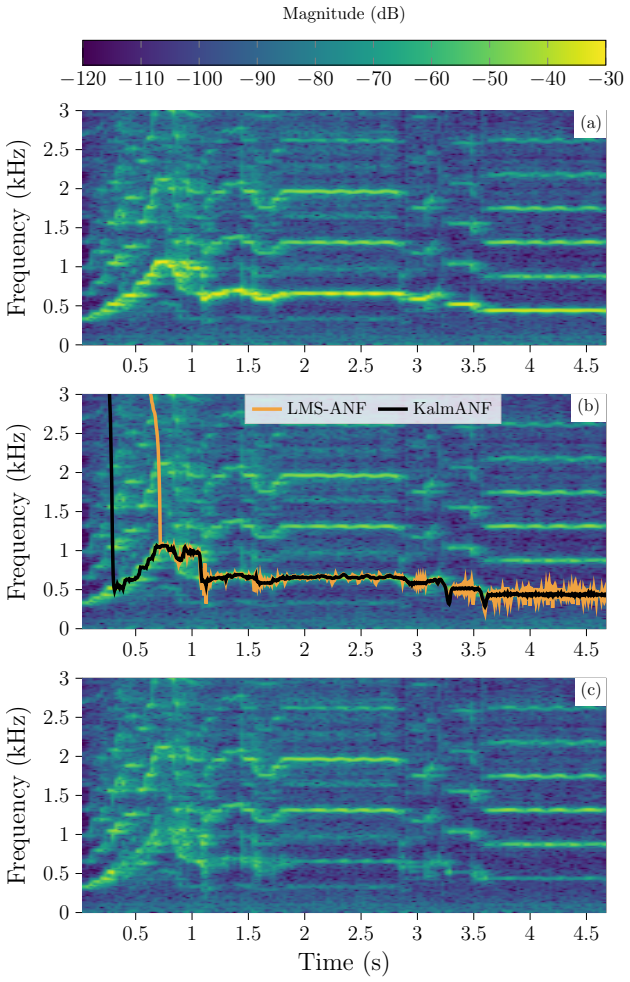


Figure 6: *(a) Spectrogram from a short flute passage. (b) Overlaid frequency tracks for the LMS-ANF and KalmANF with* $rho = 0.93, \mu = 5 \cdot 10^{-3}, q = 5 \cdot 10^{-4}, r = 10$. *(c) Output signal from the KalmANF.*

Finally, to give a better impression of the performance of the KalmANF, Fig. 5 (c) shows the spectrogram of the error signal (output) of the KalmANF defined in (22). Upon comparison with Fig. 5 (a), it can be seen that the sinusoidal component of the birdsong has been significantly attenuated, implying that the frequency of the birdsong has been accurately tracked.

4.2.2. Flute

Here we consider a short passage from a flute with both rapid and slow frequency changes taken from freesound[6]. Fig. 6 (a) shows the spectrogram of this signal (which was converted to a mono signal and resampled to 16 kHz). There is a strong sinusoidal component along with several harmonics. Hence the signal still conforms to the model of (1) with a dominant sinusoidal component, but the remaining part of the signal, $g(n)$, would now consist of the harmonics and any background noise.

In Fig. 6 (b) the estimated frequency tracks at each sample using the LMS-ANF and the KalmANF, both with $\rho = 0.93$ are overlaid on the spectrogram of 6 (a). For the LMS-ANF, $\mu = 5 \cdot 10^{-3}$ and for the KalmANF, $q = 5 \cdot 10^{-4}$ and $r = 10$. Just as in the birdsong example, the LMS-ANF could not be tuned to match the convergence rate of the KalmANF since larger values of μ resulted in an unstable filter. Hence, the KalmANF has a faster convergence rate than the LMS-ANF, and is able to quickly adapt to the frequency changes with a fairly accurate frequency track. Once the LMS-ANF has converged, it generally follows a similar frequency track to that of the KalmANF, but with a much larger variance around the fundamental frequency, particularly toward the end of the signal. Similar to Fig. 5 (c), Fig. 6 (c) shows the spectrogram of the error signal output of the KalmANF, where we can observe a significant attenuation of the dominant sinusoidal component.

[6]"Flute trill" by juskiddink (https://freesound.org/people/juskiddink/) licensed under CCBY 4.0.

5. CONCLUSION

We have developed a fast frequency tracker that is based on updating a single parameter of an adaptive notch filter (ANF) with a Kalman filter (KalmANF). Whereas this parameter is conventionally updated using a least-mean-square (LMS) algorithm, in this work, we reformulate the ANF (which is a constrained biquadratic filter) in terms of a state-space model, where the state contains the parameter to be updated. By using a Kalman filter to update the state, we have also demonstrated that such an update is equivalent to a normalized LMS (NLMS) filter update where the regularization parameter can be expressed as the ratio of the variance of the measurement noise to the variance of the prediction error. Using both simulated and realistic data, it was shown that in comparison to the ANF-based frequency tracker using an LMS algorithm, the KalmANF resulted in a more accurate performance, with a faster convergence rate, while maintaining a low computational complexity and the ability to be updated on a sample-by-sample basis.

6. REFERENCES

[1] D. Gerhard, *Pitch Extraction and Fundamental Frequency: History and Current Techniques*, Technical Report TR-CS 2003-06, Department of Computer Science, University of Regina, Regina, Canada, Nov. 2003.

[2] M. G. Christensen and A. Jakobsson, *Multi-Pitch Estimation*, Morgan and Claypool Publishers, 2009.

[3] J. M. Kates, "Feedback Cancellation in Hearing Aids: Results from a Computer Simulation," *IEEE Trans. Signal Process.*, vol. 39, no. 3, pp. 553–562, 1991.

[4] J. A. Maxwell and P. M. Zurek, "Reducing Acoustic Feedback in Hearing Aids," *IEEE Trans. Speech Audio Process.*, vol. 3, no. 4, pp. 304–313, 1995.

[5] E. Benetos, S. Dixon, Z Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Process. Mag.*, vol. 36, no. 1, pp. 20–30, 2019.

[6] M. G. Christensen, "A method for low-delay pitch tracking and smoothing," in *Proc. 2012 IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP '12)*, 2012, pp. 345–348.

[7] D. Talkin, "A robust algorithm for pitch tracking (RAPT)," in *Speech Coding and Synthesis*, W. B. Kleijn and K. K. Paliwal, Eds., chapter 14, pp. 495–518. Elsevier Science, Amsterdam, The Netherlands, 1995.

[8] J. Tabrikian, S. Dubnov, and Y. Dickalov, "Maximum a-posteriori probability pitch tracking in noisy environments using harmonic model," *IEEE Trans. Speech Audio Process.*, vol. 12, pp. 76 – 87, 2004.

[9] L. Shi, J. K. Nielsen, J. R. Jensen, M. A. Little, and M. G. Christensen, "Robust Bayesian pitch tracking based on the harmonic model," *IEEE Trans. Audio Speech Lang. Process.*, vol. 27, no. 11, pp. 1737–1751, 2019.

[10] D.V. Bhaskar Rao and S.Y. Kung, "Adaptive Notch Filtering for the Retrieval of Sinusoids in Noise," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 4, pp. 791–802, 1984.

[11] A. Nehorai, "A minimal parameter adaptive notch filter with constrained poles and zeros," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 4, pp. 983–996, 1985.

[12] J. M. Travassos-Romano and M. Bellanger, "Fast Least Squares Adaptive Notch Filtering," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 9, pp. 158–161, 1988.

[13] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[14] P. A. C. Lopes and J. B. Gerald, "New Normalized LMS Algorithms Based on the Kalman Filter," in *2007 IEEE Int. Symp. Circuits Syst.*, may 2007, vol. 1, pp. 117–120.

[15] S. Haykin, *Adaptive filter theory*, Prentice Hall, Upper Saddle River, NJ, 4th edition, 2002.

[16] R. Panchalard, J. Koseeyaporn, and P. Wardkein, "State-Space Kalman Adaptive IIR Notch Filter," in *Proc. 2006 Int. Conf. on Commun., Circuits Syst.*, 2006, vol. 1, pp. 206–210.

[17] N. G. Chernoguz, "A single-parameter adaptive comb filter," in *Proc. 2001 IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP '01)*, 2001, vol. 6, pp. 3749–3752.

[18] H. Hajimolahoseini, S. Gazor, and R. Amirfattahi, "A robust and fast method for estimating and tracking the instantaneous fundamental frequency of audio signals," in *Proc. 2016 IEEE Canadian Conf. Elect. and Comput. Eng. (CCECE)*, may 2016, vol. 2016-Octob, pp. 1–4.

[19] L. Shi, J. K. Nielsen, J. R. Jensen, M. A. Little, and M. G. Christensen, "A Kalman-based fundamental frequency estimation algorithm," in *Proc. 2017 IEEE Workshop Appls. Signal Process. Audio Acoust. (WASPAA '17)*, 2017, pp. 314–318.

[20] S. Bittanti and S. M. Savaresi, "On the parameterization and design of an extended Kalman filter frequency tracker," *IEEE Trans. Automatic Control*, vol. 45, no. 9, pp. 1718–1724, 2000.

[21] B. F. La Scala, R. R. Bitmead, and B. G. Quinn, "An extended Kalman filter frequency tracker for high-noise environments," *IEEE Trans. Signal Process.*, vol. 44, no. 2, pp. 431–434, 1996.

[22] O. Das, J. Smith, and C. Chafe, "Improved real-time monophonic pitch tracking with the extended complex kalman filter," *J. Audio Eng. Soc.*, vol. 68, no. 1/2, pp. 78–86, 2020.

[23] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, 1997.

[24] S. V. Vaseghi, *Advanced Digital Signal Processing and Noise Reduction*, John Wiley & Sons, 2008.

[25] J. Benesty, C. Paleologu, and S. Ciochină, "On Regularization in Adaptive Filtering," *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, no. 6, pp. 1734–1742, 2011.

[26] T. van Waterschoot, G. Rombouts, and M. Moonen, "MSE optimal regularization of APA and NLMS algorithms in room acoustic applications," in *Proc. 2006 Int. Workshop Acoustic Echo Noise Control (IWAENC '06)*, 2006.

[27] T. van Waterschoot, G. Rombouts, and M. Moonen, "Optimally regularized adaptive filtering algorithms for room acoustic signal enhancement," *Signal Processing*, vol. 88, no. 3, pp. 594–611, 2008.

[28] R. Ali, "KalmANF: A frequency tracker based on a kalman filter update of a single parameter adaptive notch filter," *GitHub repository*, 2023, `https://github.com/randyaliased/KalmANF`.

LOW-COST NUMERICAL APPROXIMATION OF HRTFS: A NON-LINEAR FREQUENCY SAMPLING APPROACH

Maurício do V. M. da Costa *

MTDML, Institute for Musicology and Music Pedagogy
University of Osnabrück
Osnabrück, Germany
madovalemade@uni-osnabrueck.de

Luiz W. P. Biscainho

DEL/Poli & PEE/COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
wagner@smt.ufrj.br

Michael Oehler

MTDML, Institute for Musicology and Music Pedagogy
University of Osnabrück
Osnabrück, Germany
michael.oehler@uni-osnabrueck.de

ABSTRACT

Head-related transfer functions (HRTFs) describe filters that model the scattering effect of the human body on sound waves. In their discrete-time form, they are used in acoustic simulations for virtual reality (VR) or augmented reality (AR), and since HRTFs are listener-specific, the use of individualized HRTFs allows achieving more realistic perceptual results. One way to produce individualized HRTFs is by estimating the sound field around the subjects' 3D representations (meshes) via numerical simulations, which compute discrete complex pressure values in the frequency domain in regular frequency steps. Despite the advances in the area, the computational resources required for this process are still considerably high and increase with frequency. The goal of this paper is to tackle the high computational cost associated with this task by sampling the frequency domain using hybrid linear-logarithmic frequency resolution. The results attained in simulations performed using 23 real 3D meshes suggest that the proposed strategy is able to reduce the computational cost while still providing remarkably low spectral distortion, even in simulations that require as little as 11.2% of the original total processing time.

1. INTRODUCTION

The main goal of 3D acoustic simulations is to represent acoustic sources and environments as naturally as possible to the listener. To this end, human morphology must be taken into account since it imposes specific spectral distortions on the incident sound, both in magnitude and phase, that can be interpreted by the brain to, for example, estimate the direction of arrival (DOA) and distance, or identify the timbre of sound sources [1, 2].

For instance, the sound produced by a source positioned to the left side of the listener hits the left ear first, and arrives delayed and muffled (i.e. with attenuated high frequencies) at the right ear due to the greater distance and the acoustic shading effect of the head, respectively. These effects originate two among the spectral cues that help us locate sound sources in space: the interaural time difference (ITD) and the interaural level difference (ILD), both frequency dependent. Note that when a given source is equidistant from both ears, e.g. exactly in front or behind the listener, ideally, the listener can only rely on (simultaneous) spectral distortions in level to estimate its position. It has been shown that small movements of the head have a great impact in helping estimate such spatial positions and avoid front/back confusion, which highlights the relevance of even subtle differences between the sounds captured by the listener's left and right ears [2].

Head-related transfer functions (HRTFs) describe the mentioned body's acoustic effect over sound being emitted from any direction in the 3D space until impinging the ear canal entrances or the eardrums. Their digital versions provide an interface between the virtual acoustic field and the listener, thus producing a pair of signals (one for each ear) that will be converted back to the analog domain and reproduced for the listener by means of some acoustic transducer, such as a pair of headphones [2].

Although general-purpose HRTFs have been widely used due to the difficulty of acquiring data from specific listeners in an everyday situation quickly, reliably, and comfortably, individualized HRTFs tend to produce more realistic acoustic simulations for virtual reality (VR) or augmented reality (AR) [2, 3, 4]. For this reason, the research community has been working for decades on ways to personalize HRTFs, having as goals the increase in the accuracy of the HRTFs produced and the ease of implementation of the corresponding techniques [1, 5, 2].

Among the various existing techniques to accomplish this task, one way to produce individualized HRTFs without the need for any audio equipment or acoustically treated environment is by numerically calculating the sound field around the subjects' 3D representation [2, 6, 7, 8]. In this approach, the discrete complex pressure is estimated in the frequency domain within regular frequency steps for a set of given positions in space. In terms of equipment, this strategy only requires a device to capture the user's geometry and a computer to process the data.

Some important advances in this area allow for faster calculations, such as the widely adopted boundary element method (BEM) accelerated with the fast-multipole method and coupled with the collocation method [2, 6, 7]. Following this line, a recent strategy proposed in [9] resorts to an automatic mesh-grading procedure as a way to reduce the meshes, optimizing them in terms of computational load. The mentioned approaches can be considered *numerical approximations*, since they reduce the need for computational resources at the cost of introducing some (manageable) spectral distortion.

Nevertheless, the computational cost required for simulating (or approximating) HRTFs is still considerably high: computing an HRTF for a single user might take several hours on a powerful contemporary computer. Hence, this procedure is not applicable in daily-life situations where only moderate computational resources are available for the task, e.g. when using smartphones, tablets, or

* This work was funded by Volkswagen Foundation (VolkswagenStiftung), Germany (grant no. 96 881).

personal computers. For this reason, reducing the cost of numerical simulations is still a goal to be pursued.

One aspect of numerical simulations that, to the best of the authors' knowledge, is not explored in the literature concerns the dependence of computing time on frequency: a considerable part of the processing is performed at the higher end of the frequency spectrum. As will be shown in Section 3, when simulating HRTFs in a frequency range up to $22.05\,\text{kHz}$ (i.e. at a sampling rate $r_\text{s} = 44.1\,\text{kHz}$), only roughly 10% of the total processing time is spent to simulate the frequency spectrum up to $10\,\text{kHz}$, on average. Although some details in the range above $10\,\text{kHz}$ can still contribute to localization accuracy, human hearing presents a *quasi*-logarithmic resolution in frequency, thus exhibiting poorer resolution at high-frequencies; this suggests it may not be reasonable to allocate 90% of the computational cost to the simulation of the upper-frequency range. For instance, perceptual audio coders have taken advantage of low human auditory resolution at high frequencies to drastically compress audio files while still providing transparent results, i.e. indistinguishable from the original uncompressed files [10].

In [11], the authors also investigate the use of a non-linear sampling approach, which follows the equivalent rectangular bandwidth (ERB) scale, for smoothing purposes. Nevertheless, that approach presents two main disadvantages compared to our proposal, namely: (i) the incapability of reliably simulating/estimating the phase, and (ii) the fact that it requires a higher computational cost than our approach, for a comparable spectral distortion, due to the unnecessarily high resolution used at low/mid frequencies.

In this context, the objective of this paper is to propose an alternative, non-linear approach to frequency sampling that could reduce the cost of numerical simulations while still preserving the relevant spectral details from a perceptual perspective, by performing a perceptually justifiable numerical approximation.

The rest of the text is organized as follows: in Section 2, a theoretical background of numerical simulation is disclosed, followed by a description of the proposed hybrid-resolution method; then, Section 3 presents the experiments conducted to investigate the trade-off between reducing the simulation time and increase the spectral distortion of HRTFs when using the proposed approach; at last, in Section 4, the conclusions of this manuscript are drawn along with a description of future work.

2. METHODOLOGY

2.1. Numerical Simulation of HRTFs

Computing individualized HRTFs via numerical simulation requires a 3D geometrical representation (a 'mesh') of the subject, which consists of a discrete and finite set of points in space forming triangular faces.[1] The simulation thus calculates the scattering effect of the incoming sound wave over the body geometry and the HRTFs are obtained as a set of complex pressure bins in frequency, computed for specific pairs source/receiver locations in the 3D space [2, 6, 7]. The sound sources are usually distributed on the surface of a sphere centered at the origin of the Cartesian space, within a grid that can, for instance, be uniform or distribute the locations according to the human perceptual spatial resolution [2]. The receiver positions are typically faces of the mesh that best represent the occluded ear canal entrances or some point inside the open ear canals, depending on what is to be modeled.

Considering that the mesh is positioned in such a way that the midpoint between its left and right ear canal entrances coincides with the origin of the 3D space and the head is in line with the x axis [2], i.e. the receivers are approximately on the y axis, the (left, right) HRTF pair can be described as

$$\begin{aligned} \text{H}_\text{L}[\mathbf{x}^*, f, s] &= \frac{p_\text{L}[\mathbf{x}^*, f, s]}{p_0[f]} \quad \text{and} \\ \text{H}_\text{R}[\mathbf{x}^*, f, s] &= \frac{p_\text{R}[\mathbf{x}^*, f, s]}{p_0[f]}, \end{aligned} \tag{1}$$

where p_L and p_R denote the sound pressure at the respective left and right receiver points, $\mathbf{x}^*$ denotes the sound-source location,[2] f denotes the frequency, and s indexes the s-th subject. Both p_L and p_R are normalized w.r.t. the reference sound pressure p_0, measured at the origin *in the absence of the head*.

All current methods for numerical calculation of HRTFs are based on solutions of the Helmholtz equation, which describes the propagation of sound waves in the free field around the object of interest [2]:

$$\nabla^2 p(\mathbf{x}) + \kappa^2 p(\mathbf{x}) = q(\mathbf{x}), \mathbf{x} \in \Omega_e, \tag{2}$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ is the Laplacian operator in the Cartesian 3D space; $p(\mathbf{x})$ denotes the complex sound pressure at the location $\mathbf{x}$; $\kappa = 2\pi f/c$ denotes the wave number, which is calculated using the frequency f and speed of sound c; Ω_e denotes the exterior domain around the object described by the mesh; and $q(\mathbf{x})$ denotes the complex contribution of the sound source in the acoustic field around the object. The simulation of HRTFs is performed by solving this equation for frequencies in the audible spectrum with regular frequency steps. In this context, since the sampling procedure is performed in the frequency domain, it would be inaccurate to adopt the term *sampling period*. Thus, to avoid ambiguity in notation, we will denote the frequency step mentioned as F_s, and the sampling rate related to the sampling of a signal in the time domain as r_s.

There exist different methods to solve this problem, as mentioned, the most prominent being: the finite-element method (FEM), which solves the Helmholz equation considering the object or the spatial domain around it as a volume; the finite-difference time-domain method (FDTD), which follows a similar approach to the FEM, but in the time domain; and the boundary-element method (BEM), which uses a special set of test functions in the weak formulation of the Helmholtz equation, namely the Green's function, and offers the advantage of only considering the surface of the object. This solution allows for the use of speed-up strategies, such as the fast-multi-pole method (FMM), the collocation with constant elements, and the reciprocity approaches. In addition, the resulting linear system of equations can then be solved using an iterative equation solver [2]. A complete description of methods for numerical calculation is out of the scope of this work (for more information on this topic, see [2, 6, 7, 8, 12]). This paper will focus on working with the BEM, which is the fastest and most commonly used solver for this purpose.

[1] Meshes are not necessarily described by triangles, but, in the case of numerical simulations, since connecting three points is guaranteed to define a flat surface, this choice offers the computational advantage of having a unique normal vector.

[2] The source position can be described either directly in the Cartesian 3D space as a three-dimensional vector or as a distance in meters and a direction, e.g. using azimuth and elevation angles.

Since the simulation is performed for each frequency f independently, the solution for a given set of regularly spaced frequencies $f \in \mathcal{F}$, where

$$\mathcal{F} = \{f \mid 0 \leq f \leq r_s/2, f_{i+1} - f_i = F_s\}, \tag{3}$$

can be paralleled and then processed to produce HRTFs for the desired source positions $\mathbf{x}$. Such independence is useful not only for speeding up the process but also for allowing us to arbitrarily chose which frequencies to compute the pressure with. This will be exploited in our solution to mitigate the high computational cost of numerical simulations using the BEM by estimating fewer frequency bins at high frequencies and then interpolating the resulting spectra to restore the regular frequency grid required to properly describe HRTFs and HRIRs. The next section presents the non-linear sampling approach proposed.

2.2. Numerical Approximation of HRTFs: Hybrid Linear-Logarithmic Sampling

On top of all the techniques already available to speed up the simulation of HRTFs, we aim to further reduce its computational cost. To this end, we will exploit the following particular observation: the processing cost grows nearly exponentially with frequency. As a result, a high percentage of the computing time is concentrated on simulating very high frequencies, which contrasts with the decreasing resolution of human hearing with frequency [10]. This suggests that most of the computational burden required by the simulation methods could be reduced without noticeable effects.

In line with this rationale, we propose a sampling approach that yields a hybrid linear-logarithmic frequency resolution. The main idea is to divide the frequency spectrum into two bands around a given crossover frequency f_c, keeping for simulations the usual fixed frequency steps F_s in the lower band and adopting in the upper band a logarithmic frequency spacing, specified by a number B of bins/octave. More rigorously, the set of frequencies to be simulated can be defined as $\hat{\mathcal{F}} = \{\mathcal{F}_{\text{lin}}, \mathcal{F}_{\text{log}}\}$, where

$$\begin{aligned} \mathcal{F}_{\text{lin}} &= \{f \mid 0 \leq f \leq f_c, f_{i+1} - f_i = F_s\} \text{ and} \\ \mathcal{F}_{\text{log}} &= \{f \mid f = f_{\max} 2^{-l/B}, 0 \leq l \leq B \log_2(f_{\max}/f_c)\}. \end{aligned} \tag{4}$$

The maximum frequency $f_{\max}$ to be simulated is defined by $f_{\max} = r_s/2$ and the quantity $\log_2(f_{\max}/f_c)$ indicates the number of octaves N_{octs} of the superior spectrum. This ensures that $f_{\max}$ will be simulated and the remaining frequencies will be sampled decreasing exponentially from $f_{\max}$ to f_c. The proposed HRTF is then first simulated using this non-linear frequency sampling, producing

$$\begin{aligned} \hat{\mathrm{H}}_{\mathrm{L}}(\mathbf{x}^*, \hat{f}, s) &= \frac{p_{\mathrm{L}}(\mathbf{x}^*, \hat{f}, s)}{p_0[\hat{f}]} \text{ and} \\ \hat{\mathrm{H}}_{\mathrm{R}}(\mathbf{x}^*, \hat{f}, s) &= \frac{p_{\mathrm{R}}(\mathbf{x}^*, \hat{f}, s)}{p_0[\hat{f}]}, \end{aligned} \tag{5}$$

where $\hat{f} \in \hat{\mathcal{F}}$.

Naturally, adopting the logarithmic scale only for high frequencies avoids applying an unnecessarily high resolution for the lower part of the spectrum, which would increase the computational cost without even being beneficial in any case, since we assume the spectrum is already perfectly represented by a sufficiently small F_s. In general, due to the free choice of B, the lowest frequency sampled in $\mathcal{F}_{\text{log}}$ may not coincide with the highest frequency sampled in $\mathcal{F}_{\text{lin}}$, which is not a problem. In addition, as a guideline, we can use F_s as a lower bound (equivalent to a maximum resolution) for the descending sampling procedure in $\mathcal{F}_{\text{log}}$; as a consequence, linear sampling can end up being extended beyond f_c. In fact, the strategy proposed in this paper could have been implemented differently, e.g. with the user setting only F_s and B, and allowing for the transition to happen when the resolutions meet; nevertheless, we found it useful to allow the user to set f_c, guaranteeing that a desired frequency band is simulated in predefined fixed frequency steps, independently from the choice of B. This will be especially important for the phase estimation, as will be discussed later in this section.

An example of the proposed sampling approach can be seen in Figure 1, which illustrates an HRTF simulated with the original regular sampling ($F_s = 150$ Hz) compared with the proposed hybrid sampling ($f_c = 2.76$ kHz, $F_s = 150$ Hz, and $B = 9$ bins/octave). Since the distance between samples after f_c is progressively higher, at some point, the log resolution seems to be insufficient to properly describe the original curve, starting to produce aliasing. This can be clearly verified at very high frequencies (around 18 kHz) where peaks and valleys occur between the non-linearly spaced samples. The question is how much these spectral details matter *perceptually*. This will be explored in more detail later.

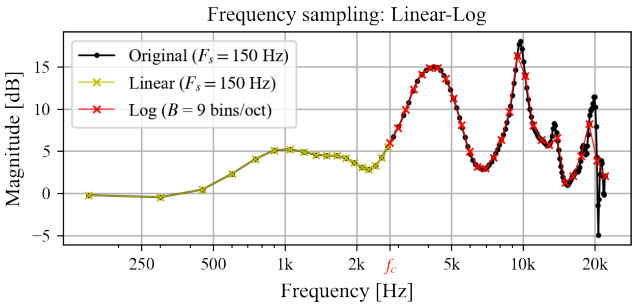


Figure 1: *Example of an HRTF simulated with the original linear sampling compared with the hybrid sampling proposed.*

Figure 2 illustrates an example of the computing time spent for each frequency bin in a real simulation using: the original regular frequency distribution ($F_s = 150$ Hz); and the proposed approach with $f_c = 5.51$ kHz, $F_s = 150$ Hz, and two different resolutions, for comparison ($B = 6$ and $B = 12$ bins/octave).

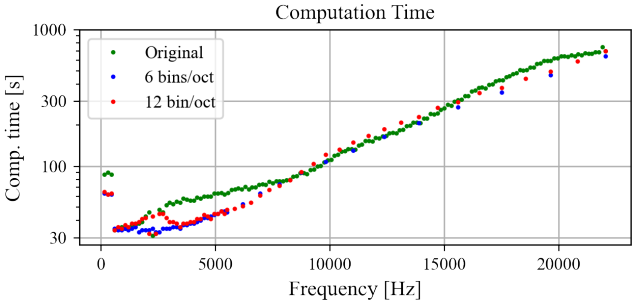


Figure 2: *Computing time varying with frequency. Original linear sampling (steps of* 150 *Hz), compared to the hybrid sampling approach (*6 *and* 12 *bins/oct).*

As can be seen in this example, which depicts a single simulation, there exists some fluctuation in processing time, even within the same frequency bins. This is expected, since computers perform several tasks in parallel, including calculating several different frequency bins sharing the same computational resources. Nevertheless, the same tendency of increasingly high computational cost with frequency is observed, regardless of the approach followed. However, since the proposed frequency distributions contain sparser bins at high frequencies, a reduction in the total computation time is expected.

To better compare the computing time for those simulations, it is worth plotting its accumulated value against frequency, as done in Figure 3, using the original linear sampling approach as a reference, with its total cost accounting for 100% of the time spent. The tendency of the accumulated computing time to grow exponentially (linearly in the logarithmic scale) towards high frequencies is evident; it also becomes clear that the resolution B used in the upper-frequency band changes the slope of the linear growth in the log scale. In this example, the HRTFs computed using the hybrid approach with $B = 6$ and $B = 12$ bins/octave cost, in total, around 12% and 20% of the linear approach, respectively.

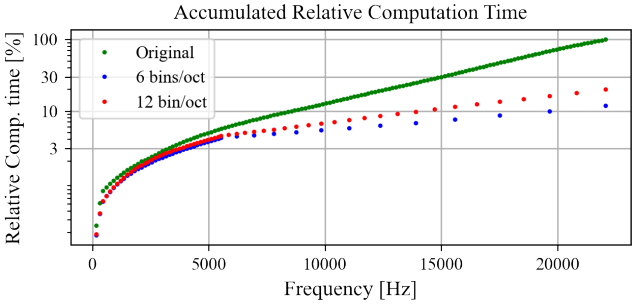


Figure 3: *Relative accumulated computing time varying with frequency. Original linear sampling ($F_s = 150$ Hz), compared to the hybrid sampling approach (6 and 12 bins/oct).*

It stands to logic that, on average, there should be an upper bound and a lower bound for the total computing time: it should never be higher than the reference, regardless of the choice of B and f_c, since F_s determines the maximum resolution possible, and F_s is also used in the reference; and it should also never be lower than the accumulated computation time at frequency f_c.

2.2.1. *Converting irregularly- to regularly-sampled HRTFs*

Certainly, some practical problems arise from using an arbitrary non-linear frequency resolution. It is required for an HRTF to have a complex set of samples in frequency evenly distributed throughout the whole frequency spectrum, allowing for the desired filtering procedure in which the HRTFs are intended to be used.

To this end, a simple linear interpolation procedure applied to the magnitude of the simulated spectrum $||\hat{\mathrm{H}}(\mathbf{x}, \hat{f}, s)||$ provides the regular frequency scale required, producing the magnitude spectrum $||\mathrm{H'}(\mathbf{x}, f, s)||$. Since the HRTFs are used in the digital domain, they will be henceforth denoted $\mathrm{H'}[\mathbf{x}, k, s]$, where $k \in \mathcal{K} \triangleq \{0, 1, 2, ..., K-1\}$ is the frequency index in the discrete frequency domain. Note that this interpolation is not able to restore or estimate parts of the spectra where information has been lost due to undersampling.[3] More sophisticated interpolation schemes might be explored in the future.

While the linear interpolation of magnitudes predicts reliable new samples, the periodicity of the phase makes estimating it between progressively more spaced samples a non-trivial problem. After trying different approaches, e.g. iteratively estimate the phase and correct its value using the unwrap procedure[4] on a frequency-bin basis, the best results were achieved by using the average group delay $d(\mathrm{H'})$ below f_c to linearly extrapolate the unwrapped phase. The average group delay is defined by

$$d(\mathrm{H'}) = \frac{1}{k_c} \sum_{0 \le k < k_c} \angle \mathrm{H'}[k+1] - \angle \mathrm{H'}[k], \tag{6}$$

where $k_c = \lfloor f_c / F_s \rfloor$ ($\lfloor . \rfloor$ denoting the floor function) is the index of the digital frequency related to f_c, and variables $\mathbf{x}$ and s have been omitted to simplify the notation.

Figure 4, illustrates the unwrapped phase of the same HRTFs shown in Figure 1. It can be observed that the higher the f_c, the lower the deviation in phase. However, this mismatch at high frequencies should have very little, if any, perceptual impact on the performance of the resulting HRTFs, especially since the phase at high frequencies is coherent with the average-group delay of the lower end of the spectrum.

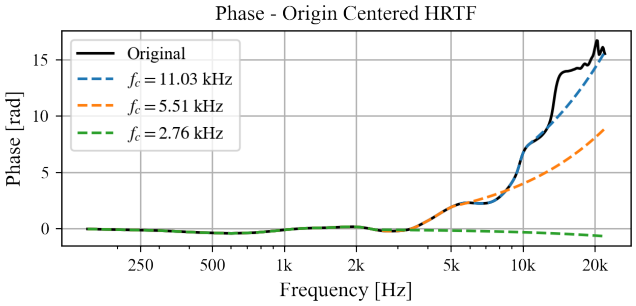


Figure 4: *Unwrapped phase of an HRTF: original regular sampling approach ($F_c = 150$ Hz); and the proposed hybrid approach ($F_c = 150$ Hz, $B = 9$ bins/octave, and f_c set to $11.03, 5.51, 2.76$ kHz).*

Since the pressure estimated at the ear canals is normalized by the pressure at the origin p_0, as mentioned in the previous section, the phase can assume negative and positive values. When it comes to generating HRIRs from the complex pressure simulated, a common procedure is to add a delay in such a way that the resulting filters become causal.

HRIRs of the same subject of the HRTF shown in Figure 1 can be seen in Figure 5, for different directions in the horizontal plane and different values of f_c; it was added a delay relative to the distance of 60 cm at speed-of-sound $c = 334$ m/s, i.e. approximately 1.8 ms. It can be observed that the impulse responses exhibit gradually higher energy concentration as f_c decreases, due to the increased frequency range whose phase gets linearized. As for the HRTF, an example of the spectra in the median plane is

[3] i.e., when the frequency spacing between the samples is below the Nyquist sampling frequency required for a lossless description of the sampled signal.

[4] Phase unwrapping algorithms aim to recover the true unwrapped phase signal by identifying and correcting phase discontinuities that occur in a phase signal wrapped between 0 and 2π.

shown in Figure 6 for the original simulation and an approximation using our proposal ($F_c = 150$ Hz, $B = 12$ bins/octave, and $f_c = 5.51$ kHz). Especially within the frequency range relevant for spatial cues (3-15 kHz), spectral detail is very well preserved.

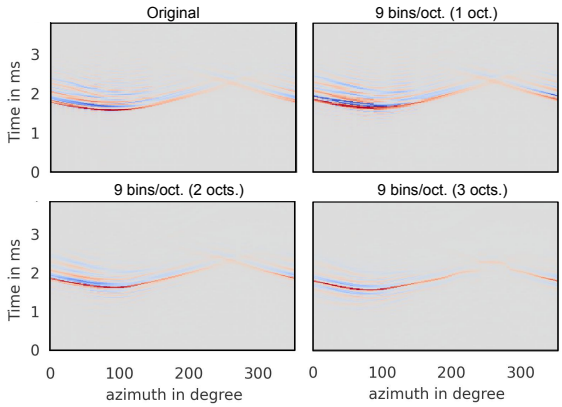


Figure 5: *Example of HRIRs of a subject (left ear) throughout the horizontal plane: original simulation ($F_c = 150$ Hz) and three examples of the proposed method ($F_c = 150$ Hz, $B = 9$ bins/octave, and $f_c = \{11.03, 5.51, 2.76\}$ kHz), respectively. Positive values are represented in blue, and negative values, in red.*

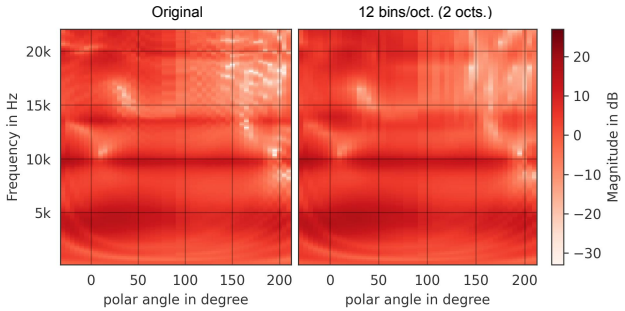


Figure 6: *HRTFs (left ear) throughout the median plane simulated with the original approach ($F_c = 150$ Hz) and with the proposed method ($F_c = 150$ Hz, $B = 12$ bins/octave, and $f_c = 5.51$ kHz).*

One thing to consider though is that, at high frequencies, wavelengths are comparable to the dimensions of the human ear, thus the incoming waves interact with the pinna in such a way as to produce steep peaks and notches that might be relevant to some degree (primarily up to 16 kHz [13, 2]) to produce spectral cues. For instance, the perception of elevation is highly dependent on those spectral cues, since the sound hits both ears virtually with no difference in time. Such characteristics of HRTFs help define important subject-dependent features that must be represented with some accuracy in the HRTFs to yield realistic results; a poor frequency resolution might underrepresent these spectral details.

Several experiments related to spectral smoothing have been conducted to determine to what degree HRTFs can be simplified before this can be noticed by listeners [14, 11, 15]. Thus, spectral smoothing might provide some indirect information with respect to the perceptual domain. For instance, using a fractional-octave smoothing window in the log spectrum can provide a realistic expectation of how much detail can be perceived in terms of spectral distortion, since the resolution of human hearing tends to follow an approximately logarithmic distribution. In Figure 7, one can see the same spectra illustrated in Figure 1, estimated for three different f_c, and smoothed using a third-octave rectangular window. As the resulting magnitudes are very similar, it is expected that the HRTFs sound very similar, if not indistinguishable.

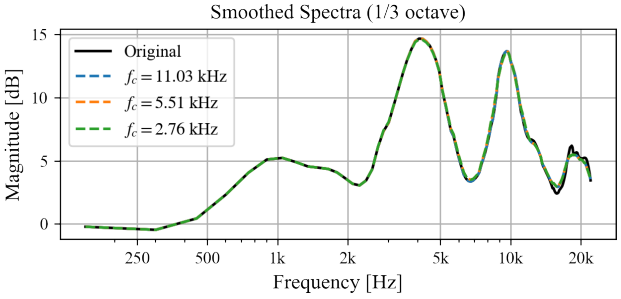


Figure 7: *Magnitude spectra of an HRTF smoothed using a third-octave filter: original regular sampling approach ($F_c = 150$ Hz); and the proposed hybrid approach ($F_c = 150$ Hz, $B = 9$ bins/octave, and $f_c = \{11.03, 5.51, 2.76\}$ kHz).*

Since spectral smoothing is a common post-processing procedure to remove measurement noise in HRTFs, this could also be included in the proposed approach, thus providing spectral curves that would smoothly follow the original simulated HRTFs. For example, a Hamming window with constant selectivity, or quality Q, could be used as a smoothing filter in the frequency domain, providing a smoothed curve more faithful to the original spectra the higher the resolution B compared to Q. In addition, the smoothing procedure would ensure smoother transitions between HRTFs of different directions, which is relevant whenever sound sources are not static.

In the case of the proposed approach, smoothing can be interpreted as a statistical estimation of the local energy in each frequency region, whose width varies geometrically in frequency. Such an estimate will be more precise the higher the resolution B.

3. EXPERIMENTS

The experiments performed in this paper consist of an acoustic analysis conducted on real 3D meshes acquired from 23 volunteers, over which different combinations of the parameters available for the simulations (the resolution parameters F_s and B, and the crossover point f_c) are tested. The spectral distortion and the total computing time are then presented for each case.

3.1. Spectral Difference Error and Computation Time Assessment

In order to compare the different versions of HRTFs, the Spectral Difference Error (SDE) is used. It can be computed for a specific frequency bin k by averaging the results over the D source positions $\mathbf{x}_d$ and the S subjects s as

$$\mathrm{SDE}[k] = \sqrt{\frac{1}{DS}\sum_{d=1}^{D}\sum_{s=1}^{S}\left(20\log_{10}\frac{||\mathrm{H'}[\mathbf{x}_d,k,s]||}{||\mathrm{H}[\mathbf{x}_d,k,s]||}\right)^2}, \quad (7)$$

where $||.||$ denotes the magnitude of the complex values. In turn, the overall SDE can be computed as

$$\text{SDE} = \sqrt{\frac{1}{DKS}\sum_{d=1}^{D}\sum_{k=0}^{K-1}\sum_{s=1}^{S}\left(20\log_{10}\frac{||\text{H'}[\mathbf{x}_d,k,s]||}{||\text{H}[\mathbf{x}_d,k,s]||}\right)^2}. \tag{8}$$

To produce the different simulations varying the available parameters, instead of simulating $\hat{\text{H}}$ for every single case, a preliminary study showed that estimating $\hat{\text{H}}$ from H via interpolations would be sufficiently accurate. Similarly, for the analysis of the computing time, a reference average time was calculated for each frequency bin using all available original simulations, with $F_\text{s} = 150\,\text{Hz}$. Then, estimating the cost in $\hat{f} \in \hat{\mathcal{F}}$ via interpolation over the average times in $f \in \mathcal{F}$ of the original simulations resulted in the percentage of the new average computing time w.r.t the reference, which allowed us to avoid possible fluctuations in computing time between different versions of the same HRTFs.

3.2. Participants

In total, 23 volunteers were recruited at the University of Osnabrück and compensated with 15 € after participation. They were, on average, *M* = 25.61 years old (*SD*= 4.77); 43.5% were female, and 56.5% were male; and all of them reported having normal hearing.

3.3. Ethical Approval

The experiments were all performed using 3D scans of voluntaries in accordance with the Declaration of Helsinki, with ethical approval obtained from Osnabrück University Ethics Committee (AZ.: 4/71043.5). In the process, the anonymity of participants and the confidentiality of their data were ensured. Participants were informed about the objectives and the procedure of the study as well as about their right to withdraw from the study at any time without adducing reasons or experiencing any negative consequences. All participants provided informed consent before participation in the study, which will also include future listening tests.

3.4. Acquisition and Preparation of the 3D Meshes

A 3D scan of the head and torso of each participant was obtained with the POP 3D scanner from Revopoint[5] with a resolution of 0.3 mm. The same procedure conducted in [16] was followed: The test subjects were asked to wear a nylon hair net to facilitate the scanning procedure since the hair is not modeled; then, point clouds were created and converted into 3D meshes; the scanning procedure lasted about 20 min and the results were verified visually to ensure the ears were captured without any artifacts and no major problems happened in the overall shape of the head and torso. Small artifacts caused by hair and clothes were neglected in this initial phase and dealt with later.

The meshes were then carefully treated for the numerical calculations in Blender.[6] First, artifacts and details related to hair and clothes in the meshes were smoothed out using editing tools in Blender. Such regions were transformed into flatter/smoother surfaces, contributing to saving computational resources by lowering the number of points in the mesh. The automatic mesh-grading procedure described in [9] was then used with the objective of optimizing the meshes in terms of computational burden. This procedure consists of assigning different resolutions to different regions of the mesh, based on both the degree of curvature and the distance from the ear canal. To determine the minimum and the maximum distances between points in the mesh, different values were tried, starting from 0.7 mm and 10 mm, respectively. The minimum distance was set to 1.25 mm, and the maximum to 18.75 mm, these being the largest distances that did not cause significant spectral distortion ($\text{SDE}[k] < 0.5\,\text{dB}$ below 16 kHz).

3.5. Simulation

After the pre-processing stage, numerical calculations were performed with the Mesh2HRTF [6, 7] library, an open-source code that simulates HRTFs and is integrated with Blender. The implementation of the proposed method was also entirely based on this library. The simulations were run for both ears using faces at the ear canal (which was occluded) as vibrating elements, with the frequency spectrum sampled every 150 Hz and within the range 0-22.05 kHz. The results were then sampled for 1550 spatial positions distributed on the surface of a sphere with a radius of 1.2 m centered on the participant's head.

In the experiments conducted, a relatively wide range of the adjustable variables was spanned. As before, the original simulations used the frequency resolution of $F_\text{s} = 150\,\text{Hz}$, which provides good quality sampling and can easily produce both 48 kHz and 44.1 kHz HRTF files. Simulations run to compare different values of F_s produced negligible differences (below 0.1 dB for the whole frequency spectrum) between the spectra. Although lowering F_s would make the proportional computation time saved dramatically higher (to our advantage), such a comparison would be unfair since a higher regular resolution is not necessary.

The values assigned to the parameters were then $F_\text{s} = 150\,\text{Hz}$, $B = \{1, 3, 6, 9, 12, 15, 18\}$ bins/octave; with f_c varying in octaves, related to $N_\text{octs} = \{1, 2, 3, 4, 5, 6\}$ octaves. As mentioned earlier, certain combinations of B and N_octs can cause the linear sampling to be extended beyond the crossover point to prevent the logarithmic resolution from exceeding the regular resolution defined by F_s. This will be clearly shown in the results reported in the next subsection.

3.6. Results

The results obtained with this variety of configurations are summarized in Table 1, in which the computation times are shown, and in Table 2, where the SDEs are presented.[7]

As can be observed in Table 1, the proposed hybrid-resolution approach is capable of saving large amounts of processing time. For instance, using $B = 6$ and only two octaves ($f_\text{c} = 5.51\,\text{kHz}$) logarithmically sampled, the total computational cost is expected to be, on average, only 12.9% of the cost of the original version.

Figure 8 illustrates the curves of the average accumulated computing times for the different versions of the proposed method in relation to the original one, thus showing lower proportional values the greater the savings in computing time, over frequency. B varies within the range $\{1 - 18\}$ for $N_\text{octs} = 2$ octaves down from the maximum frequency 22.05 kHz and $N_\text{octs} = 6$ octaves, which effectively sets the minimum f_c. As mentioned above, setting a

[5] Available from `http://www.revopoint3d.com` (last viewed: March 17, 2023).

[6] Available from `http://www.blender.org` (last viewed: March 17, 2023).

[7] Not all results are presented in the table, as to avoid repeating the relative total computation time of configurations with extended linear sampling.

Table 1: *Relative total computing time (%) for simulations using different hybrid lin-log resolutions (F_s = 150 Hz, B bins/octave) and varying the crossover point frequency f_c in octaves descending from the maximum frequency available (22.05 kHz). For a given choice of f_c, repeated values are omitted. Low distortion (SDE< 1.5 dB @0-15 kHz.) results are presented in boldface.*

	Hybrid resolution: $F_s = 150$ Hz, B [bins/oct.]						
N_{octs}	1 b/o	3 b/o	6 b/o	9 b/o	12 b/o	15 b/o	18 b/o
1 oct.	15.3%	17.2%	**20.3%**	**23.4%**	**26.6%**	**29.7%**	**33.0%**
2 octs.	6.9%	9.2%	**12.9%**	**16.7%**	**20.5%**	**24.3%**	**28.2%**
3 octs.	4.9%	7.4%	**11.5%**	**15.6%**	**19.8%**	**23.9%**	**28.0%**
4 octs.	4.2%	6.9%	**11.2%**	**15.5%**	**19.8%**	-	-
5 octs.	3.9%	6.7%	**11.2%**	-	-	-	-
6 octs.	3.8%	-	-	-	-	-	-

crossover point determines the linear behavior of the curves, whose inclination is controlled by B. When using minimum f_c (Figure 8 right), though, since the more linear parts of the curves start in different parts of the spectrum, they tend to share a common inclination, although a less linear behavior can be observed, especially at low frequencies.

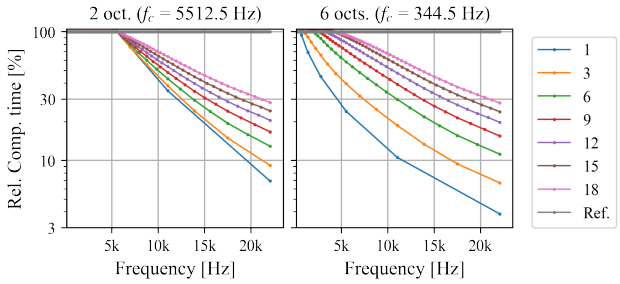


Figure 8: *Accumulated relative computing time for B within the range $1-18$ bins/octave, using $N_{octs} = 2$ octaves and $N_{octs} = 6$ octaves (minimum f_c) down from the maximum frequency 22.05 kHz.*

It is worth highlighting that, despite the great savings achieved with the proposed method, the growth rates of the computing time in frequency are still geometric (as illustrated in Figure 3), meaning that high frequencies still cost relatively much more than low frequencies. Since the average curve of the regular sampling approach tends to grow with a higher inclination, the relative saving rates are specific to the maximum frequency of the spectrum to be simulated. Naturally, the lower the maximum frequency, the closer the relative costs get, since the logarithmic resolution range becomes reduced and does not reach those regions of the spectrum where the frequency bins are more widely spaced. By way of comparison: if HRTFs were only simulated up to 16kHz, the full-range relative cost of 12.9% mentioned above would grow to $\approx 20\%$ when using $N_{\text{octs}} = 2$ and $B = 6$ (see Figure 8).

Table 2 shows, for each choice of B, the SDE for the minimum f_c possible, which provides the minimum computational burden, but also the maximum spectral distortion. For the previous example using $B = 6$, the maximum SDE calculated was 1.7 dB, spending 11.2% of the original computing time. Comparing the smoothed versions of the original and the proposed HRTFs using a third-octave smoothing window, the differences are even lower, as expected. For this case, the SDE_s (the 's' subscript denoting the smoothing procedure) was lowered to only 0.9 dB. For high values of B, such as 18 bins/octave, the SDE_s was as low as 0.3 dB, spending 28.0% of the original computing time.

Table 2: *Spectral difference error (SDE), in dB, for simulations using different hybrid lin-log resolutions (F_s = 150 Hz, B bins/octave). Values reported for the interpolated HRTFs (SDE) and their smoothed versions (SDE_s). The minimum f_c was used, as to provide the maximum SDE values.*

B [bins/octave]	1	3	6	9	12	15	18
SDE [dB]	4.2	2.5	1.7	1.4	1.2	1.0	0.9
SDE_s [dB]	3.5	1.6	0.9	0.6	0.5	0.4	0.3

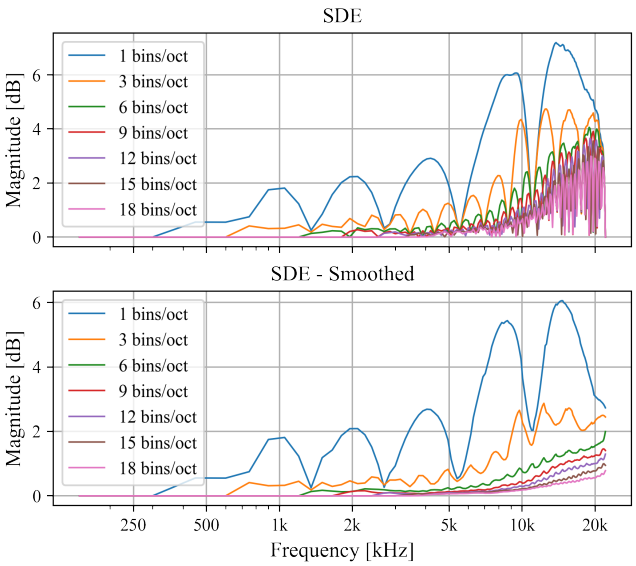


Figure 9: *SDE$[k]$ and $\text{SDE}_s[k]$ (spectra smoothed with a third-octave filter) for different resolutions.*

In order to analyze the spectral differences in more detail, SDE$[k]$ illustrates the distortion throughout the frequency spectrum, as can be seen in Figure 9 (top), where once again the minimum f_c is used to better illustrate the distortion occurred in the whole frequency spectrum. The distortion produced is primarily concentrated at high frequencies, as expected, because the sampling resolution is lowered with frequency. Even without considering the perceptual dimension, the distortion observed was considerably low for high values of B, peaking at around 3 dB at the very top of the frequency spectrum.

As can be expected, increasing the resolution B provides ripples with lower peaks in the SDE curves, and thus more transparent results. This can be seen in Figure 9 (bottom), where the SDE_s is shown for each configuration. Considering these curves, using $B > 6$ bins/octave creates SDEs below 1.5 dB, which we anticipate will provide perceptual transparency for all curves at least in the frequency range below 15 kHz. Some preliminary listening tests have shown that the just noticeable difference (JND) might lay between using $B = 3$ and $B = 6$ bins/octave. More rigorous listening tests using signal detection theory are planned to be carried out in the

future to assess the transparency of the HRTFs simulated with our method.

4. CONCLUSIONS

This paper presented a novel hybrid linear-logarithmic resolution approach for the numerical approximation of HRTFs. The proposed method requires remarkably less processing time than the regular sampling resolution usually employed. Experiments using 3D scans of 23 volunteers were conducted to compare computing time and spectral distortion for a wide range of resolutions. Preliminary experiments suggest that perceptual transparency may be achieved whilst saving around 89% of the processing time traditionally required. This solution is presented as a step towards everyday applications, lowering the computational power required for numerical simulations of HRTFs.

Since the original numerical simulations also present some spectral distortion in comparison to acoustic measurements, one concern to consider is the overall validity of the HRTFs obtained via the proposed numerical approximation, which further distorts the HRTFs to some degree. In future work, a subjective assessment will be conducted with the same volunteers from whom the 3D models were made to determine whether the subjects can notice a difference between simulations using their individualized HRTFs simulated with the traditional regular frequency resolution and using the proposed approach. This will involve different acoustic scenarios, sound sources with various characteristics, and dynamic changes in sound source position, thus covering many potential uses of the HRTFs. In addition, a more in-depth analysis will be carried out to determine the impact of the proposed procedure on localization accuracy and the spatial distribution of the spectral difference. Besides, it will be explored the use of upsampling procedures that could correct or restore spectral detail lost when simulating HRTFs using low spectral resolution.

5. ACKNOWLEDGMENTS

This work is funded by the Volkswagen Foundation (VolkswagenStiftung) Germany (grant no. 96 881).

6. REFERENCES

[1] Corentin Guezenoc and Renaud Seguier, "HRTF individualization: A survey," in *Proceedings of the 145th Audio Engineering Society International Convention*, New York, USA, October 2018.

[2] K. Pollack, W. Kreuzer, and P. Majdak, "Modern acquisition of personalised head-related transfer functions – an overview," in *Advances in Fundamental and Applied Research on Spatial Audio*, Brian Katz and Piotr Majdak, Eds. IntechOpen, London, United Kingdom, 1 edition, January 2022.

[3] Kazuhiro Iida, *Head-Related Transfer Function and Acoustic Virtual Reality*, Springer, 2019.

[4] Michele Geronazzo and Stefania Serafin, *Sonic Interactions in Virtual Environments*, Springer Nature Switzerland AG, Cham, Switzerland, 1 edition, 2023.

[5] S. Li and J. Peissig, "Measurement of head-related transfer functions: A review," *Applied Sciences*, vol. 10, no. 2, pp. 5014, July 2020.

[6] H. Ziegelwanger, W. Kreuzer, and P. Majdak, "Mesh2hrtf: Open-source software package for the numerical calculation of head-related transfer functions," in *Proceedings of the 22nd International Congress on Sound and Vibration*, Florence, Italy, July 2015.

[7] H. Ziegelwanger, P. Majdak, and W. Kreuzer, "Numerical calculation of listener-specific head-related transfer functions and sound localization: Microphone model and mesh discretization," *The Journal of the Acoustical Society of America*, vol. 138, no. 1, pp. 208–222, July 2015.

[8] Brian F. G. Katz, "Boundary element method calculation of individual head-related transfer function. i. rigid model calculation," *The Journal of the Acoustical Society of America*, vol. 110, no. 2449, pp. 2440–2448, October 2001.

[9] T. Palm, S. Koch, F. Brinkmann, and M. Alexa, "Curvature-adaptive mesh grading for numerical approximation of head-related transfer functions," in *In Fortschritte der Akustik - DAGA 2021 : 47. Jahrestagung für Akustik*, Vienna, Austria, August 2021.

[10] Marina Bosi and Richard E. Goldberg, *Introduction to Digital Audio Coding and Standards*, vol. 721, Springer Science & Business Media, 2002.

[11] Laurence J. Hobden and Anthony I. Tew, "Investigating head-related transfer function smoothing using a sagittal-plane localization model," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2015.

[12] Brian F. G. Katz, "Boundary element method calculation of individual head-related transfer function. ii. impedance effects and comparisons to real measurements," *The Journal of the Acoustical Society of America*, vol. 110, no. 2449, pp. 2449–2455, October 2001.

[13] J. Hebrank and D. Wright, "Spectral cues used in the localization of sound sources on the median plane," *The Journal of the Acoustical Society of America*, vol. 56, no. 6, pp. 1829–1834, December 1974.

[14] Areti Andreopoulou and Brian F. G. Katz, "Comparing the effect of HRTF processing techniques on perceptual quality ratings," in *Proceedings of the 144th Convention of the Audio Engineering Society*, Milan, Italy, may 2018.

[15] Eugen Rasumowa, Matthias Blau, and Martin Hansen, "Smoothing individual head-related transfer functions in the frequency and spatial domains," *The Journal of the Acoustical Society of America*, vol. 135, no. 4, pp. 2012–2025, May 2014.

[16] M. Oehler, da Costa, M. V. M., M. Regener, and T. M. Voong, "Relevance of individual numerically simulated head-related transfer functions for different scenarios in virtual environments," in *Proceedings of the International Conference on Audio for Virtual and Augmented Reality 2022*, Redmond, USA, August 2022.

PYWDF: AN OPEN SOURCE LIBRARY FOR PROTOTYPING AND SIMULATING WAVE DIGITAL FILTER CIRCUITS IN PYTHON

Gustav Anthon, Xavier Lizarraga-Seijas and Frederic Font

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
anthon.gus1@gmail.com | xavier.lizarraga@upf.edu | frederic.font@upf.edu

ABSTRACT

This paper introduces a new open-source Python library for the modeling and simulation of wave digital filter (WDF) circuits. The library, called `pwydf`, allows users to easily create and analyze WDF circuit models in a high-level, object-oriented manner. The library includes a variety of built-in components, such as voltage sources, capacitors, diodes etc., as well as the ability to create custom components and circuits. Additionally, `pywdf` includes a variety of analysis tools, such as frequency response and transient analysis, to aid in the design and optimization of WDF circuits. We demonstrate the library's efficacy in replicating the nonlinear behavior of an analog diode clipper circuit, and in creating an all-pass filter that cannot be realized in the analog world. The library is well-documented and includes several examples to help users get started. Overall, `pywdf` is a powerful tool for anyone working with WDF circuits, and we hope it can be of great use to researchers and engineers in the field.

1. INTRODUCTION

Wave digital filters were initially developed by Alfred Fettweis in the '70s and '80s in order to digitize ladder and lattice circuits [1–3]. They have gained popularity in recent years as interest has grown in virtual analog (VA) modeling of audio and music applications [4,5]. Many analog audio effect circuits are exceedingly rare and/or expensive for the majority of music makers, so making these effects more accessible by faithfully recreating them in the digital domain has become an important goal of audio engineers and developers. [6, 7].

Wave digital modeling is a form of white box VA modeling that takes into account the entirety of a circuit's internal structure. A wave digital model of a circuit is composed by replicating each of the circuit's elements one by one, and connecting them with "adaptors", which inform what type of topology is configured [8]. Series and parallel adaptors of course connect elements in series and parallel and are the most common wave digital adaptors, while polarity inverters can be considered two-port adaptors, and R-type adaptors are used for more complicated topologies [9].

The elements and adaptors are arranged in an *SPQR* tree, with one element at the root and its children elements organized below it [10, 11]. This is done by representing the reference circuit as a graph, in which nodes are circuit nodes and edges are circuit ports. Then a graph decomposition algorithm is performed to yield the SPQR tree. Waves propagate throughout the tree from one wave digital element to the next to simulate the analog circuit.

Circuit elements are discretized locally and are therefore very modular as compared to traditional techniques of physical modeling entire circuits. This allows users to swap out components or change parameters without the need to recompute the entire system's transfer function. This also enables users to reuse element models in multiple circuits, needing only to change parameter values, or occasionally, methods of discretization for stateful components. WDFs work not with Kirchoff variables like voltage and current, but rather wave variables, namely 'incident' and 'reflected' waves at each circuit element's port. All WDF elements accept an incident wave, and propagate a reflected wave as their output[1]. The main work of deriving wave digital models of circuit elements involves computing the reflected wave based on the incoming incident wave.

In this paper we introduce a new open-source Python library for modeling and simulating WDF circuits called `pywdf`. Section 2 provides an overview of related work. Section 3 details the structure of the library and describes basic functionalities. Section 4 offers examples of circuits built with this library.

2. RELATED WORK

Several libraries for implementing wave digital filters tailored to audio circuits exist, notably including: `Faust` framework `wdmodels` [12], and `C++` libraries `chowdsp_wdf` [13] and `RT-WDF` [14]. These libraries are implemented such that the circuits can be reliably run and tested in real time. This however necessitates that a low level language, such as `C++`, or very specific knowledge about `Faust` programming is used to implement the models. The learning curve of `C++` is significantly steeper than that of higher level and less performant languages, which makes the barrier of entry quite high for researchers and engineers to begin experimenting with wave digital filters. This is the central motivation for developing this library; there is not currently a library for modeling wave digital filters in Python, where the process of prototyping and programming is much less difficult. As real time processing of audio in Python results in higher latency, the library is best suited for prototyping, though real time is still possible thanks to frameworks like `pyaudio`[2].

Python is also a programming language that is typically

[1] Some wave digital adaptors accept multiple incident waves and propagate multiple reflected waves, such as R-type adaptors

[2] `https://people.csail.mit.edu/hubert/pyaudio/`

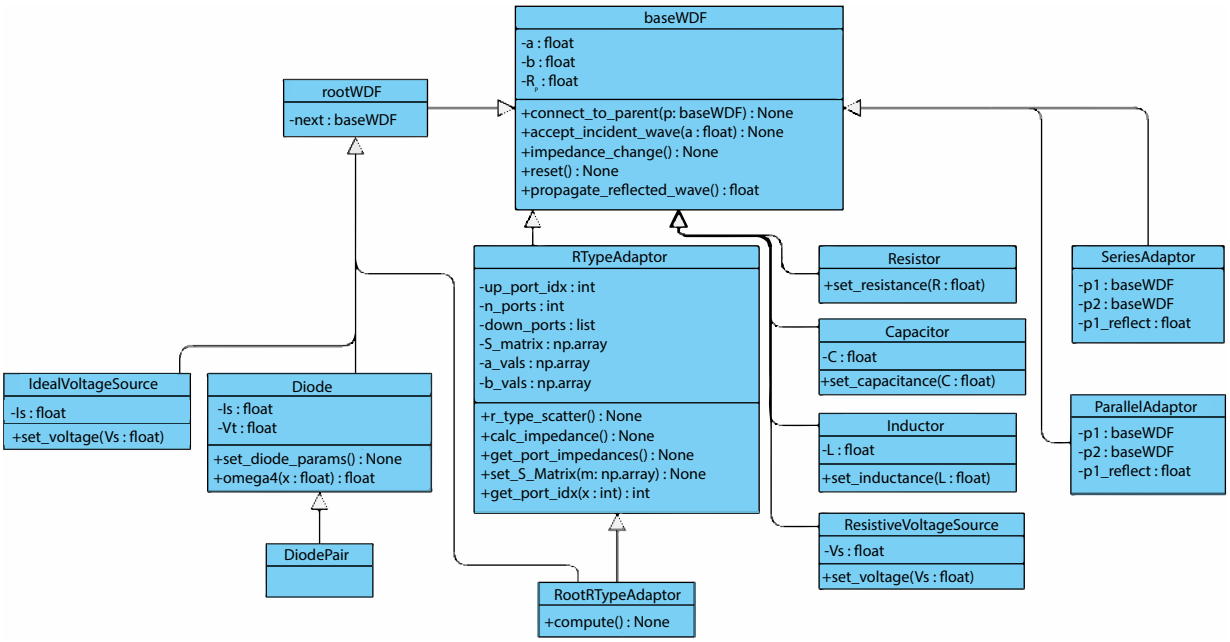


Figure 1: `pywdf` circuit elements UML class diagram

used for research and prototyping as frameworks like `numpy`[3], `scipy`[4], and `matplotlib` [5] allow easy access for plotting and visualizing the behavior of a system. This process is significantly easier than building and rendering a `C++` plugin, opening a DAW session, and configuring the correct channel strip settings to ascertain the same information about the system. Python has also become one of the more popular environments for experimenting with Machine Learning (ML). Recent works such as [15, 16] have united the world of wave digital filters and machine learning, so we believe a WDF library implemented in Python will further facilitate research and development at the intersection of these two subjects.

3. STRUCTURE

The `pywdf` library[6] is built following the object oriented paradigm used in the `C++ chowdsp_wdf` library[7]. The base class from which all wave digital elements and adaptors inherit basic functionalities is called `baseWDF`. This class initializes variables like incident and reflected waves, parent elements, and contains functions to probe and calculate the port resistance at each element and connect elements to one another according to the composition of the *SPQR*-tree representing the circuit. Basic wave digital elements in the repository include resistors, ideal and resistive voltage sources, capacitors and inductors, 3-port series and parallel adaptors and more. Also included is the the diode and diode pair, the nonlinear behavior of which we model using the reflected wave equation derived by Werner et al. in [17]:

$$b = a - 2\lambda V_T \left[\mathcal{W}\left(\frac{R_p I_s}{V_T} e^{\frac{\lambda a}{V_T}} \right) + \mathcal{W}\left(-\frac{R_p I_s}{V_T} e^{-\frac{\lambda a}{V_T}} \right) \right] \quad (1)$$

Where a is the diode pair's incident wave, λ is $\mathrm{signum}(a)$ (defined in equation 2), V_t is thermal voltage, $\mathcal{W}$ is the Lambert W function, R_p is port resistance, and I_s is the reverse bias saturation current.

$$\mathrm{signum}(x) = \begin{cases} -1 & , \quad x < 0 \\ 0 & , x = 0 \\ +1 & , \quad x > 0 \end{cases} \quad (2)$$

The thermal voltage is typically about 25.85 mV at room temperature, but depends on the number of antiparallel diodes used in series, giving us the ability to modify the number of diodes used in the circuit. While the reverse bias saturation current varies from diode to diode, we use an I_s value of 2.52e-9 which is standard for silicon diodes such as the common 1N4148. In practice, we employ the fast approximation of the Lambert W function published by D'Angelo et al. in [18].

Also included are *R*-type adaptors, which allow for implementations of circuits whose topologies cannot be broken down into strictly series or parallel, such as the bridged T circuit. Additionally, they can be used to implement models of operational amplifiers. We include *R*-type adaptors that are unadaptable and can only be used at the root of a connection tree, as well as adaptable ones that can be used more flexibly. The outputs of *R*-type adaptors are computed with scattering matrices, which can be found using methods from Modified Nodal Analysis [19]. The full structure of the library's circuit elements is depicted by the Unified Modeling Language (UML) class diagram in figure 1, and shows how and from where each WDF element inherits its variables and methods.

There also exists a Github repository with a Python script to generate a scattering matrix for a `chowdsp_wdf` circuit given a

[3] `https://numpy.org/`
[4] `https://scipy.org/`
[5] `https://matplotlib.org/`
[6] `https://github.com/gusanthon/pywdf`
[7] `https://github.com/Chowdhury-DSP/chowdsp_wdf`

circuit's netlist [8]. This repository also contains a fork that allows users to generate a scattering matrix[9] compatible with `pywdf` circuits.

Lastly we have created a `Circuit` class from which any wave digital circuit built using this library can inherit basic functionalities. These functions are useful for research and analysis and include:

- `process_sample()` : a function that contains a generic method to process a single sample of a wave digital circuit
- `process_signal()` : uses `process_sample` to process entire signals with a circuit
- `get_impulse_response()` : uses `process_signal()` to process a Dirac delta function and returns the output
- `plot_freqz()` : uses `get_impulse_response()` and takes the Fast Fourier Transform (FFT) to plot the system's magnitude and phase responses
- `plot_freqz_list()` : allows user to visualize how the system's frequency response changes as a parameter is varied. Figure 9 was generated with this function

4. EXAMPLES

In this section we will describe some of the examples offered by the library such as the Diode Clipper and a Passive All Pass Filter. Although the library includes additional circuits such as the RCA Mark II Sound Effects Filter [6] and the Bassman Tone Stack [20, 21], among others.

4.1. Diode Clipper Evaluation

This library was initially developed to thoroughly examine how effectively wave digital filters can replicate the nonlinear behavior of an analog diode clipper [22] . We do so by examining frequency response comparisons, AC transient analysis, and harmonic series analysis. The Diode Clipper WDF model was constructed by converting the circuit to an *SPQR* tree as shown in 3. We then instantiate these components in a `DiodeClipper` class `__init__` function, as shown in listing 1. The parameters used to instantiate these components such as the resistance and capacitance are calculated according to the cutoff value provided by the user.

Listing 1: Instantiating WDF elements of diode clipper

```
self.R1 = Resistor(self.R)
self.Vs = ResistiveVoltageSource()

self.S1 = SeriesAdaptor(self.Vs, self.R1)
self.C1 = Capacitor(self.C, self.fs)

self.P1 = ParallelAdaptor(self.S1, self.C1)
self.Dp = DiodePair(self.P1, 2.52e-9,
    n_diodes=n_diodes)
```

[8] https://github.com/jatinchowdhury18/R-Solver
[9] https://github.com/gusanthon/R-Solver

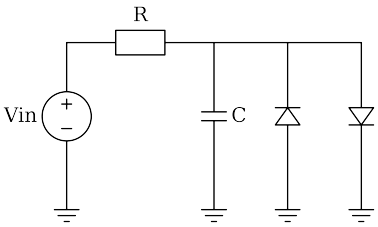


Figure 2: Diode clipper circuit.

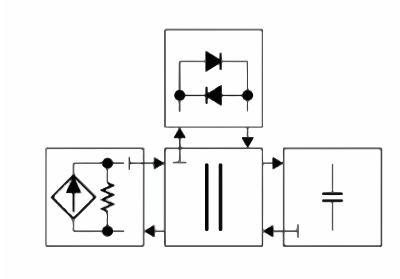


Figure 3: Diode clipper *SPQR* tree.

4.1.1. Frequency Response Comparison

We compare the magnitude and phase responses generated by the `pywdf` circuit model to those of a SPICE[10] circuit model, with the frequency cutoff at the following values:

$$F_c = \{70, 150, 250, 500, 1000, 2000, 4000, 8000, 16000\}[\text{Hz.}] \tag{3}$$

At each of the following sample rates:

$$F_s = \{44100, 48000, 88200, 96000\}[\text{Hz.}] \tag{4}$$

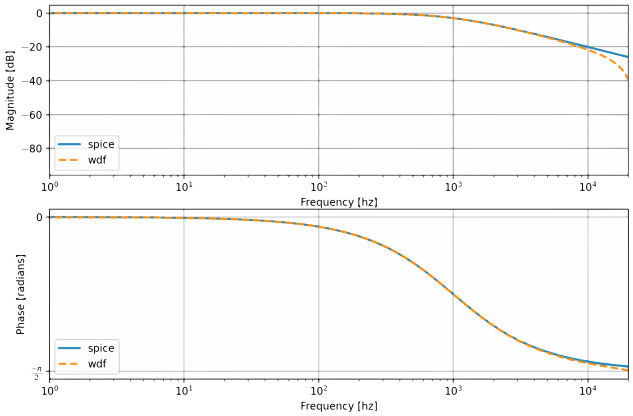


Figure 4: Spice vs `pywdf` frequency response, with 44.1kHz sample rate and cutoff frequency at 1kHz

[10] http://bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE/

Figure 4 shows this frequency response comparison with a sample rate of 44.1 kHz and a cutoff parameter of 1 kHz. We observe only slight deviations as the WDF model's frequency approaches Nyquist - where its behavior is technically undefined. We also compute error metrics between the two models using Mean Square Error (MSE)[11] and Error-to-Signal ratio (ESR).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{5}$$

$$ESR = \frac{\sum_{n=-\infty}^{\infty}|y_p[n] - \hat{y}_p[n]|^2}{\sum_{n=-\infty}^{\infty}|y_p[n]|^2} \tag{6}$$

The MSE and ESR results averaged across all parameter changes for each sample rate are listed in table 1.

	Magnitude [dB]		Phase [rad]	
Sample Rate [Hz]	MSE	ESR	MSE	ESR
44100	7.215	1.703	0.001	0.019
48000	6.837	1.324	0.015	0.015
88200	7.035	0.416	0.003	0.003
96000	6.660	0.344	0.003	0.003

Table 1: Averaged MSE and ESR of magnitude and phase across all parameter changes by sample rate

4.1.2. AC Transient Analysis

A diode clipper typically includes an input gain stage, to raise the level of the input signal and consequently cause it to be clipped even harder. We examine how sinusoidal inputs respond to raising the input gain parameter and show how the signal becomes a square wave in Figure 5. One can observe that even at negative input gain values the signal is being saturated, which implies that the diodes contribute nonlinearities even when the input signal is not crossing the clipping voltage. Table 2 shows the amount of total harmonic distortion and noise (THD+N) introduced at different input gain levels, which also indicates that negative and small input gain values result in additional saturation to the input.

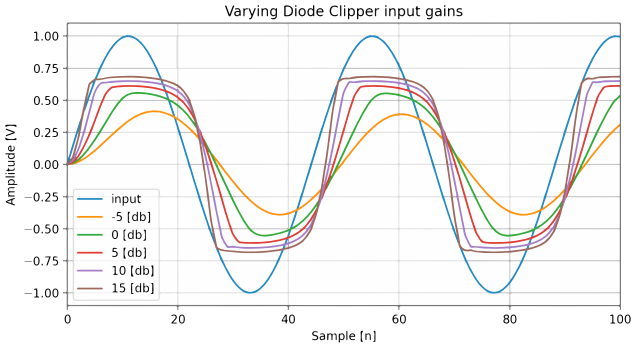


Figure 5: AC transient analysis varying input gains.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html

Input gain [dBFS]	THD+N [%]
-20	0.017
-15	0.017
-10	0.029
-5	0.437
0	6.897
5	15.519
10	21.859
15	26.425
20	29.611
25	31.978
30	33.559
35	35.055
40	37.115

Table 2: THD+N% introduced at each input gain value with 44.1 khz sample rate and cutoff frequency at 1 khz

4.1.3. Harmonic Series Analysis

We also perform a swept sine analysis as first described by Farina in [23], which involves feeding an exponentially swept sine wave across all frequencies below Nyquist into a nonlinear system, and convolving the output with an inverse filter of the input sweep to create a multidimensional impulse response (IR). Because it is a nonlinear system, the diode clipper adds additional frequencies (harmonics) to its input. The multidimensional IR represents the impulse response of each of the harmonics introduced by the system. The multidimensional IR of the diode clipper is shown in 6, with each vertical line corresponding to the IR of each harmonic. We can isolate each individual IR from the multidimensional IR and take its FFT to visualize each harmonic's magnitude response, as shown in figure 7.[12] It is interesting to note that the 0th harmonic has very low magnitude below 20 Hz, which is likely due to the bandwidth of the sweep tone being limited between 20 Hz and 20 kHz. It is also interesting to note that the bandwidth of each successive harmonic decreases, and is essentially high-passed further and further.

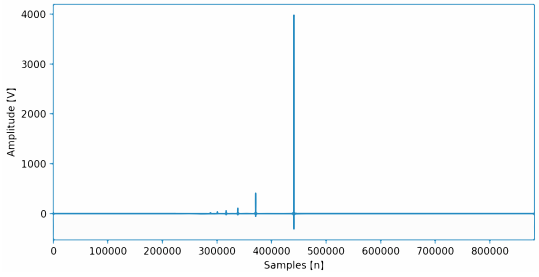


Figure 6: Diode Clipper multidimensional impulse response.

To assess the contribution of each harmonic in terms of the signal, we reused the idea of the Signal-to-Noise Ratio (SNR) in this scenario. It drove us to define the Harmonic-to-Signal Ratio (HSR), that allows us to compare the magnitude level of the 0th-harmonic (the desired signal) to each higher harmonic.

[12] Responses were normalized 0 dB

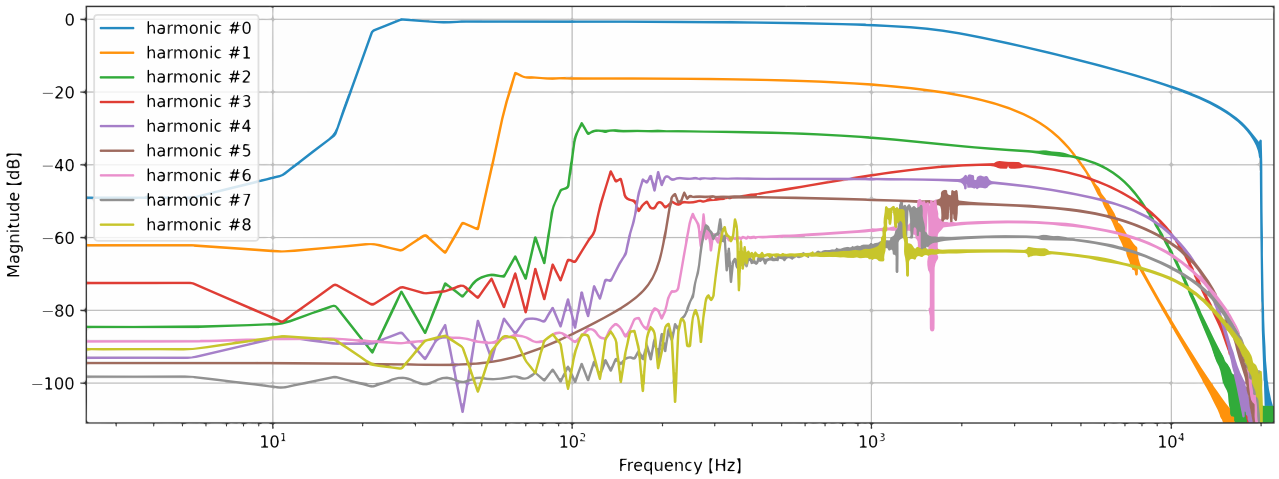


Figure 7: Magnitude response of each harmonic.

# Harmonic	HSR [dB]
0	0.0
1	34.63
2	60.51
3	73.69
4	81.27
5	91.58
6	101.77
7	108.91
8	115.14

Table 3: Harmonic-to-Signal Ratio (HSR) for the first 8 harmonics.

$$HSR_{dB_i} = 2(20\log_{10}(H_i) - (20\log_{10}(H_0))) \quad (7)$$

HSR is defined in Equation 7 [13]. The measurements of each harmonic's HSR can be seen in Table 3. The HSR of the 0th harmonic (fundamental) is of course 0 dB as it is being compared to itself. Each successive harmonic's HSR is greater than its previous, as the difference between the harmonic and the fundamental increases as the harmonic increases.

4.2. All Pass Filter

[24] describes a new resistor-capacitor (RC) circuit realization of a first-order all-pass filter (APF). The implementation is very particular because the APF scheme is composed of a single grounded capacitor, and three resistors, one of which is negative. Negative resistance means acceptance and it is of course not possible in the real world, but is an interesting experiment when simulating circuits digitally. This demonstrates that the digital domain allows us to modify the behavior of analog circuits, for example when each of the resistors are positive the circuit behaves as a high shelving filter.

APF is an important filter function for analog processing design because it provides phase shifting whereas the magnitude is constant at all frequencies, keeping the amplitude of the input constant over the frequency bandwidth of interest. APFs can be used to correct undesired phase change as a result of a processing signal such as the high-pass filtering in a loudspeaker crossover [25].

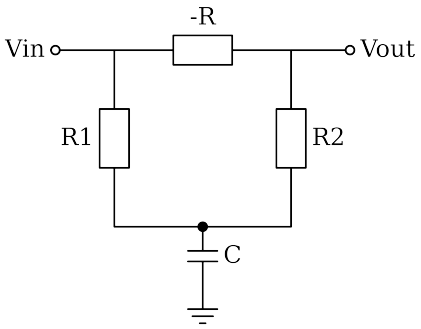


Figure 8: APF circuit.

This model was first implemented in SPICE and then we built the netlist of the R-type adaptor with the approach based on a graph decomposition of the reference circuit [10, 11]. Later we used R-Solver to compute the scattering matrix that allows us to calculate the impedance of the R-type adaptor, which was computed without an adapted port. Listing 2 shows the instantiation of each wave digital component in the APF class `__init__` function.

Figure 9 depicts the frequency response for a list of cutoffs, that was generated using the `plot_freqz_list()` command. We can observe how the magnitude is unaltered while the phase is more affected by the circuit with a 180° delay in the frequency components below the cutoff frequency. The phase of the first-order APF varies from 0 at $\omega = 0$ to -π at $\omega = \infty$ and the pole ω_p

[13] `https://wikimedia.org/api/rest_v1/media/math/render/svg/ed42497b9008934f5bcbab43fc64c4d815b142ee`

and zero ω_z frequencies are calculated as [24] indicates,

$$\omega_p = \omega_z = \frac{1}{CR} \tag{8}$$

where C is the capacitance and R the resistance value. Equation 8 enabled us to implement the `set_cutoff` method to modulate the cutoff in our APF-WDF, which may be useful to help fix unwanted phase shifting. It also can be used to implement a phasing/flanging effect for music production applications, which could be achieved by modulating the cutoff frequency with an `LCOscillator`, also included in this library.

Listing 2: Instantiating WDF elements of APF

```
# Port B
self.R1 = Resistor(self.R1_value)

# Port C
self.R2 = Resistor(self.R2_value)

# Port D
self.R3 = Resistor(self.R3_value)

# Port E
self.C1 = Capacitor(self.C1_value, self.fs)

# define R-TypeAdaptor
self.R_adaptor = PolarityInverter(
    RTypeAdaptor([self.R1, self.R2, self.R3,
        self.C1], self.impedance_calc, 0)
)

self.Vin = IdealVoltageSource(self.
    R_adaptor)
```

This APF configuration is unique in that it cannot be realized in the analog domain, because negative resistance cannot be achieved. We hope that `pywdf` can further allow users to experiment with physically impossible circuits in the digital domain, either by aiming to replicate known filter behaviors, or by tweaking existing circuits in ways that otherwise could not be done.

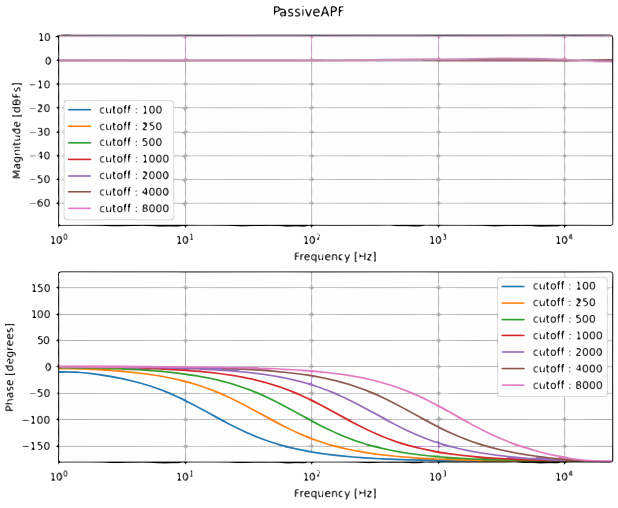


Figure 9: Cutoffs in the APF-WDF implementation.

5. CONCLUSIONS AND FUTURE WORK

This paper presents `pywdf`, an open-source Python library to prototype and evaluate WDFs. The library is available under an MIT license on Github and it provides examples of implementation and usage for different circuits. Currently, the library models each electrical component, such as Resistor, Capacitor, Inductor, Switch, Diode Pair, Adaptors and Sources. We have shown its ease of use in prototyping and analyzing digitally modeled analog circuits, as well as its efficacy in replicating their behavior. Overall, we believe `pywdf` to be a powerful resource with a low barrier of entry to begin experimenting with virtual analog modeling.

Future work for this project can involve developing additional circuit elements and models. For example, existing literature discusses nullors, which are helpful in modeling ideal operational amplifiers and transistors [26]. This would allow for much greater flexibility in building circuits with the library. We also hope to add support for differentiable wave digital filters, and improved integration with solving R-type adaptors' scattering matrices.

Further, while we have presently implemented a tolerance parameter for the Capacitor model, we would like to add this to more circuit elements to get more realistic modelling, and include analysis options on tolerance behavior such as is described in [27].

Additionally, stateful components like capacitors and inductors are only currently implemented as discretized by the bilinear transform. It would be helpful to add support for additional conformal maps such as the forwards and backwards Euler transforms and others, which are included as parameters for stateful components in `chowdsp_wdf`.

6. ACKNOWLEDGEMENTS

This research was carried out under the project Musical AI - PID2019- 111403GB-I00/AEI/10.13039/501100011033, funded by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación. Many thanks to the great number of anonymous reviewers! The authors would also like to thank Xavier Serra and Jatin Chowdhury for sharing their help and knowledge throughout the process.

7. REFERENCES

[1] A. Fettweis, "Wave digital filters: Theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986.

[2] Alfred Fettweis, "Pseudo-passivity, sensitivity, and stability of wave digital filters," *IEEE Transactions on Circuit Theory*, vol. 19, no. 6, pp. 668–673, 1972.

[3] Alfred Fettweis and K Meerkotter, "On adaptors for wave digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 6, pp. 516–525, 1975.

[4] Giovanni De Sanctis and Augusto Sarti, "Virtual analog modeling in the wave-digital domain," *IEEE transactions on audio, speech, and language processing*, vol. 18, no. 4, pp. 715–727, 2009.

[5] Mattia Verasani, Alberto Bernardini, and Augusto Sarti, "Modeling Sallen-Key audio filters in the Wave Digital domain," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*, March. 2017, pp. 431–435.

[6] Kurt James Werner, Ezra J. Teboul, Seth Cluett, and Emma Azelborn, "Modeling and Extending the Rca Mark Ii Sound Effects Filter," in *Proceedings of the 25-th Int. Conf. on Digital Audio Effects (DAFx20in22)*, G. Evangelista and N. Holighaus, Eds., Sept. 2022, vol. 3, pp. 25–32.

[7] B Psenicka, Francisco J García-Ugalde, and Andrés Romero Mier y Terán, "Synthesis of the low-pass and high-pass wave digital filters.," in *ICINCO-SPSMC*, 2008, pp. 225–231.

[8] Augusto Sarti and Giovanni De Sanctis, "Systematic methods for the implementation of nonlinear wave-digital structures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 2, pp. 460–472, 2008.

[9] Kurt James Werner, Vaibhav Nangia, Julius O. Smith, and Jonathan S. Abel, "A general and explicit formulation for wave digital filters with multiple/multiport nonlinearities and complicated topologies," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2015, pp. 1–5.

[10] D. Franken, Jörg Ochs, and Karlheinz Ochs, "Generation of wave digital structures for networks containing multiport elements," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, pp. 586 – 596, 04 2005.

[11] D. Franken, J. Ochs, and K. Ochs, "Generation of wave digital structures for connection networks containing ideal transformers," in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03.*, 2003, vol. 3, pp. III–III.

[12] Dirk Roosenburg, Eli Stine, Romain Michon, and Jatin Chowdhury, "A Wave-Digital Modeling Library for the Faust Programming Language," June 2021, Zenodo.

[13] Jatin Chowdhury, "chowdsp_wdf: An advanced c++ library for wave digital circuit modelling," *arXiv preprint arXiv:2210.12554*, 2022.

[14] Maximilian Rest, W. Ross Dunkel, Kurt James Werner, and Julius O. Smith III, "Rtwdf—a modular wave digital filter library with support for arbitrary topologies and multiple nonlinearities," in *International Conference on Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, 09/2016 2016.

[15] Jatin Chowdhury and Christopher Johann Clarke, "Emulating diode circuits with differentiable wave digital filters," in *Proceedings of the 19th Sound and Music Computing Conference. Zenodo, Saint-Etienne, France*, 2022, pp. 2–9.

[16] Champ C. Darabundit, Dirk Roosenburg, and Julius O. Smith III, "Neural Net Tube Models for Wave Digital Filters," in *Proceedings of the 25-th Int. Conf. on Digital Audio Effects (DAFx20in22)*, G. Evangelista and N. Holighaus, Eds., Sept. 2022, vol. 3, pp. 153–160.

[17] Kurt Werner, Vaibhav Nangia, Alberto Bernardini, Julius Smith, and Augusto Sarti, "An improved and generalized diode clipper model for wave digital filters," 10 2015.

[18] Stefano D'Angelo, Leonardo Gabrielli, and Luca Turchet, "Fast approximation of the lambert w function for virtual analog modelling," 09 2019.

[19] Kurt Werner, Vaibhav Nangia, Julius Smith, and Jonathan Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," 11 2015.

[20] David T. Yeh and Julius Orion Smith, "Discretization of the '59 Fender Bassman tone stack," in *Proc. Int. Conf. Digital Audio Effects (DAFx-06)*, Sept. 2006.

[21] Jatin Chowdhury, "Wave Digital Circuit Models with R-Type Adaptors — jatinchowdhury18.medium.com," https://jatinchowdhury18.medium.com/wave-digital-filter-circuit-models-with-r-type-adaptors-39ad0ad658ce, Oct. 2022.

[22] Gustav Anthon, *Evaluation of Nonlinearities in a Diode Clipper Circuit based on Wave Digital Filters*, Ph.D. thesis, Universitat Pompeu Fabra, Sept. 2022.

[23] Angelo Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," *Journal of the Audio Engineering Society*, pp. 1–24, Feb. 2000.

[24] Norbert Herencsar, Jaroslav Koton, Kamil Vrba, Shahram Minaei, and Izzet Cem Göknar, "Voltage-mode all-pass filter passive scheme based on floating negative resistor and grounded capacitor," in *2015 European Conference on Circuit Theory and Design (ECCTD)*, 2015, pp. 1–4.

[25] Stanley P Lipshitz and John Vanderkooy, "In-phase crossover network design," *Journal of the Audio Engineering Society*, vol. 34, no. 11, pp. 889–894, 1986.

[26] Kurt James Werner, "Virtual analog modeling of audio circuitry using wave digital filters," Dissertation, Stanford University, Stanford, 12/2016 2016, .

[27] Jatin Chowdhury, "Bad Circuit Modelling Episode 1: Component Tolerances — jatinchowdhury18.medium.com," https://jatinchowdhury18.medium.com/bad-circuit-modelling-episode-1-component-tolerances-3ffdbe4e980c.

Demo Track + Late Breaking Results

DYNAMIC STOCHASTIC WAVETABLE SYNTHESIS

Raphael Radna

Department of Music
University of California, Santa Barbara
Santa Barbara, CA
rradna@ucsb.edu

ABSTRACT

Dynamic Stochastic Synthesis (DSS) is a direct digital synthesis method invented by composer Iannis Xenakis and notably employed in his 1991 composition *GENDY3*. In its original conception, DSS generates periodic waves by linear interpolation between a set of breakpoints in amplitude–time space. The breakpoints change position each period, displaced by random walks via high-level parameters that induce various behaviors and timbres along the pitch–noise continuum. The following paper proposes Dynamic Stochastic Wavetable Synthesis as a modification and generalization of DSS that enables its application to table-lookup oscillators, allowing arbitrary sample data to become the basis of a DSS process. We describe the considerations affecting the development of such an algorithm and offer a real-time implementation informed by the analysis.

1. INTRODUCTION

Iannis Xenakis proposed Dynamic Stochastic Synthesis (DSS) as a time-domain method of producing "complex sonorities" with "numerous and complicated" transients [1]. In DSS, the cyclical portion of a periodic wave (*wave cycle*) is defined by a number of breakpoints in amplitude–time space. Waves are produced by linear interpolation between adjacent breakpoints. The breakpoints shift positions each period, continuously affecting the pitch, amplitude, and timbre of the synthetic tone produced and giving rise to its "dynamic" character (Fig. 1).

The "stochastic" element refers to the displacement of the breakpoints in both dimensions by random walks. Various probability distributions (Cauchy, logistic, etc.) can be applied, each affecting the movement of the breakpoints in its distinctive way [2]. Xenakis also specified high-level parameters to constrain the displacement: the random walk step size governs its magnitude, and elastic barriers reflect excessive values back within a specified range. These parameters influence the degree of similarity between successive wave cycles. If only slight variation is permitted, tones of stable pitch and timbre are produced; conversely, a parameter state that allows profound dissimilarities between wave cycles causes the output to tend towards noise. Additionally, the number of breakpoints used correlates with spectral brightness.

The original DSS software *GENDY* dates from the late 1980s and operated in a non-realtime capacity. Since then, several researchers have implemented DSS with experimental alterations

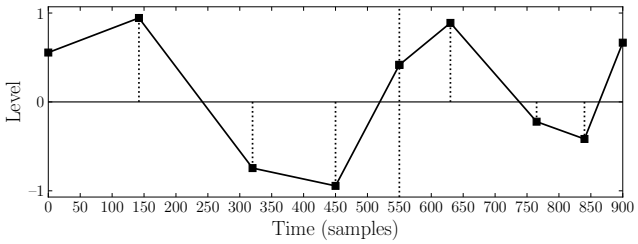


Figure 1: *Two contiguous DSS wave cycles. The second is a variation of the first, produced by stochastic displacement of its four breakpoints. The period of the second cycle is shorter, indicating an increase in pitch.*

that seek to enhance its sound or functionality in some way, including interactive operation and analysis [3], time-variant parameter automation [4], wave-cycle sequencing strategies [5], touch-sensitive and gestural interfaces [6], and applications of physical models to the algorithm [7, 8]. Our own previous DSS-related research culminated in the *Xenos* plug-in synthesizer, which introduced pitch quantization to DSS [9].

While these contributions have all helped to extend and sustain interest in DSS, none have addressed the inherent limitation of its basis in breakpoint interpolation synthesis. One possibility in this direction is the application of DSS to standard wavetable oscillators. Although DSS does not read a lookup table directly, it realizes equivalent sample data at runtime through breakpoint interpolation; this process bears similarity to dynamic wavetable techniques, such as scanned synthesis [10, 11]. Dynamic Stochastic Wavetable Synthesis (DSWS) thus uses the procedures of DSS to apply its characteristic, stochastic pitch and timbre evolution to arbitrary sample data, instead of generating abstract waves from linear ramps. In this way, DSWS reimagines DSS as an audio processor rather than a synthesizer, increasing the timbral range of the technique, enabling general DSS-based modulation, and facilitating interpolation between arbitrary timbres and DSS.

2. DYNAMIC STOCHASTIC WAVETABLE SYNTHESIS

The DSWS prototype described in this paper is implemented in Max/MSP using the GenExpr metalanguage for audio programming. The code is open source and available for download from https://github.com/raphaelradna/dsws.

2.1. Wavetable Segmentation

While DSS begins with the definition of breakpoints for linear interpolation, DSWS begins by segmenting a wavetable into a num-

ber of regions whose pitches and amplitudes will be individually manipulated. For simplicity and efficiency, we have chosen to divide the wavetable into an arbitrary number of segments of equal size. Our implementation allows for as few as one segment, in which case the entire wavetable is affected uniformly, or as many as 256. The number of segments is variable at runtime. As in DSS, a greater number of segments results in a brighter timbre (Fig. 2).

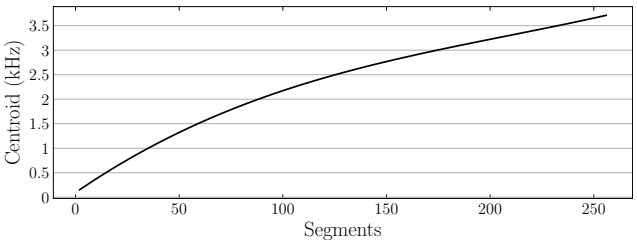


Figure 2: *The spectral centroid increases with the number of wavetable segments. Measurements were taken using a sinusoidal wavetable and with otherwise constant parameters: a steady pitch of A1 (55 Hz) and maximum amplitude fluctuation.*

2.2. Table Lookup, Modification, and Output

DSWS has at its core a wavetable oscillator that derives sample values by reading through a lookup table at a variable frequency [12]. It imposes DSS-like behavior on a wavetable of N samples by dividing it into M segments and applying individual, random, pitch and amplitude deviations to each segment. For each sample in the input wavetable $x[n]$, where $0 \leq n < N$, the index m of its containing segment is the greatest integer less than M multiplied by ϕ, its phase within the wavetable:

$$m = \lfloor M\phi \rfloor \,, \tag{1}$$

where the phase ϕ in range $0 \leq \phi < 1$ is given by

$$\phi = \frac{n}{N} \,. \tag{2}$$

The deviations are regenerated each period and stored in series P and A, respectively, each of length M. The pitch deviation $P[m]$ of the segment containing $x[n]$ modulates the base oscillator pitch p. These values, initially expressed as floating-point MIDI notes, are summed, converted into a frequency in Hz, and divided by the sampling rate f_s to produce a phase increment φ for table lookup:

$$\varphi = \frac{440 \cdot 2^{\frac{p+P[m]-69}{12}}}{f_s} \,. \tag{3}$$

The effective frequency therefore changes with each segment as the table is read and fluctuates around p. By contrast, the per-segment amplitude deviation is added to the wavetable data. To avoid introducing discontinuities into the wave cycle, an individual amplitude deviation a is derived for each input sample $x[n]$ by linear interpolation between the amplitude deviation $A[m]$ of its containing segment and that of the subsequent segment $A[k]$:

$$a = (1-\mu)A[m] + \mu A[k] \,, \tag{4}$$

with the index k of the subsequent segment given by

$$k = \begin{cases} m+1, & \text{if } m < M-1 \\ 0, & \text{if } m \geq M-1 \end{cases} , \tag{5}$$

and the interpolation parameter μ attained by

$$\mu = M\phi - m \,. \tag{6}$$

Finally, a (4) is added to $x[n]$ to produce output sample $y[n]$. To prevent clipping, any $y[n]$ greater than 1 or less than -1 is reduced or increased, respectively, by the amount d that it lies out of range:

$$y[n] = \begin{cases} 1-d, & \text{if } x[n]+a > 1 \\ -1+d, & \text{if } x[n]+a < -1 \\ x[n]+a, & \text{if } -1 \leq x[n]+a \leq 1 \end{cases} , \tag{7}$$

where

$$d = |x[n]+a| - 1 \,. \tag{8}$$

DSWS can thus be conceptualized in part as a segmented, stochastic wavefolder (Fig. 3) [13].

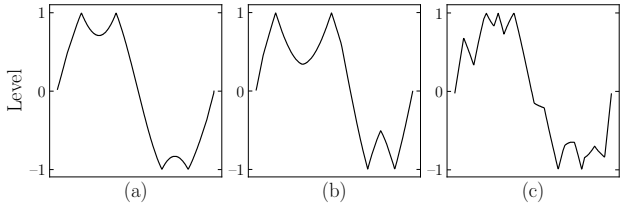


Figure 3: *Asymmetrical amplitude folding of a sinusoidal wavetable resulting from a DSWS process with (a) two segments, (b) four segments, and (c) sixteen segments.*

2.3. Deviation Generation and Iteration

The per-segment pitch and amplitude deviations are produced by random walks that are iterated every wave cycle. New values are generated in the range $[-1, 1]$ according to some probability distribution (we use uniform randomness for demonstration purposes) and added to the previous deviations to produce new ones. Two parameters influence this process: the step size scales the random value, constraining its magnitude and thus limiting the difference between deviations across cycles, and the barrier position defines the random walk space, i.e., the minimum and maximum possible deviation values. The sum of the previous deviation and new random value can in general fall outside of the range defined by the barriers; following Xenakis's design, any such sums are reflected back into range in the manner of (7) and (8).

Because the random value can be either positive or negative, the deviation can either increase or decrease from one wave cycle to the next, regardless of the step size. The barrier position parameter limits the range of the random walks symmetrically around a single value; the amplitude walks center around zero, while the pitch walks center around p, the base oscillator pitch. As a result, reducing both barrier position parameters to zero reproduces the input wavetable at a constant pitch. The parameters of the pitch random walk are specified in equal-tempered semitones, and those of the amplitude random walk are specified as proportions of the full amplitude range.

2.4. Single-Segment Pitch Fluctuation

Manipulating individual sections of a wavetable in the manner of DSWS introduces knees in the output wave at the segment bound-

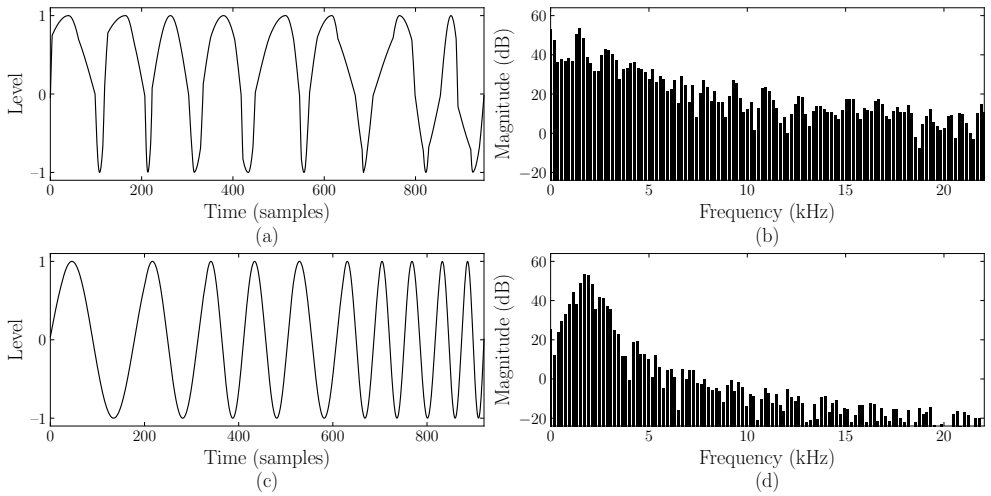


Figure 4: *Effects of DSWS pitch fluctuation on the waveform and spectrum in (a), (b) standard and (c), (d) single-segment modes. The deformations visible in waveform (a) are caused by eight extreme pitch deviations per wave cycle. These scatter partials throughout spectrum (b), including by aliasing. Because the frequency of waveform (c) modulates only once per cycle, its sinusoidal shape is preserved, producing spectrum (d), which shows less energy in the high-frequency range despite otherwise identical parameters: a center pitch of C5 (523.25 Hz), pitch barrier range of ± two octaves, pitch step size of six semitones, and no amplitude fluctuation.*

aries, causing high-frequency distortion. While the amplitude fluctuation writes these directly into the sample data, the pitch fluctuation also causes them implicitly by modifying the phase increment for each segment, potentially tens or hundreds of times per cycle. As a result, pitch fluctuation is not generally timbre-neutral, but also affects the spectrum of the resulting tone.

To better isolate perceived pitch and timbre transformations, we can treat the entire wavetable as a single segment for the purpose of pitch fluctuation, regardless of the number of segments used for amplitude fluctuation. In this case, the pitch fluctuation occurs once per cycle, i.e., at oscillator frequency in Hz. Since this rate typically still falls within the microsound timescale, rapid pitch modulation remains perceptible while timbral distortions are reduced (Fig. 4). This method produces more volatile pitch movement for the same step size and barrier position parameters, because the frequency of the wave cycle depends on a single pitch deviation instead of the average of several. It also contradicts DSS, which stipulates an equal number of pitch and amplitude fluctuations per cycle, but may be subjectively preferable in its adaptation to wavetable synthesis due to its ability to preserve the shape, and therefore timbre, of a particular wavetable. We thus propose single-segment pitch fluctuation as the default behavior of DSWS.

3. DISCUSSION

This section elaborates on our DSWS prototype, providing insight into certain design choices and their ramifications, and suggesting possible alternatives or areas for further development.

3.1. Wavetable Selection and Sound Quality

Sine, triangle, square, and sawtooth wavetables are included in our DSWS implementation. The classical waveforms were produced by additive synthesis using 64 harmonics, which, at a sampling rate of 44.1 kHz, prevents aliasing for fundamental frequencies up to approximately 344.5 Hz. Further antialiasing measures were not taken, as the linear interpolation of DSWS, like that of DSS, ultimately produces its own aliasing artifacts [2]. A more complete implementation could apply additional solutions for antialiased wavetable synthesis, such as those described in [14], and offer band-limited interpolation algorithms for producing the per-sample amplitude deviations [15].

Our implementation can also generate noise wavetables using uniform randomness, load external single-cycle sample data in .wav format, or use an empty wavetable. Since using a wavetable with all values zeroed produces linear ramps between amplitude deviations, we can say that DSS is a special case of DSWS.

3.2. Wavetable Segmentation Method

Although we divide the wavetable into equal parts, other segmentation methods could be applied and affect the results in distinct ways. A piecewise linear approximation algorithm, for example, could fit the segment boundaries to places of pronounced change in the slope of the sample data (Fig. 5). Furthermore, a method that optimizes the fit within a specified error tolerance, as in [16], could determine the ideal number of segments and their boundaries for arbitrary wavetables. Further investigation is needed to assess the significance of the wavetable segmentation method in DSWS.

3.3. Optimization Considerations

The DSWS prototype reads the sample data from a source wavetable, transforms them, and writes the results into a second wavetable, which is read at the oscillator output. Although every wave cycle is regenerated from the source data in this manner, continuity between them is maintained because the deviations are influenced by their own previous values. Our implementation uses a second wavetable primarily for the purpose of visualization; an alternative and possibly more efficient design would calculate the values

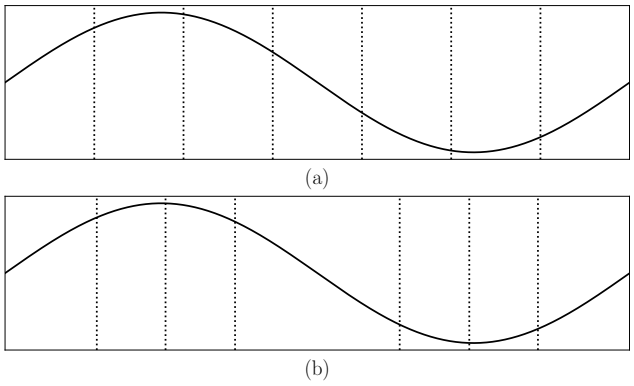


Figure 5: *Segmentation of a sinusoidal wavetable into seven parts using (a) equal distribution and (b) piecewise linear fit via global optimization of the least squares method* [17].

continuously and write them directly to the output buffer, without otherwise storing them.

The per-segment pitch and amplitude deviation data defining the transformation have the same memory footprint as the amplitude–time breakpoints of traditional DSS. In a polyphonic implementation using this method, each voice would have unique arrays of deviations but read a single source wavetable, avoiding expensive array-copy operations. Because the barrier position parameters control the random walk space, i.e., the degree of pitch or amplitude deviation from the source, this approach also affords interpolation between the original wavetable and its transformation.

3.4. Limitations

A complete reproduction of all DSS features is beyond the scope of this work. Therefore, certain components of the mature form of the technique, namely second-order random walks and variable probability distributions, have not been implemented. This decreases the number of parameters, which simplifies the prototype, but also reduces its sound design potential.

Additionally, because the per-segment pitch fluctuations affect table read frequency and are not written explicitly into the sample data, the built-in waveform display object in Max/MSP does not represent the horizontal distortions they produce. These are visualized, however, in the oscilloscope-style display.

4. CONCLUSION AND FUTURE WORK

We have described DSWS, an experimental synthesis method exploring the application of Iannis Xenakis's DSS algorithm to table-lookup oscillators. We also presented a prototype demonstrating its basic principles. The technique could be developed further, e.g., by integrating additional probability distributions, segmentation methods, and interpolation algorithms; or by combining with sophisticated wavetable techniques, such as wavetable crossfading and multiple-wavetable synthesis [18]. Furthermore, by expanding DSS into a sample-processing paradigm, DSWS suggests implementation as a filter that applies an iterative window of stochastic pitch and amplitude distortions to streaming audio input.

5. REFERENCES

[1] Iannis Xenakis, *Formalized Music: Thought and Mathematics in Composition*, Pendragon Press, Stuyvesant, revised edition, 1992.

[2] Marie-Hélène Serra, "Stochastic Composition and Stochastic Timbre: *GENDY3* by Iannis Xenakis," *Perspectives of New Music*, vol. 31, no. 1, pp. 236–257, 1993.

[3] Peter Hoffmann, "The New GENDYN Program," *Computer Music Journal*, vol. 24, no. 2, pp. 31–38, 2000.

[4] Andrew Brown, "Extending Dynamic Stochastic Synthesis," in *Proc. 2005 Int. Computer Music Conf.*, Barcelona, Spain, 2005, pp. 111–114.

[5] Sergio Luque, "The Stochastic Synthesis of Iannis Xenakis," *Leonardo Music Journal*, vol. 19, pp. 77–84, 2009.

[6] Nick Collins, "Implementing Stochastic Synthesis for SuperCollider and iPhone," in *Proc. Xenakis Int. Symposium*, 2011.

[7] Luc Döbereiner, "Phingen: A Physically Informed Stochastic Synthesis Generator," in *Proc. 2011 Int. Computer Music Conf.*, Huddersfield, UK, 2011, pp. 57–60.

[8] Emilio Rojas and Rodrigo Cádiz, "A Physically Inspired Implementation of Xenakis's Stochastic Synthesis: Diffusion Dynamic Stochastic Synthesis," *Computer Music Journal*, vol. 45, pp. 48–66, 2022.

[9] Raphael Radna, "Xenos: Xenharmonic Stochastic Synthesizer," M.S. thesis, University of California, Santa Barbara, 2022.

[10] Robert Tubb, Anssi Klapuri, and Simon Dixon, "The Wablet: Scanned Synthesis on a Multi-Touch Interface," in *Proc. 15th Int. Conf. Digital Audio Effects (DAFx-12)*, York, UK, 2012, pp. 192–199.

[11] Tendsin Mende, Lars Engeln, Matthew McGinity, and Rainer Groh, "Creative sound modeling with signed distance fields," in *Mensch und Computer 2022 - Workshopband*, Bonn, 2022, Gesellschaft für Informatik e.V.

[12] Curtis Roads, *The Computer Music Tutorial*, MIT Press, Cambridge, MA, 1996.

[13] Fabián Esqueda, Henri Pöntynen, Vesa Välimäki, and Julian Parker, "Virtual Analog Buchla 259 Wavefolder," in *Proc. 20th Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, 2017, pp. 192–199.

[14] Günter Geiger, "Table Lookup Oscillators Using Generic Integrated Wavetables," in *Proc. 9th Int. Conf. Digital Audio Effects (DAFx-06)*, Montreal, Canada, 2006.

[15] Julius O. Smith, *Digital Audio Resampling Home Page*, http://www-ccrma.stanford.edu/~jos/resample/, January 28, 2002.

[16] Bernd Hamann and Jiann-Liang Chen, "Data Point Selection for Piecewise Linear Curve Approximation," *Computer Aided Geometric Design*, vol. 11, pp. 289–301, 1994.

[17] Charles Jekel and Gerhard Venter, "pwlf: A Python Library for Fitting 1D Continuous Piecewise Linear Functions," 2019.

[18] Andreas Franck and Vesa Välimäki, "Higher-Order Integrated Wavetable Synthesis," in *Proc. 15th Int. Conf. Digital Audio Effects (DAFx-12)*, York, UK, 2012.

TOWARDS HIGH SAMPLING RATE SOUND SYNTHESIS ON FPGA

Romain Michon,[a] Julien Sourice,[b] Victor Lazzarini,[b] Joseph Timoney,[b] and Tanguy Risset[c]

[a]Univ. Lyon, Inria, INSA Lyon, CITI, EA3720, 69621 Villeurbanne, France
[b]Maynooth University, Maynooth, Ireland
[c]Univ Lyon, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France
romain.michon@inria.fr

ABSTRACT

This "Late Breaking Results" paper presents an ongoing project aiming at providing an accessible and easy-to-use platform for high sampling rate real-time audio Digital Signal Processing (DSP). The current version can operate in the megahertz range and we aim to achieve sampling rates as high as 20 MHz in the near future. It relies on the Syfala compiler which can be used to program Field Programmable Gate Array (FPGA) platforms at a high level using the FAUST programming language. In our system, the audio DAC is directly implemented on the FPGA chip, providing exceptional performances in terms of audio latency as well. After giving an overview of the state of the art of this field, we describe the way this tool works and we present ongoing and future developments.

1. INTRODUCTION

Sampling rate selection in the context of real-time audio Digital Signal Processing (DSP) is impacted by a wide range of factors. Beside psychoacoustic considerations (i.e., Nyquist frequency, human hearing range, etc.), aliasing, hardware, and computational power all play an important role. While 48 kHz is fairly standard as it puts the Nyquist frequency (24 kHz) well above the human hearing range, it is fairly common nowadays to see professional audio systems running at 96, 192, 384, and even 768 (in some rare cases) kHz, minimizing aliasing and audio latency (to the detriment of computational power).

Until recently, the use of sampling rates in the megahertz range for real-time applications was mostly reserved to researchers with very specific needs. At this rate, real-time constraints are such that standard processor architectures which are traditionally used for audio DSP (i.e., CPUs, DSPs, and microcontrollers) can't really keep up. That's why ASICs[1] and FPGAs[2] are used when such performances are needed. On this kind of platform, the speed at which an algorithm can be run is mostly limited by the maximum clock tolerated by the system and the "length" of the corresponding electronic circuit (i.e., the time it takes to go from point A to point B in the algorithm).

In the music technology industry, FPGAs have been used for high sampling rate applications in a few rare products. A good example is the Novation Summit[3] which is a "hybrid" analog/digital synthesizer where basic digital oscillators (e.g., sine, etc.) are implemented on an FPGA and computed at 24 MHz. They "market" this as "digital approximating analog," which is certainly appealing in the analog synth community.

Beyond simple waveform oscillators, implementing more complex audio DSP algorithms on an FPGA is a notoriously hard task. The use of Hardware Description Languages (HDLs) such as Verilog or VHDL combined with fixed point arithmetic makes the programming of this kind of platform completely out of reach to most audio DSP programmers. While some high-level environments are available such as Simulink[4] or Vivado Block Design,[5] they're almost all based on the combination of pre-"compiled" ("synthesized"[6] using FPGA terminology) objects, significantly limiting the scope of what can be implemented.

We recently released Syfala[7] [1], the first "audio DSP to FPGA compiler" relying on the FAUST programming language[8] [2]. This open source tool allows us to fully program a series of Digilent development boards (i.e., Zybo Z7-10/20 and Genesys) based on Xilinx/AMD FPGAs to carry out real-time audio signal processing tasks. In this context, we explored a wide range of target applications leveraging the unique performances of FPGAs for audio DSP: ultra-low latency processing [1, 3], spatial audio [4], etc.

While FPGAs can easily keep up with very high audio sampling rates (in the megahertz range, as mentioned above), the fastest audio codec[9] chips available on the market such the Analog Devices ADAU1787 don't go beyond 768kHz. There are two potential solutions to this problem: (i) using general-purpose Analog to Digital/Digital to Analog Converters (ADC/DAC), etc., (ii) implementing audio ADC/DACs directly on the FPGA. Both methods require the use of additional circuitry to implement reconstruction filters, carry out impedance matching, etc. The main disadvantage of (i) is that interfacing an external chip operating in the megahertz range with an FPGA through its GPIOs can be challenging for prototyping if the circuit is not properly shielded. On the other hand, (ii) offers an extremely robust and reliable solution since everything happens directly on the FPGA. If the sampling rate is high enough, there's no need for complex reconstruction filters and a simple RC lowpass filter consisting of a resistor and a capacitor (providing a very slow roll-off) is sufficient for this task.

[1] *Application-Specific Integrated Circuit*.

[2] *Field-Programmable Gate Array.*

[3] https://novationmusic.com/en/synths/summit – All URLs provided in this paper have been verified on April 23d, 2023.

[4] https://www.mathworks.com/discovery/fpga-programming.html

[5] https://www.xilinx.com/products/design-tools/vivado.html

[6] In the context of FPGAs, the word "synthesized" is the equivalent to "compiled" on other platforms.

[7] https://github.com/inria-emeraude/syfala

[8] https://faust.grame.fr

[9] Throughout this paper, "audio codec" will refer to a hardware component implementing an audio ADC/DAC (not an audio compression algorithm).

In this "late breaking result" paper, we present the current status of an ongoing project aiming at providing built-in audio ADC/DAC support as part of the Syfala tool-chain without using any additional complex hardware. Our goal is to offer the same level of performances as commercial audio codecs while allowing for sampling rates in the megahertz range. This will potentially open the way to a broad range of new developments towards improving virtual analog systems, considering audio DSP from a more continuous standpoint, reducing aliasing and artifacts, etc.

First, we give a brief overview of the state of the art of the field of ADC and DAC design as well as of existing works around implementing ADCs and DACs on FPGAs. Then we demonstrate how we implemented a second order delta-sigma ($\Delta\Sigma$) DAC in our Syfala tool-chain and we present its performances. We finally discuss the prospect and potential challenges of implementing a higher order $\Delta\Sigma$ DAC as well as an ADC on an FPGA. We also briefly talk about some possible difficulties related to running audio DSP algorithms in real-time at a high sampling rate.

2. BACKGROUND

2.1. $\Delta\Sigma$ ADCs and DACs

$\Delta\Sigma$ modulation [5] is by far the most commonly used technique for implementing audio ADC and DAC nowadays [**?**]. Most audio codec chips available on the market rely on this method. A first order $\Delta\Sigma$ DAC is based on an integrator, a 1-bit DAC, and a comparator (see Figure 1). It converts a digital signal (i.e., a stream of samples) into a stream of pulses (bits) generated at a high frequency. The more pulses, the higher the analog voltage at the output of the DAC. The audio sampling rate, the clock of the DAC, and the order of the $\Delta\Sigma$ modulator are interconnected parameters which all influence the resulting Signal to Noise Ratio (SNR) [6]. A higher order $\Delta\Sigma$ modulator allows for a lower OverSampling Ratio (OSR) [6]. The OSR directly determines the SNR of the system. For example, if a first order $\Delta\Sigma$ modulator is used with an OSR of 32, then the SNR will be around 40 dB. With an OSR of 32, for an audio sampling rate of 48 kHz, the clock of the $\Delta\Sigma$ modulator has to be 1.536 MHz. On the other hand, if a fifth order $\Delta\Sigma$ (which is the standard used for audio codec chips) modulator is used with the same configuration, then the SNR will increase to 124 dB. Note that there exists many different $\Delta\Sigma$ modulator topology when going beyond second order presenting different advantages and tradeoffs in terms of numerical stability [6]. These SNR figures are independent from the bit depth of incoming samples which can also be a source of quantization noise. Hence, if 16 bits audio samples are provided to a fifth order $\Delta\Sigma$ DAC with an OSR of 32, the resulting SNR should be around 98 dB.

To summarize, a higher order $\Delta\Sigma$ modulator can help increasing the audio sampling rate of the system while preserving a reasonable SNR. Also, the higher the clock of the $\Delta\Sigma$ modulator, the better the performances of the system in all respects.

Once the stream of pulses is generated, it must of course be filtered (lowpass) to reconstruct the analog signal. Using a high audio sampling rate can help decrease the complexity of the filter needed for this task. The lower the order of the reconstruction filter, the more progressive its roll-off. Hence, the -6 dB per octave provided by a simple first order RC filter (which can be implemented with just a resistor and a capacitor) is enough if the audio sampling rate is in the megahertz range.

$\Delta\Sigma$ ADC work in a similar way but are usually slightly more challenging to implement as they imply the use of a hardware comparator which is not necessarily built-in/directly available on FPGAs [7].

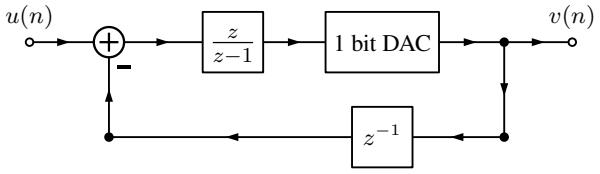


Figure 1: *First order $\Delta\Sigma$ DAC where $u(n)$ is the digital signal input and $v(n)$ is a stream of pulses.*

2.2. $\Delta\Sigma$ ADCs and DACs on FPGAs

FPGAs are a convenient platform for implementing $\Delta\Sigma$ DACs and ADCs (if the FPGA provides differentiated general purpose inputs). Indeed, running a $\Delta\Sigma$ modulator at a very high speed (more than 100 MHz) and connecting its output to a General Purpose Input/Output (GPIO) is trivial. In fact, coding a first order $\Delta\Sigma$ DAC is often a basic exercise/example when learning FPGA programming.[10] Implementing a second order $\Delta\Sigma$ DAC is not that much more complicated and examples of such projects can be easily found on the web.[11] Things get significantly more complex when considering third order and beyond because of stability issues. Hence, while constructing a third, forth, or fifth order $\Delta\Sigma$ DAC is fairly straightforward, formatting coefficients and preventing numerical/rounding errors is potentially very challenging. Various tools such as the Matlab delta-sigma toolbox[12] can help with that. Additionally, various papers on on this topic have been written over the years [8, 9, 10, 11].

$\Delta\Sigma$ ADCs face more or less the same challenges as their DAC counterparts. As mentioned previously, implementing a $\Delta\Sigma$ ADC on an FPGA is significantly simpler if the chip provides differential inputs as those can potentially be used to implement the required differentiator at the beginning of the algorithm [6]. If the FPGA chip doesn't provide differential inputs, then a hardware differentiator should be used, making the design significantly more complex and hence "defeating the purpose" of using an FPGA for this task.

3. IMPLEMENTATION IN THE CONTEXT OF SYFALA

Syfala [1] allows us to run FAUST programs on Xilinx FPGAs-based boards such as the Digilent Zybo Z7 or Genesys without having to write a single line of hardware description language code (which is normally used to program FPGAs).

The standard version of the Syfala compiler can target various audio codec chips including the one built-in on the Digilent FPGA Zybo (see Figure 2) and Genesys boards. We implemented a custom first and second order $\Delta\Sigma$ DAC integrating to the Syfala

[10] https://www.fpga4fun.com/PWM_DAC_2.html

[11] https://github.com/hamsternz/second_order_sigma_delta_DAC

[12] https://www.mathworks.com/matlabcentral/fileexchange/19-delta-sigma-toolbox

tool-chain and that can be used as an alternative for audio codecs. Hence, if the `-sd` option is used when calling the Syfala compiler, the audio output of the system is implemented through a second order $\Delta\Sigma$ DAC. The current version is multichannel which means that a new $\Delta\Sigma$ DAC is instantiated for each output of the FAUST DSP program and associated to a GPIO on the board.

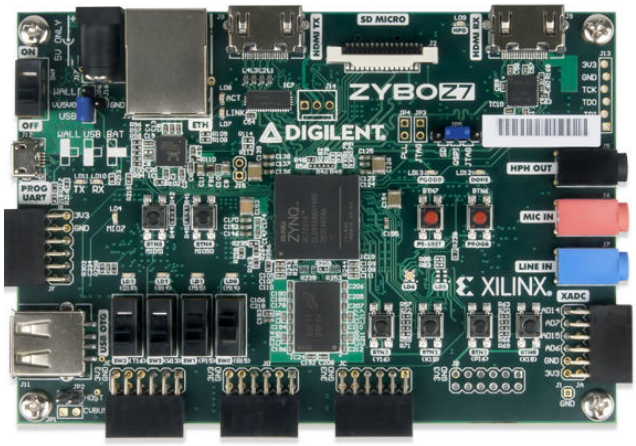


Figure 2: *The Digilent Zybo Z7 FPGA board used for this project.*

When the `-sd` option is used, the audio sampling rate of the system is automatically switched to 5 MHz. The master clock of the FPGA is 125 MHz yielding an OSR of 25 and hence providing a SNR of about 70 dB. At such a high sampling rate, using a one pole RC filter as described in §2.1 is acceptable and is enough to minimize aliasing. Hence, each DAC GPIO must be connected to such a filter. For a cut-off frequency of 20 kHz, a 880 Ω resistor and a 0.01 μF capacitor can be used. A 10 μF capacitor should also be put in series to get rid of DC. Different values for the sampling rate can be specified too using the appropriate Syfala option (`--sample-rate`), bearing in mind that diminishing its value increases the SNR and vice versa.

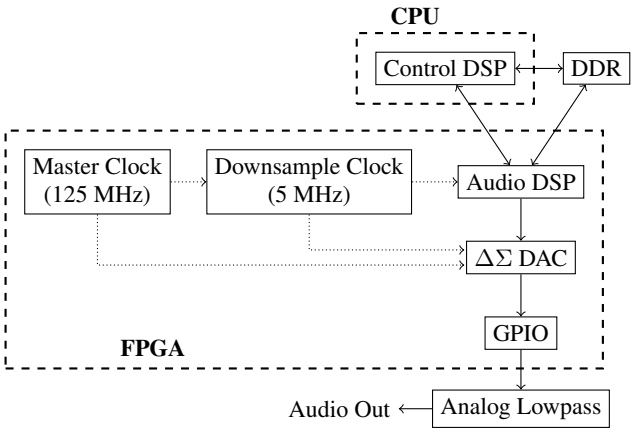


Figure 3: *Implementation overview of the system. Clock signals are depicted with dotted arrows.*

As the $\Delta\Sigma$ DAC is seamlessly integrated to the Syfala tool-chain (see Figure 3), all the other functionalities of this environment remain active/available (i.e., use of DDR for long delays, control computations happening on the ARM processor which is part of the board, etc.).

4. HIGH SAMPLING RATE AUDIO DSP

Running audio DSP algorithms in real-time at a high audio sampling rate can present various challenges which are often related to precision/numerical errors. This of course can greatly vary from one algorithm to another, but obviously running more samples through a filter or computing a wave-table oscillator index using a phasor based on a delta increment can all be significantly impacted by this. A good example of that is the default sine wave oscillator in FAUST which is based on a wave table (represented here by the `sin` function) and whose implementation takes the following form:

```
phasor(freq) = (+(freq/ma.SR) ~ ma.frac);
osc(freq) = sin(phasor(freq)*2*ma.PI);
```

The `~` in `phasor` represents a recursive signal and `ma.frac` yields the fractional part of a decimal number. In that case, `freq/ma.SR` is not precise enough at 5 MHz to provide an accurate frequency. This is just a simple example to demonstrate that high audio sampling rate can be really enabling on one side but can also creates many issues on the other.

5. FUTURE WORK

The ultimate goal for this project is to eventually integrate a 5th order $\Delta\Sigma$ DAC as well as ADC to the Syfala tool-chain.

As mentioned in §2.2, implementing a 5th order $\Delta\Sigma$ DAC on an FPGA is not trivial because of the potential instability of this kind of algorithms due to numerical/rounding errors. This problem is reinforced by the fact that these errors tend to get worse as the clock of the $\Delta\Sigma$ modulator increases.

VHDL-based solutions do exist though [11] and we plan to exploit them to potentially reach this goal. Alternatively, we're also particularly interested in investigating the potential use of FAUST for writing these algorithms and comparing their performances with their VHDL counterparts. The FAUST version would mitigate numerical errors because of the use of floating points whereas the VHDL version would probably be more efficient from a computational standpoint but more prompt to rounding errors. It would also be interesting to investigate the use of FloPoCo[13] [12] (which is a tool for generating arithmetic cores on FPGAs) in this context.

Along the same lines, we hope to be able to provide a $\Delta\Sigma$ ADC in Syfala using similar approaches to that described in [7]. This should be possible on the boards supported by Syfala which all have differential general purpose inputs. As for the DAC, it will be interesting to compare a FAUST implementation to a VHDL one.

6. CONCLUSIONS

The domain of high sampling rate real-time audio DSP is widely unexplored because it has been out of reach for a very long time. Many algorithms would benefit from running at a higher sampling rate, mitigating artifacts and potentially opening new possibilities such as improving virtual analog systems, considering audio DSP

[13] `https://www.flopoco.org/`

from a more continuous standpoint, etc. The system that we presented in this paper and that we're currently developing provides an accessible and easy-to-use platform for this kind of applications without making compromises in terms of performances, quality, etc. Beyond this, it might also allows us to simplify the overall design of Syfala by completely getting rid of audio codecs and providing even better latency performances.

7. ACKNOWLEDGMENTS

This project has been partially funded by the French ANR (Agence Nationale de la Recherche) through the FAST project[14] (ANR-20-CE38-0001).

8. REFERENCES

[1] Maxime Popoff, Romain Michon, Tanguy Risset, Yann Orlarey, and Stéphane Letz, "Towards an fpga-based compilation flow for ultra-low latency audio signal processing," in *Proceedings of the 2022 Sound and Music Computing Conference (SMC-22)*, Saint-Étienne, France, June 2022.

[2] Yann Orlarey, Stéphane Letz, and Dominique Fober, *New Computational Paradigms for Computer Music*, chapter "Faust: an Efficient Functional Approach to DSP Programming", Delatour, Paris, France, 2009.

[3] Loïc Alexandre, Pierre Lecomte, Marie-Annick Galland, and Maxime Popoff, "Feedback acoustic noise control with faust on fpga : application to noise reduction in headphones," in *Proceedings of the 2022 Sound and Music Computing Conference (SMC-22)*, Saint-Étienne, France, June 2022.

[4] Romain Michon, Joseph Bizien, Maxime Popoff, and Tanguy Risset, "Making frugal spatial audio systems using field-programmable gate arrays," in *Proceedings of the 2023 New Interfaces for Musical Expression (NIME-23)*, Mexico City, Mexico, 2023 (Paper accepted to the conference and to be published in June 2023).

[5] Francis DeJager, "Deltamodulation, a method of pcm transmission using the 1-unit code," *Philips Research Reports*, vol. 7, no. 6, pp. 442–466, 1952.

[6] Richard Schreier, Gabor C Temes, et al., *Understanding delta-sigma data converters*, vol. 74, IEEE press Piscataway, NJ, 2005.

[7] PA Harsha Vardhini, "Analysis of integrator for continuous time digital sigma delta adc on xilinx fpga," in *Proceedings for the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016, pp. 2689–2693.

[8] AJ Magrath, IG Clark, and MB Sandler, "Design and implementation of a fpga sigma-delta power dac," in *Proceedings for the 1997 IEEE Workshop on Signal Processing Systems. SiPS 97 Design and Implementation formerly VLSI Signal Processing*. IEEE, 1997, pp. 511–521.

[9] Ralf Ludewig, Oliver Soffke, Peter Zipf, Manfred Glesner, Kong Pang Pun, Kuen Hung Tsoi, Kin Hong Lee, and Philip Leong, "Ip generation for an fpga-based audio dac sigma-delta converter," in *Field Programmable Logic and Application*, Jürgen Becker, Marco Platzner, and Serge Vernalde, Eds., Berlin, Heidelberg, 2004, pp. 526–535, Springer Berlin Heidelberg.

[10] R.C.C. Cheung, K.P. Pun, S.C.L. Yuen, K.H. Tsoi, and P.H.W. Leong, "An fpga-based re-configurable 24-bit 96khz sigma-delta audio dac," in *Proceedings. 2003 IEEE International Conference on Field-Programmable Technology (FPT) (IEEE Cat. No.03EX798)*, 2003, pp. 110–117.

[11] Zbigniew Kulka and Marcin Lewandowski, "An fpga-based sigma-delta audio dac," in *New Trends in Audio and Video / Signal Processing Algorithms, Architectures, Arrangements, and Applications SPA 2008*, 2008, pp. 39–42.

[12] Florent De Dinechin and Bogdan Pasca, "Designing custom arithmetic data paths with flopoco," *IEEE Design & Test of Computers*, vol. 28, no. 4, pp. 18–27, 2011.

[14] `https://fast.grame.fr`

REAL-TIME SINGING VOICE CONVERSION PLUG-IN

Shahan Nercessian, Russell McClellan, Cory Goldsmith, Alex M. Fink, and Nicholas LaPenn

iZotope, Inc.
Boston, MA, USA
{shahan|rmcclellan|cgoldsmith|afink|nlapenn}@izotope.com

ABSTRACT

In this paper, we propose an approach to real-time singing voice conversion and outline its development as a plug-in suitable for streaming use in a digital audio workstation. In order to simultaneously ensure pitch preservation and reduce the computational complexity of the overall system, we adopt a source-filter methodology and consider a vocoder-free paradigm for modeling the conversion task. In this case, the source is extracted and altered using more traditional DSP techniques, while the filter is determined using a deep neural network. The latter can be trained in an end-to-end fashion and additionally uses adversarial training to improve system fidelity. Careful design allows the system to scale naturally to sampling rates higher than the neural filter model sampling rate, outputting full-band signals while avoiding the need for resampling. Accordingly, the resulting system, when operating at 44.1 kHz, incurs under 60 ms of latency and operates 20 times faster than real-time on a standard laptop CPU.

1. INTRODUCTION

Singing voice conversion (SVC) is an audio style transfer application which converts the voice of a sung performance to that of another without changing its underlying content or melody [1]. It can be used for expressive and creative voice manipulations that go beyond conventional effects. Relative to voice conversion applied to speech, SVC has stronger demands on accurate pitch preservation as humans are sensitive to pitch instabilities in singing [2].

SVC has been dominated by deep learning approaches of late. With some exceptions, methods attempt to predict converted acoustic features, and use vocoders to synthesize waveforms from said features [3]. The end-to-end adversarial SVC (EA-SVC) method inverts a learned latent representation with a MelGAN [4] generator, using adversarial training to improve signal plausibility [5]. DiffSVC uses a diffusion model to improve acoustic feature modeling [6]. As state-of-the-art neural vocoders often lack sufficient pitch stability, FastSVC [1] conditions waveform generation on a harmonic excitation signal. Most approaches focus on fidelity improvements, with less attention placed on their deployability as real-time streaming plug-ins that operate seamlessly in conventional audio workflows.

In our own previous works, we have considered different feature representations and end-to-end training mechanisms [2, 3]. Most recently [7], we explored a vocoder-free alternative [8] that was implemented in an end-to-end context using a variation of a WORLD feature representation [9]. In this case, we achieved SVC by processing the input signal rather than synthesizing a new one based on it, effectively guaranteeing its relative pitch contour.

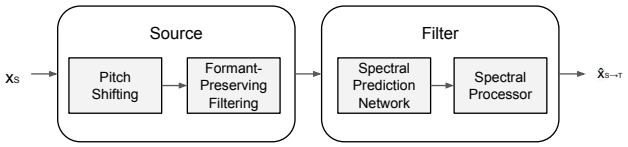


Figure 1: *Proposed RT-SVC system block diagram.*

In this work, we propose a real-time SVC (RT-SVC) approach and implement it as a real-time plug-in. Drawing from [7], we model SVC using a source-filter method, combining pitch-shifting techniques with a lightweight, low-latency neural filter model. We highlight design choices to balance fidelity and performance, and means of scaling our methods to different sampling rates. Our paper is organized as follows: Section 2 describes the proposed method, Section 3 discusses its realization as a real-time plug-in, Section 4 reports experimental findings, and Section 5 draws conclusions.

2. PROPOSED METHOD

2.1. System overview

We consider source and target vocalists S and T, respectively. Our aim is to determine a suitable waveform $x_{S\to T}$ capturing the performance (content) of an input source waveform x_S, while assuming the character of T (style). The conversion task involves two transformations on x_S, as illustrated conceptually in Figure 1. The first stage pitch shifts the input by a constant factor, such that the resulting $x_{S,PS}$ is reflective of the register of the vocalist T. The second stage applies linear time-varying (LTV) filtering to the pitch-shifted result so that the timbre of the result conceivably matches that of the vocalist T. This is modeled using a deep learning model trained in an end-to-end manner over a dataset of recordings of the vocalist T. As stated, the problem naturally lends itself to a source-filter approach.

While many SVC approaches depend on neural vocoders to synthesize new waveforms from inferred representations, we approach the task by processing the input signal. In the context of a real-time system, this offers several advantages, including:

- *Pitch preservation*: Parametric/neural vocoders are prone to pitch errors, and cannot ensure relative pitch input contour preservation in their outputs. Meanwhile, we preserve pitch exactly, barring absolute shifts which we can reliably apply.
- *Reduced complexity*: Removing the need for a vocoder reduces computational footprint and latency in our system.

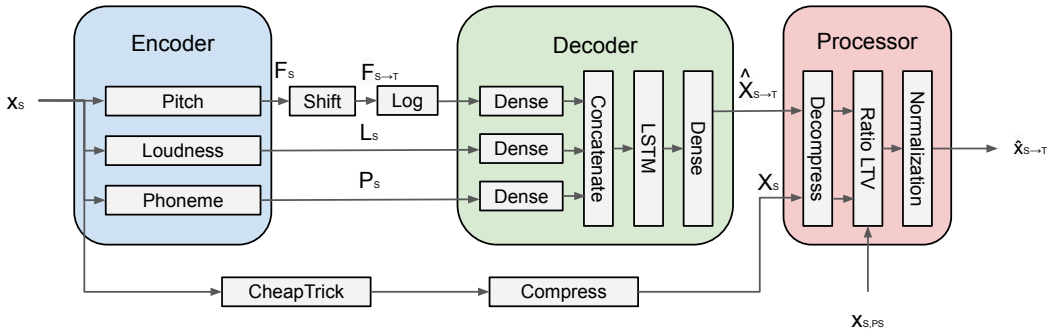


Figure 2: *Fully differentiable end-to-end neural filter model block diagram.*

- *Extension to arbitrary sampling rates*: We can extend our method to sampling rates beyond that used in training, producing wide-band outputs while only needing to model the SVC effect over a perceptually relevant sub-band [10].
- *Voice interpolation*: We can interpolate between source and modeled timbres via a convex combination of acoustic features, with perfect recovery of the source timbre if desired.

2.2. Vocoder-free design with WORLD log Mel spectrograms

Defining $\mathcal{F}$ and $\mathcal{F}^{-1}$ as the forward and inverse short-time Fourier transforms (STFT), respectively, we describe our approach as a spectral process. At a nominal sampling rate $f_{s,0} = 22.05$ kHz, we use hop and frame lengths of $H = 256$ and $N = 1024$ samples, respectively. A signal x has the source-filter decomposition

$$x = \mathcal{F}^{-1}\left[\mathcal{F}(e) \odot \mathcal{F}(h)\right] \tag{1}$$

where e and h denote the time-varying source (i.e. excitation) and filter function of x, respectively. Proper estimation of the spectral envelope $|\mathcal{F}(h)|^2$ leads to extraction of a predominantly flat excitation spectrum. To this end, we leverage the CheapTrick algorithm from WORLD analysis [9], which estimates spectral envelopes $sp = \left|\mathcal{F}(\hat{h})\right|^2$ via a fundamental frequency (f_0) dependent smoothing of signal power spectra.

We seek to determine $\mathcal{F}(e_{S\to T})$ and $\mathcal{F}(h_{S\to T})$ given x_S, from which we can compute $x_{S\to T}$ according to equation (1). To model the excitation spectrum, we apply a pitch shift to x_S to register match it against the target vocalist, resulting in $x_{S,PS,0}$. Robust formant preservation during pitch shifting is a system requirement, and we propose a generic means for providing this for any pitch shifting process through *formant-preserving post-filtering*. Leveraging CheapTrick again, we derive the spectrum of what would be the post-filtered, pitch-shifted signal $x_{S,PS}$ as

$$\mathcal{F}(x_{S,PS}) = \sqrt{\frac{sp_S}{sp_{S,PS,0}}} \odot \mathcal{F}(x_{S,PS,0}) \tag{2}$$

where sp_S and $sp_{S,PS,0}$ are estimates of the spectral envelopes of x_S and $x_{S,PS,0}$, respectively.

To reduce the dimensionality of features ultimately predicted by our deep learning model, we use a compressed representation called the WORLD log Mel spectrogram [7], given by

$$X = \log_{10}(\mathbf{M}\sqrt{sp} + \epsilon) \tag{3}$$

where $\mathbf{M}$ is the Mel basis matrix used to compute an M-band Mel spectrogram from an N-point STFT, and $\epsilon = 10^{-5}$ is used for numerical stability. It is similar to the generalized Mel cepstrum [3], except that it makes for a more obvious differentiable implementation to support end-to-end training. We use $M = 80$ in this work. A decompressed approximation of X is then

$$sp^{\dagger} = \left[\mathbf{M}_0^{\dagger}(10^X - \epsilon)\right]^2 \tag{4}$$

where $\mathbf{M}_0^{\dagger} = \max(\mathbf{M}^{\dagger}, 0)$ and $\mathbf{M}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{M}$. Given $sp_S^{\dagger}$ derived from the source WORLD log Mel spectrogram X_S, the estimated excitation spectrum is given by

$$\mathcal{F}(\hat{e}_{S\to T}) = \sqrt{\frac{1}{sp_S^{\dagger}}} \odot \mathcal{F}(x_{S,PS}) \tag{5}$$

We task a deep learning model (discussed in Section 2.3) to provide estimates for $\hat{X}_{S\to T}$, resulting in $\mathcal{F}(\hat{h}_{S\to T}) = \hat{sp}_{S\to T}^{\dagger}$ via equation 4. Combining into equations (1) and (5), the converted waveform is estimated as

$$\hat{x}_{S\to T} = \mathcal{F}^{-1}\left[\kappa \odot \sqrt{\frac{\hat{sp}_{S\to T}^{\dagger}}{sp_S^{\dagger}}} \odot \sqrt{\frac{sp_S}{sp_{x_{S,PS,0}}}} \odot \mathcal{F}(x_{S,PS,0})\right] \tag{6}$$

where κ is a normalization term computed empirically at each time step to ensure that the modified spectrum L1 norm matches that of $\mathcal{F}(x_S)$. Equation 6 contains four distinct parts: 1) pitch shifting, 2) formant-preserving post-filtering, 3) timbral modification, and 4) normalization. The resulting ratio-based LTV filter is non-negative by design, mitigating potential phase coherence issues.

2.3. Neural filter model

The goal of the neural filter model, outlined in Figure 2, is primarily to infer WORLD log Mel spectrograms $\hat{X}_{S\to T}$ to match the timbre of the target singer while maintaining the content of the source. The LTV filtering outlined in Section 2.2 can be made fully differentiable given access to the various spectral envelopes and waveforms as input, so it is implemented as part of our model in order to enable its end-to-end training [11], yielding $\hat{x}_{S\to T}$.

To perform SVC for any source S, we must create a singer-independent encoding for x_S. Our encoder extracts source loudness L_S deterministically, using the frame-level root-mean-square (RMS) converted to decibels during training (with system hop and frame lengths). We also use the tonality-gated f_0 contour F_S extracted using DIO [9] within WORLD analysis during training. At

inference time, the pitch feature is offset based on the pitch shift that we apply, yielding the target pitch contour $F_{S\to T}$. Lastly, to capture linguistic content, we extract a phonetic posteriorgram P_S [5] from a phoneme classifier. We found it instructive to pre-emphasize the input prior to passing it to the phoneme classifier using a finite impulse response filter of the form

$$y[t] = k_0 x[t] - k_1 x[t-1] \tag{7}$$

with $k_0 = 1.0$ and $k_1 = 0.97$. The classifier then passes 40 Mel frequency cepstral coefficients (MFCCs) extracted from the pre-emphasized signal through a unidirectional recurrent architecture consisting of two long short-term memory (LSTM) layers with 256 units, and a final dense layer yielding a 61-dimensional output vector of phoneme class probabilities at each time step. The network is trained on the TIMIT dataset [12].

The decoder builds upon a lightweight, real-time variant of the architecture in [11], where each multi-layer perceptron (MLP) head uses a single dense layer with 256 units, layer normalization and ReLU nonlinearity. As the initial architecture only considered frequency and loudness features, we add an additional head for P_S. In doing so, we effectively model the way in which the timbre of a phonetic sequence of a target vocalist is varied based on changes in delivery (e.g. when belting a high note loudly). The inputs and outputs of each encoding head are combined and fed through a single 256-unit LSTM layer, followed by a dense layer which outputs a WORLD log Mel spectrogram. The end-to-end audio processor contained within the model filters input audio using the decompressed spectral envelopes, resulting in the model output waveform.

2.4. Training objective

The neural filter model is trained as an autoencoder, and therefore, we have $x_S = x_{S,PS,0} = x_{S,PS} = x_T = x_{S\to T}$, $X_S = X_T = X_{S\to T}$, $F_S = F_T = F_{S\to T}$, etc. during training. In this sense, the model is trained end-to-end, but admittedly, is never exposed to pitch-shifted audio (or any of its associated audio artifacts), as the training objective is merely one of self-reconstruction.

Model training minimizes a combination of conventional negative log likelihood loss terms ensuring good average fidelity and adversarial loss terms promoting plausible system outputs as determined by a discriminator network [4]. To this end, we considered the time-domain multi-scale discriminator architecture in [4] and a single-scale version of the spectral domain architecture as in [13]. Also similar to [13], we actually observed better performance using a spectral domain discriminator for the task. Given a suitably trained and frozen (phonetic) encoder, the full objective function for the neural filter model is

$$\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{MSL} + \mu\mathcal{L}_G \tag{8}$$

where $\mathcal{L}_{MSE}$ is the mean squared error (MSE) defined on WORLD Mel spectrograms (i.e. the decoder outputs), $\mathcal{L}_{MSL}$ is the end-to-end multi-spectrogram loss (MSL) [11], $\mathcal{L}_G$ is the end-to-end adversarial generator hinge loss, and μ is a hyperparameter set to 0.5 in this work. As in [14], we noticed that the usual deep feature matching loss associated with [4] tended to slow down convergence, and that $\mathcal{L}_{MSE} + \mathcal{L}_{MSL}$ was sufficient for stabilizing adversarial training. The discriminator is trained to minimize its corresponding end-to-end discriminator hinge loss.

3. REAL-TIME PLUG-IN IMPLEMENTATION

We have implemented our system as a real-time plug-in in C++. Here, we outline practical considerations for such a realization.

3.1. Streaming feature extraction

We approximate the frame-based loudness feature used during training, computing RMS in a zero-latency fashion via a 1-pole infinite impulse response (IIR) filter with a time constant equal to the system stride. We replace DIO with our proprietary low-latency pitch detection algorithm, and use our implementation of the Lent algorithm [15] as a real-time pitch shifter. Loudness and pitch features are sampled according to the system stride so that they are aligned to their frame-based counterparts. Lastly, we refactor the CheapTrick C++ implementation [9] to handle buffered audio streams.

3.2. Extension to higher sampling rates

We design the plug-in to extend processing to a sampling rate $f_s = G \cdot f_{s,0}$ ($G \geq 1$) without the need for resampling. We consider how our feature representations vary as a function of G, and devise schemes to roughly neutralize this effect, so as not to create a large input feature mismatch from training. Our pitch detection/shifting algorithms and the loudness and pitch feature computations in Section 3.1 are sample-rate agnostic by construction. STFT frame and hop lengths scale with G (while ensuring N to be an even power of 2), and as most fast Fourier transform (FFT) implementations are unnormalized, we carefully scale power and magnitude spectra (as used in CheapTrick or to generate MFCCs) by $1/G^2$ and $1/G$, respectively. We construct Mel bases and pseudo-inverse matrices at f_s while maintaining their respective lower and upper frequency bounds at $f_{s,0}$. We roughly preserve the response of the pre-emphasis filter in equation (7) using generalized filter coefficients $k_0 = G$, $k_1 = G - 1 + k_{1,0}$, $k_{1,0} = 0.97$.

Lastly, we found that we only need to model SVC up to around 10 kHz (arguably lower) to yield a convincing effect, and that we can safely extend the LTV filter gain derived near this boundary to frequencies above it. This effectively amounts to injecting a properly scaled version of $\mathcal{F}(x_{S,PS})$ at frequencies above 10 kHz. This way, we produce wide-band SVC results, even though our network is only trained to model a smaller bandwidth arguably considered too narrow for music production purposes.

3.3. Model export

According to its gains in [16], we use TFLite as a real-time deep learning inference engine. To do so, we recreate encoder/decoder architectures in TensorFlow, explicitly defining a single time step of acoustic features as model input/output. We add LSTM state vectors as additional inputs/outputs so that we can propagate them between time steps and reset them as needed in the plugin. Lastly, we convert the resulting TensorFlow model to the TFLite format.

3.4. Plug-in performance

We incur about 45 ms of latency due to windowing in CheapTrick and the forward/inverse STFTs, and less than 15 ms of residual latency due to pitch detection/shifting, resulting in a total plug-in latency of just under 60 ms. The plug-in runs 20 times faster than real-time on a standard laptop CPU and can be launched from a conventional digital audio workstation.

Table 1: *Quantitative and qualitative model comparisons.*

Model	L_1	# Params.	Pitch shifter	MOS
Offline [7]	**0.042**	25.3M	Lent	3.99
			Phase Vocoder	**4.52**
RT-SVC	0.135	**2.19M**	Lent	3.37
			Phase Vocoder	3.89
RT-SVC$_{GAN}$	0.225	**2.19M**	Lent	*3.69*
			Phase Vocoder	4.20

4. EXPERIMENTAL RESULTS

We exemplify our methods using an internal collection of voice data. The dataset features recordings from 15 different singers, with approximately 2 hours of data for each singer. We consider 3 SVC models for each vocalist: our offline model used in [7], as well as RT-SVC models trained with and without adversarial loss terms (RT-SVC and RT-SVC$_{GAN}$, respectively). All systems are trained at 22.05 kHz, using 2-second audio clips and a batch size of 4. We use the Adam optimizer with a learning rate of 10^{-4} and train for 500,000 training steps. When leveraging adversarial training, we train the generator for 50,000 steps before training the discriminator. For subjective listening, we refer readers to our demo website at https://sites.google.com/izotope.com/rtsvc-demo.

Table 1 reports L_1 reconstruction error of log Mel spectrograms computed between inferred waveforms and their targets and mean opinion scores (MOS) collected from participants within our organization, across models trained on one of said singers. For the latter, participants were asked to rate examples from 0 to 100, and responses were rescaled to the conventional 1 to 5 scale. Overall, 11 people with proficient listening and varied musical experience evaluated our models. Indeed, our offline model outperforms RT-SVC models in terms of fidelity quantitatively and qualitatively. It tends to reconstruct prolonged vowels more consistently, and generally exhibits less leakage of the input speaker identity. For added perspective, we note that regardless of the neural filter model, the use of our proprietary high-latency phase vocoder pitch shifting algorithm can noticeably and universally affect fidelity as well, experiencing fewer audio artifacts across consonants and vowels relative to our Lent implementation. Nonetheless, our real-time model is over 10 times smaller, and when paired with the Lent pitch shifter, is amenable to streaming applications. Lastly, while RT-SVC$_{GAN}$ achieves worse average L_1 performance than RT-SVC, it produces more plausible outputs, as per its higher MOS score (see supplemental figures on our website for details).

5. CONCLUSIONS

We proposed an approach for real-time SVC and implemented it as a streaming plug-in. The method combines pitch shifting of the input signal and timbral transformation provided by a deep learning model. The novelty of the method is that it acts directly on the input signal instead of synthesizing a new waveform, reducing complexity and preserving the pitch of the original signal. As such, the model can extend to sampling rates beyond the nominal rate used by its deep learning model component. In future work, we are interested in improving excitation signal modeling, and specifically, to see if it is possible to inject the pitch shifting algorithm "in-the-loop" during training to improve fidelity. We would also like to improve acoustic feature/filter modeling for the real-time case, potentially leveraging recent advances on integrating streaming convolutional layers and related architectures [17].

6. REFERENCES

[1] S. Liu et al., "FastSVC: Fast cross-domain singing voice conversion with feature-wise linear modulation," in *IEEE Int. Conf. on Mul. and Ex. (ICME)*, 2021, pp. 1–6.

[2] S. Nercessian, "Zero-shot singing voice conversion," in *Int. Soc. for Mus. Inf. Ret. Conf. (ISMIR)*, 2020, p. 70–76.

[3] S. Nercessian, "End-to-end zero-shot voice conversion using a DDSP vocoder," in *IEEE Work. on App. of Sig. Proc. to Aud. and Ac. (WASPAA)*, 2021, pp. 1–5.

[4] K. Kumar et al., "MelGAN: Generative adversarial networks for conditional waveform synthesis," in *Adv. in Neu. Inf. Proc. Sys. (NeurIPS)*, 2019, vol. 32.

[5] H. Guo et al., "Phonetic posteriorgrams based many-to-many singing voice conversion via adversarial training," *arXiv:2012.01837*, 2020.

[6] S. Liu, Y. Cao, D. Su, and H. Meng, "DiffSVC: A diffusion probabilistic model for singing voice conversion," *arXiv:2105.13871*, 2021.

[7] S. Nercessian, "Differentiable WORLD synthesizer-based neural vocoder with application to end-to-end audio style transfer," in *154th Aud. Eng. Soc. Conv. (AES)*, 2023.

[8] J.W. Kim, H.Y. Jung, and M. Lee, "Vocoder-free end-to-end voice conversion with transformer network," in *IEEE Int. Joint Conf. on Neu. Net. (IJCNN)*, 2020, pp. 1–8.

[9] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. on Inf. and Sys.*, vol. E99-D, no. 7, pp. 1877–1884, 2016.

[10] T. Saeki, Y. Saito, S. Takamichi, and H. Saruwatari, "Real-time, full-band, online DNN-based voice conversion system using a single CPU," in *Interspeech*, 2020, pp. 1021–1022.

[11] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Int. Conf. on Learn. Rep. (ICLR)*, 2020, pp. 26–30.

[12] J. S. Garapolo et al., *TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1*, Linguistic Data Consortium, Philadelphia, 1993.

[13] A. Wright, V. Välimäki, and L. Juvela, "Adversarial guitar amplifier modelling with unpaired data," in *IEEE Int. Conf. on Ac., Speech and Sig. Proc. (ICASSP)*, 2023.

[14] R. Yamamoto, E. Song, , and Jae-Min Kim, "Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *IEEE Int. Conf. on Ac., Speech and Sig. Proc. (ICASSP)*, 2020, p. 6199–6203.

[15] K. Lent, "An efficient method for pitch shifting digitally sampled sounds," *Computer Music Journal*, vol. 13, no. 4, pp. 65–71, 1989.

[16] D. Stefani, S. Peroni, and L. Turchet, "A comparison of deep learning inference engines for embedded real-time audio classification," in *Int. Conf. on Dig. Aud. Eff. (DAFx)*, 2022, pp. 256–263.

[17] A. Caillon and P. Esling, "Streamable neural audio synthesis with non-causal convolutions," in *Int. Conf. on Dig. Aud. Eff. (DAFx)*, 2022, pp. 320–327.

EXPRESSIVE PIANO PERFORMANCE RENDERING FROM UNPAIRED DATA

Lenny Renault, Rémi Mignot and Axel Roebel

UMR9912 - STMS
Ircam, Sorbonne Université, CNRS, Ministère de la Culture
Paris, France
lenny.renault@ircam.fr | remi.mignot@ircam.fr | axel.roebel@ircam.fr

ABSTRACT

Recent advances in data-driven expressive performance rendering have enabled automatic models to reproduce the characteristics and the variability of human performances of musical compositions. However, these models need to be trained with aligned pairs of scores and performances and they rely notably on score-specific markings, which limits their scope of application. This work tackles the piano performance rendering task in a low-informed setting by only considering the score note information and without aligned data. The proposed model relies on an adversarial training where the basic score notes properties are modified in order to reproduce the expressive qualities contained in a dataset of real performances. First results for unaligned score-to-performance rendering are presented through a conducted listening test. While the interpretation quality is not on par with highly-supervised methods and human renditions, our method shows promising results for transferring realistic expressivity into scores.

1. INTRODUCTION

Performance rendering is the task of imbuing a music score with expressive features as if a musician performed the score in a way to bring out emotional qualities. To get an expressive rendition of the music, performers have the liberty to shape sound parameters that are not explicitly described by the written score [1]: for piano pieces, musicians make an interpretation of the score by mainly reshaping the timing, articulation and nuance of the notes. An automated system that can reproduce such a complex and artistic behavior can find its usage in assisting composers for obtaining musical renditions of their pieces.

Previous works for the task used data-driven methods to predict performance features that enhance the score note indications [2, 3, 1, 4]. More recently, *Variational Auto-Encoders* (VAE) conditioned on score features have proven to be successful at modeling the diversity in performance expressivity, as several renditions of the same piece are conceivable [5, 6, 7, 8]. The performance features are defined as the difference in timing, articulation, and velocity of the played notes compared to the exact rendition of the score [9]. However, obtaining such features requires the collection of *Musical Instrument Digital Interface* (MIDI) performances with their associated digital scores and to align them at note-level [10, 11]. These required matching and alignment steps limit the amount of data available for training [12] and the application of the models to piano music, where performance MIDI data can be collected more easily [13]. Also, most of these works are highly-informed as they take different markings in the digital scores into account for guiding the expressive rendering, such as rests, beat information, hand part, position in the measure, key and time signatures, articulation and ornament markings, slurs or beams. This reliance on markings specific to the sheet music format hinders the usage of these models in modern music production frameworks (DAW, sequencers) where MIDI data are directly manipulated without using such markings.

Concurrently, *Generative Adversarial Networks* (GAN) have been successfully applied for various tasks transferring data from one domain to another without aligned pairs, such as image-to-image translation [14], audio timbre matching [15] or music genre transfer [16]. In the light of such results, this work attempts to address expressive performance rendering as a domain transfer task, by transforming MIDI scores into human-like performances without supervision on the performance features and reliance on score markings. To this end, an adversarial approach is employed to map the outputs of a low-informed performance rendering model to the distribution of human performances, without providing matching pairs of scores and performances. Trained on publicly available datasets, the proposed method and its experiments are presented here, including an early subjective evaluation.

The experiments show promising results for the method as it can infer expressive qualities into scores, although not with the same amount of naturalness as in performances rendered by real pianists and by a highly-informed supervised baseline. Accompanying this paper, audio samples are provided online [1].

2. PROPOSED APPROACH

The proposed approach, illustrated in Figure 1, is composed of a performance rendering model G that takes a score X as input and produces an expressive interpretation $\tilde{X}$. The rendered performances are fed into a discriminator D, among performances Y from a dataset of recorded human performances. The performance rendering model and the discriminator have opposed objectives, as the discriminator D aims to differentiate the real performances from the ones rendered by the model G, while the latter tries to produce performances indistinguishable from the real ones.

2.1. Data formatting

Both the scores X and real performances Y are encoded as sequences of N notes with the minimal amount of features needed for describing them:

$$\mathbf{X} = \{x_n\}_{n \leq N} = \{p_n, o_n, d_n, v_n\}_{n \leq N}. \quad (1)$$

[1] http://renault.gitlab-pages.ircam.fr/dafx23

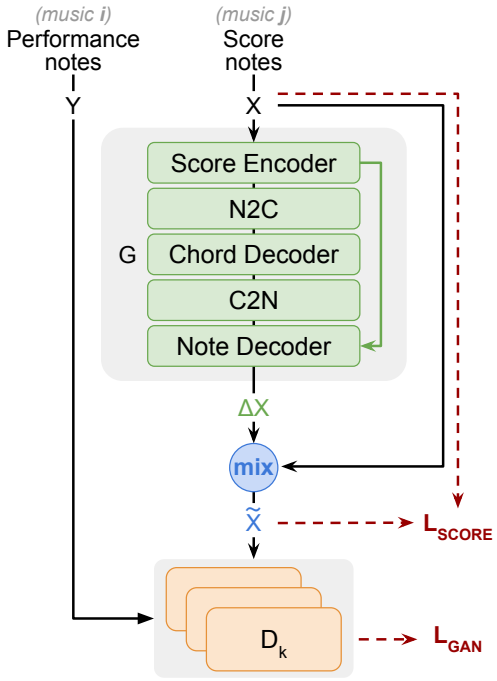


Figure 1: *Training pipeline for the proposed model: the mix function modifies the score X with features output by the performance rendering model G (in green), in order to deceive the discriminators D_k (in orange). During training, the unaligned score X and performance Y are drawn at random from their respective sets.*

The notes are ordered by their absolute onset time: for the n-th note, p_n is its normalized MIDI pitch, o_n its delta-time with the previous note onset, or relative *inter-onset-interval* (IOI), capped at 4s, d_n its duration in absolute time and v_n its normalized MIDI velocity.

2.2. Performance rendering model

The performance rendering model G predicts modifying features $\Delta X = G(X)$ from the score note features in order to modify them into performance-like note features $\tilde{X}$ through the mix function:

$$\begin{aligned}\tilde{X} &= \text{mix}(X, G(X)) \\ &= \{p_n, o_n + \delta o_n, d_n \times \delta d_n, v_n \times \delta v_n\}_{n \leq N},\end{aligned} \tag{2}$$

with δo_n the micro-onset timing, δd_n the articulation and δv_n the expressive velocity of the n-th score note.

These modifying features are obtained by first processing the note-wise score features with a convolutional Score Encoder. Then, the same hierarchical modeling from [5] is applied: the note-wise features are merged into chord-wise features, which enables a more coherent modeling of the full sequence. This note-to-chord operation, or $N2C$, is performed by average pooling the features of simultaneous notes into a common chord-wise feature. The inverse operation $C2N$ can later convert chord-wise features into note-wise features by duplicating the chord feature for each of its notes. On the contrary of hierarchical strategies employed in other works [7, 8], the note-to-chord alignment matrix required for $N2C$ and $C2N$ can be directly extracted from our low-informed MIDI data representation, using the sequence of relative IOI $\{o_n\}_{n \leq N}$. Further implementation details on the $N2C$ and $C2N$ operations can be found in [5].

Before returning to the note-granularity, the chord-wise features are further processed by a Chord Decoder, which is a *Convolutional Recurrent Neural Network* (CRNN) with a bidirectional *Gated Recurrent Unit* (GRU) layer. Finally, fine-grained adjustments at note-level are made with the Note Decoder and a skip connection from the note-wise score encoding. The final micro-onset timing δo_n is obtained through a linear activation function, while the articulation δd_n and the expressive velocity δv_n are mapped to $[0.25, 4]$ with a scaled sigmoid function.

2.3. Discriminator

Taking inspiration from speech processing using discriminators with a multi-scale architecture [17], we use $k = 3$ discriminators D_k with identical architectures, mirrored from the performance rendering model, with the exceptions of the $N2C$ and $C2N$ operations, as chords in real performances are not as easily defined as in scores. Each discriminator is fed with a downsampled sequence of (real or rendered) performance notes by average pooling with sizes $\{1, 3, 9\}$. Discriminators with longer pool sizes look at features at higher levels in the performances and thus, can help transferring such knowledge and long-term coherence to the performance rendering model G. To stabilize the GAN training, gaussian noise is added to the inputs of the discriminators, as in [16].

2.4. Loss functions

The least-square variant of the GAN objective (LSGAN) is used to train the discriminators and the performance rendering model. Their respective loss functions L_{D_k} and $L_{G,gan}$ are defined as:

$$\begin{aligned}L_{D_k} &= \mathop{\mathbb{E}}_{Y \sim p_{perf}} \left[\|D_k(Y) - 1\|_2\right] + \mathop{\mathbb{E}}_{X \sim p_{score}} \left[\|D_k(G(X))\|_2\right], \\ L_{G,gan} &= \mathop{\mathbb{E}}_{X \sim p_{score}} \left[\sum_{k=1,2,3} \|D_k(G(X)) - 1\|_2\right].\end{aligned} \tag{3}$$

We have observed that the instability of the vanilla adversarial training may lead the performance rendering model to displace the notes in extreme values, causing the original piece to be unrecognizable. To ensure that the performances remain fairly close to their scores, an additional regularization term L_{score} is added:

$$L_{score}(X) = \boldsymbol{\lambda}_{score} \left\| \frac{G(X) - X}{X} \right\|_2, \tag{4}$$

with $\boldsymbol{\lambda}_{score}$ a fixed vector weighting how much each performance component (timing, articulation, velocity) can deviate from the score indication. Here, $\boldsymbol{\lambda}_{score} = \{1, 1, 0.1\}$.

The total loss for the performance rendering model G is:

$$L_G(X) = \lambda_{gan} L_{G,gan}(X) + L_{score}(X), \tag{5}$$

with λ_{gan} the balance between the GAN objective and the score regularization loss. This balance is decisive for the final behavior of G since the two loss components have opposite influences on its training: L_{score} refrains G from modifying the scores while $L_{G,gan}$ encourages exploring different interpretations in order to deceive the discriminator. In our experiments, $\lambda_{gan} = 2$.

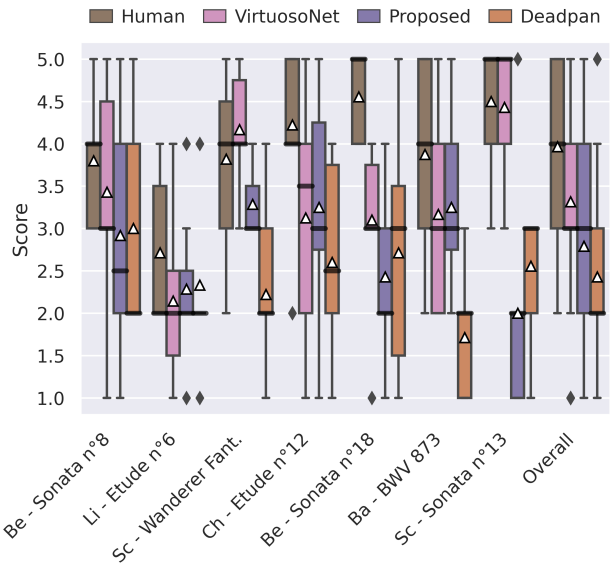


Figure 2: *Box-plot of the* Mean Opinion Score *(MOS) of the different performance rendering methods (overall and piece-wise). The thickened bars indicate the median values while the white triangles indicates the mean values. The composers are Bach (Ba), Beethoven (Be), Chopin (Ch), Liszt (Li) and Schubert (Sc).*

3. EXPERIMENTS

3.1. Score and performance datasets

The proposed approach was trained and evaluated using the scores from the ASAP dataset [12] and all performances from the MAESTRO dataset (v3.0.0) [13], which are both publicly available. The human performances from MAESTRO were recorded in MIDI format using Yamaha Disklaviers. The ASAP dataset has notably matched a set of these performances with their original scores at note-level, and has thus been used to some extent in previous performance rendering works [5]. However, since the proposed method does not require aligned scores and performance, the entirety of both datasets can be used, which amounts for 962 training performances, 137 validation performances, 107 training scores, 15 validation scores and 35 test scores (following the train-validation-test split of [18]).

The velocity indications were kept from the ASAP scores in MIDI format, which can either be constant throughout the piece or mapped from the score nuances and markings using simple rules. The scores and performances are split into segments of 128 consecutive notes, with random pitch shifting during training by ± 7 semi-tones, as in [6]. Validation data is used to monitor and avoid potential over-fitting of the performance rendering model by reproducing the training performances from their corresponding scores.

3.2. Early subjective evaluation

A listening test has been conducted to evaluate the interpretation quality of the performances rendered by our model. 7 scores from the ASAP test subset were selected, covering 5 different composers. 4 performances were generated by different methods for each score: a corresponding human performance from the MAESTRO dataset (**Human**), the direct export of the MIDI score (**Deadpan**), a rendition by our approach (**Proposed**) and a rendition from the graph-based variant of **VirtuosoNet** [8], a highly-informed model using score markings in MusicXML format and is trained with an private dataset of 226 scores matched and aligned with MAESTRO performances, which is larger than ASAP. The first 20s of each performance were synthesized using the Arturia Piano V3 software [2], a physical-based piano synthesizer. 19 professional audio and piano players were asked to rate the naturalness of the presented performances, using a 5-point Likert scale (from 1 - Bad, to 5 - Excellent). Each trial randomly presented 3 different performances from each method. Results are reported in Figure 2.

The Holm-Bonferroni corrected two-sided Mann-Whitney U tests indicate a statistical difference at $\alpha = 0.05$ between the Human rendition and each other methods, and between VirtuosoNet and Deadpan. The overall results show that the proposed approach does enhance the scores with expressive features in comparison to the raw rendition of the piece, but still not with the same amount of naturalness as actual pianists and the highly-informed VirtuosoNet. This was to be expected as our proposed unsupervised training task without score markings is harder than the training objectives of VirtuosoNet, for about the same quantity of training data. By examining the ratings piece-wise, one can notice the poorer renditions of the proposed method for slower tracks (Schubert's 13th Sonata and Beethoven's 18th Sonata). This may suggest that the model has a mode collapse on faster paced music and that it applies similar modifying features on every tracks, which renders inappropriate performances for slower musical pieces.

4. FUTURE WORK

As suggested by the subjective evaluation, the model lacks in understanding the musical content of a score and can apply inappropriate performance features. Also, on the contrary of the most recent supervised performance rendering methods [8, 5, 6], the model does not allow for external controls (tempo, articulation, nuance) on the rendering process. Both issues can be addressed by organizing the performances into sub-domains with either domain labels (such as composer or genre) or with extracted performance features (note density, statistics on durations and velocities)[19].

Moreover, the present work only focuses on classical piano music to be comparable with previous supervised approaches, but without reliance on training pairs, the approach can be extended to render symbolic performances for other genres and instruments.

Finally, GAN enable unsupervised cross-modal domain transfer where the target domain can be in a different modality from the source domain. By including a differentiable sound synthesizer [20] after the performance rendering model and using a audio-based performance discriminator, the model could potentially render scores with expressive features by learning from performances in the audio domain instead of MIDI.

5. CONCLUSIONS

This work presents a performance rendering model for converting piano scores into expressive performances, without supervised training on performance features nor relying on sheet music markings. Using a performance discriminator, the model reshapes the

[2] `https://www.arturia.com/products/software-instruments/piano-v/overview`

basic score note properties into a sequence of expressive notes. Trained on publicly available datasets of scores and performances, the approach shows expressive qualities in the performance renditions compared to the plain score, although not with the same quality as a fully supervised approach, according to a conducted listening test. Still, by removing the reliance on training with paired data and on score markings, the approach can be further used in broader settings with music in different modalities and genres.

6. ACKNOWLEDGMENTS

This work was supported by European Union's Horizon 2020 research and innovation programme under grant number 951911 - AI4Media.

7. REFERENCES

[1] Carlos E Cancino-Chacón, Maarten Grachten, Werner Goebl, and Gerhard Widmer, "Computational models of expressive music performance: A comprehensive and critical review," *Frontiers in Digital Humanities*, vol. 5, 2018.

[2] Iman Malik and Carl Henrik Ek, "Neural translation of musical style," in *Workshop on Machine Learning for Creativity and Design, Neural Information Processing Systems (NIPS)*, Long Beach, California, USA, Dec. 8, 2017.

[3] Fábio José Muneratti Ortega et al., *A machine learning approach to computer modeling of musical expression for performance learning and practice*, Ph.D. thesis, Universitat Pompeu Fabra, 2022.

[4] Hao-Wen Dong, Cong Zhou, Taylor Berg-Kirkpatrick, and Julian McAuley, "Deep performer: Score-to-audio music performance synthesis," in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 951–955.

[5] Seungyeon Rhyu, Sarah Kim, and Kyogu Lee, "Sketching the expression: Flexible rendering of expressive piano performance with self-supervised learning," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Bengaluru, India, Dec. 4-8, 2022.

[6] Akira Maezawa, Kazuhiko Yamamoto, and Takuya Fujishima, "Rendering music performance with interpretation variations using conditional variational RNN," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, Nov. 4-8, 2019, pp. 855–861.

[7] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, Kyogu Lee, and Juhan Nam, "VirtuosoNet: A Hierarchical RNN-based System for Modeling Expressive Piano Performance," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 908–915.

[8] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam, "Graph neural network for music score data and modeling expressive piano performance," in *Proc. of the International Conference on Machine Learning (ICML)*, Long Beach, California, USA, June. 9-15, 2019, PMLR, pp. 3060–3070.

[9] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam, "Score and performance features for rendering expressive music performances," in *Proc. of the Music Encoding Conference*, Vienna, Austria, May 2019.

[10] Francesco Foscarin, Emmanouil Karystinaios, Silvan David Peter, Carlos Cancino-Chacón, Maarten Grachten, and Gerhard Widmer, "The match file format: Encoding alignments between scores and performances," in *Proc. of the Music Encoding Conference*, Halifax, Canada, May. 19-22, 2022.

[11] Eita Nakamura, Kazuyoshi Yoshii, and Haruhiro Katayose, "Performance error detection and post-processing for fast and accurate symbolic music alignment," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, Oct. 2017, pp. 347–353.

[12] Francesco Foscarin, Andrew Mcleod, Philippe Rigaux, Florent Jacquemard, and Masahiko Sakai, "ASAP: a dataset of aligned scores and performances for piano transcription," in *Proc. of the International Society for Music Information Retrieval (ISMIR)*, Montreal / Virtual, Canada, Oct. 2020.

[13] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proc. of the International Conference on Learning Representations (ICLR)*, New Orleans, Louisiana, USA, May 2019, OpenReview.net.

[14] Yingxue Pang, Jianxin Lin, Tao Qin, and Zhibo Chen, "Image-to-image translation: Methods and applications," *IEEE Transactions on Multimedia*, vol. 24, pp. 3859–3881, 2022.

[15] Alec Wright, Vesa Välimäki, and Lauri Juvela, "Adversarial guitar amplifier modelling with unpaired data," in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece, June. 4-10, 2023, IEEE.

[16] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Sumu Zhao, "Symbolic music genre transfer with CycleGAN," in *International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2018, pp. 786–793.

[17] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems (NIPS)*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. 2019, vol. 32, Curran Associates, Inc.

[18] Lele Liu, Veronica Morfi, and Emmanouil Benetos, "AC-PAS: a dataset of aligned classical piano audio and scores for audio-to-score transcription," in *Late-Breaking Demos of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[19] Kristy Choi, Curtis Hawthorne, Ian Simon, Monica Dinculescu, and Jesse Engel, "Encoding musical style with transformer autoencoders," in *Proc. of the International Conference on Machine Learning (ICML)*, Jul. 13–18, 2020, pp. 1899–1908.

[20] Lenny Renault, Rémi Mignot, and Axel Roebel, "Differentiable piano model for midi-to-audio performance synthesis," in *Proc. of the International Conference on Digital Audio Effects (DAFx)*, Vienna, Austria, Sep. 2022, pp. 232–239.

INTERPRETABLE TIMBRE SYNTHESIS USING VARIATIONAL AUTOENCODERS REGULARIZED ON TIMBRE DESCRIPTORS

Anastasia Natsiou, Luca Longo, and Seán O'Leary

Technological University Dublin
Dublin, Ireland
{anastasia.natsiou, luca.longo, sean.oleary} @tudublin.ie

ABSTRACT

Controllable timbre synthesis has been a subject of research for several decades, and deep neural networks have been the most successful in this area. Deep generative models such as Variational Autoencoders (VAEs) have the ability to generate a high-level representation of audio while providing a structured latent space. Despite their advantages, the interpretability of these latent spaces in terms of human perception is often limited. To address this limitation and enhance the control over timbre generation, we propose a regularized VAE-based latent space that incorporates timbre descriptors. Moreover, we suggest a more concise representation of sound by utilizing its harmonic content, in order to minimize the dimensionality of the latent space.

1. INTRODUCTION

The emergence of deep generative models has contributed to the development of natural and expressive music synthesizers [1, 2]. One type of generative model, Variational Autoencoders (VAEs), can create a compact representation based on the distribution of the input data and this representation forms a *latent space* [3]. In this space, samples that are similar to each other are positioned closer. However, this proximity may not always align with human perception of similarity. Explainable artificial intelligence (XAI) has been employed in the past to regularize the latent space of generative models [4, 5]. A regularized space can offer interpretability of the latent space and control of the synthesis process to generate sound with desired characteristics.

In the context of music, the perception of different instruments can be characterized by a multidimensional space called *timbre space*. Timbre space is created by asking listeners to rate the dissimilarity between different instruments [6, 7]. In order to utilize this information effectively, a recent approach in sound synthesis has been proposed to incorporate timbre space into the regularization of VAEs [8]. This enables the model to generate sounds based on specific instrument timbres. While utilizing a timbre space has various advantages for generative models, creating a timbre space is not always feasible. The addition of a new instrument would necessitate new listening tests and analyses.

Timbre descriptors are mathematical or statistical functions designed to capture various aspects of human perception of sound. They were originally designed by studying the listening tests and using the information obtained to associate mathematical formulas with timbre space. Several research studies have found that spectral centroid and attack time are the key audio features that are crucial for describing the timbre of musical instruments [9]. Our goal in this study is to utilize the timbre descriptors, particularly the spectral centroid and attack time, to construct a latent space that aligns with human perception. To maximize the compression of the latent space, we developed a novel input representation with lower dimensionality than spectrograms that focuses on the harmonic content of the instruments. The rest of the manuscript is divided as follows. Section 2 provides a literature review on the regularization of VAEs for sound synthesis. Details on the proposed representation along with VAEs and timbre descriptors are described in Section 3. Finally, Section 4 provides information on the experimental setup and Section 5 demonstrates the results.

2. RELATED WORK

Recent research has demonstrated the capability of unsupervised models to acquire invertible audio representations through the use of autoencoders [10]. However, autoencoders have certain limitations that prevent them from creating a coherent and comprehensible latent space, which, in turn, limits their ability to produce audio with specific characteristics. Variational autoencoders (VAEs) overcome this limitation by incorporating a Gaussian distribution into the latent space, which encourages local smoothness and provides interpretability of the latent variables [11]. In numerous applications, however, this approach may not be sufficient.

To achieve additional disentanglement of the latent space, various approaches have been proposed. Autocoder [12] is a simplified VAE trained on audio samples with specific attributes. The main objective of the Autocoder is to produce outputs that fit within the distribution of the training data. Luo et al. [13] proposed a network with two encoders and one decoder to address the disentanglement of pitch and timbre in audio signals. The first encoder learns the pitch and the second learns the timbre, while the decoder reconstructs the original audio signal from the concatenation of the learned pitch and timbre representations. The network can be conditioned on the two categorical variables separately because pitch and timbre are independent of each other. Both of these studies used mel-spectrograms as the input data representation for their models.

An alternative method for creating a latent space that focuses on specific attributes was deployed in [14]. Instead of relying solely on the original input data, this method incorporates an additional representation for chords to help shape the latent space. The model was trained on 32 one-hot vectors of MIDI pitches that represented the rhythm for an analogy-making task. The VAE used Gated Recurrent Units (GRUs) conditioned on chromagrams that represented chords. The attributes of the latent space were implicitly formed by incorporating information about melody and chords

in the loss function. However, other models took a different approach and explicitly designed the latent space by enforcing specific attributes to represent distinct characteristics of the sound. In [4] the latent space consists of 256 dimensions, where the first four dimensions are specifically associated with rhythmic complexity, note range, note density, and average interval jump. The regularization is incorporated into the VAE as a loss function to the training objective. To achieve this, a musical metric value is computed for each item in a mini-batch for each attribute. The distance between each item's metric value and all other items' metric values is then calculated, resulting in a distance matrix. The distance matrix is used for the computation of the mean square error.

Our method follows an approach that is heavily influenced by [8]. Esling et al. introduced an additional regularization loss to form a latent space based on human perception. The latent attributes were generated to satisfy the perceptual similarity ratings of listeners. The architecture of their network consists of a VAE that includes three dense layers and produces a latent space of 64 dimensions. The neural network architecture successfully captured a continuous and generalized timbre representation for a wide range of musical instruments. In our work, we aim for a regularized latent space that aligns with human perception using audio descriptors that are capable to represent timbre.

3. PROPOSED FRAMEWORK

3.1. Variational Autoencoders

Variational Autoencoders (VAEs) [3] are deep generative models with the ability to map high-dimensional observed data $x \in \mathbb{R}^d$ (such as audio samples or spectrograms) to a latent space $z \in \mathbb{R}^e$ with $d > e$. VAEs consist of an encoder and a decoder network. The latent variable is created by the encoder that uses a distribution $q_\theta(z|x)$ to approximate $p_\theta(z|x)$. The decoder then attempts to reconstruct the input data by approximating the distribution $p_\theta(x|z)$. The encoder and decoder are trained together to parametrize $_\theta$. To elaborate further, the encoder generates the mean μ_M and the covariance σ_M to represent the Gaussian distribution function $N(z; \mu_M, \sigma_M^2 I)$ in a latent space of M dimensions. The main goal of the network is to maximize the evidence lower bound by minimizing the Kullback-Leibler (KL) divergence between the distribution function $q(z|x)$ and the prior distribution function $p(z)$. The loss function of VAEs is represented in Eq. 1.

$$\mathbb{E}[\log p(x|z)] - KL(q(z|x) \parallel p(z)) \leq \log p(x) \tag{1}$$

The initial component of the equation evaluates the degree of similarity between the original data and the reconstructed data. The second component assesses the dissimilarity between the approximated posterior distribution $q(z|x)$ and the prior distribution $p(z)$. By including the KL divergence term, the learned posterior distribution is pushed towards the prior distribution, thereby promoting the regularization of the learned latent space representation.

3.2. Proposed Representation

In this study, we introduce a novel audio representation for capturing monophonic and harmonic sounds of musical instruments based on acoustic features. The aim of this approach is to generate a concise representation that can enhance the efficacy of deep neural networks. In order to achieve this goal, we are based on the observation that monophonic, and harmonic sounds can be represented by their fundamental frequency, the first seven harmonics, and the energy of the higher bands. A method for synthesizing from a representation like the one described can be accomplished by employing a sinusoidal model, similar to the approach described in the work of [15].

In this work, we estimate the fundamental frequency using a pre-trained model of CREPE [16], and then we calculate the logarithm of the amplitudes of the first seven harmonics. The harmonics can be estimated as the integer multiples of the fundamental frequency:

$$f_n = nf_0 \tag{2}$$

where the variable n denotes the specific harmonic number for each frame i of the sound, where $n \in [1, 7]$. The first seven harmonics are selected as they are deemed to contain the most perceptually significant aspects of a note, thereby providing essential information on the spectral shape of the acoustic signal. The remaining spectral information is represented by the energy of the higher bands. The higher spectral content is divided into 4 bands using the Equal Rectangular Bandwidth (ERB) [17].

3.3. Timbre Descriptors

Timbre constitutes the quality of sound that is conceptually separated from pitch or loudness. It refers to the unique hearing "color" provided by each instrument or voice. Timbre descriptors are used to quantify and describe the timbre of a sound. The association between timbre descriptors and the perception of sound is strong. Certain timbre descriptors are correlated with specific percepts of sound, such as brightness, dullness, spectral tilt, and tonality [9]. However, many studies have shown that timbre can accurately be described only with spectral centroid and attack time [18].

3.3.1. Spectral centroid

The spectral centroid is a measure of the center of mass of the spectrum of a sound. It is calculated by weighting each frequency in the spectrum by its magnitude and averaging the result as it is demonstrated in Eq. 3.

$$centroid = \frac{\sum_{k=b_1}^{b_2} f_k M(f_k)}{\sum_{k=b_1}^{b_2} M(f_k)} \tag{3}$$

where b_1 and b_2 are the band edges, f_k is the frequency in the bin k, and M(f) is the magnitude of the frequency in the spectrum. The spectral centroid is a useful measure of the timbre of a sound, as it provides information about the distribution of energy across different frequencies in the sound. It is often used to describe the brightness or dullness of a sound [19]. Sounds with a high spectral centroid tend to be brighter and more focused, while sounds with a low spectral centroid tend to be duller and more diffuse.

3.3.2. Attack time

Attack time is a parameter that determines how quickly the amplitude of a sound increases from zero to its maximum level. Attack time is an important factor in shaping the envelope of a sound and has a significant impact on its timbre [20]. The computation of attack time has been under investigation for many years. The most common methods include a fixed threshold. Thresholds can vary

and effective ones can be from 10% to 90% of the maximum value of the energy envelope [21] or from 20% to 80% of the maximum value of the envelope [22]. In this work, we measure attack time as a fixed threshold between 10% to 90%.

4. EXPERIMENTAL SETUP

4.1. Dataset

The NSynth dataset [1], containing a wide range of monophonic notes from various instruments, was utilized in the experiments. The dataset was filtered to include only samples that were harmonic, without variations in fundamental frequency or amplitude. The resulting subsample was composed of 101,911 training samples and 1,324 testing samples of various instruments, including guitar, bass, brass, keyboard, flute, organ, reed, and string, with a pitch range of 80Hz-2100Hz. The audio samples were preprocessed to create a representation that comprises the fundamental frequency, the logarithm of the amplitude of the first 7 harmonics, and 4 ERBs in overlapping segments of audio signals with a window of 690 and a hop size of 172.

4.2. Hyperparameters

The encoder is composed of two 2D convolutional layers with 32 filters, a kernel size of 3, a stride of 2, and the same padding. Two dense layers calculate the mean and variance of the Gaussian distribution. Sampling from the distribution creates a latent space with 14 dimensions. The decoder follows a mirrored architecture of the encoder generating the proposed representation. The ReLU is used as an activation function for the convolutional layers while the softmax function is applied to the output layer to form the generated normalized representation. The network is trained using the ADAM optimizer with an initial learning rate of 0.001 in batches of size 128. The loss function includes a binary cross-entropy reconstruction loss, a regularization loss consisting of a KL-divergence term, and a mean absolute error of the spectral centroid and attack time. The model was trained using the TensorFlow library[2] on a Tesla P100 GPU.

4.3. Evaluation

For the evaluation of the reconstruction capacity of the VAE, we used the Mean Squared Error (MSE) and the Structural SIMilarity (SSIM) index between the original and generated audio representation based on the harmonic content. The latent space was visualized by projecting the high-dimensional features into a two-dimensional space. For the dimensionality reduction, we used the Stochastic Neighbor Embedding (t-SNE) algorithm [23] with PCA initialization and perplexity of 50.

5. RESULTS AND CONCLUSION

Table 1 presents the reconstruction error for the model with and without the regularization of the timbre descriptors. The results showed that convolutional VAEs are able to adequately reconstruct the proposed representation. However, the additional regularization decreased the quality of the generated samples but it provided a clear representation of the latent space. Fig. 1 displays the 14-dimensional regularized latent space projected into a 2-dimensional space. It is evident from the figure that different instrument categories occupy distinct positions in the space, indicating that the model is capable of distinguishing between instrument timbre. However, some instrument categories may be broad and have multiple variations, resulting in clusters in different regions of the space. The obtained results demonstrate that using timbre descriptors to regularize VAEs can lead to a high-level latent representation that is interpretable. This research work illustrates an early investigation of the merging of timbre descriptors with deep generative models.

Table 1: *Reconstruction error*

Model	MSE	SSIM
Without timbre descriptors	0.0432	0.9515
With timbre descriptors	0.0518	0.9414

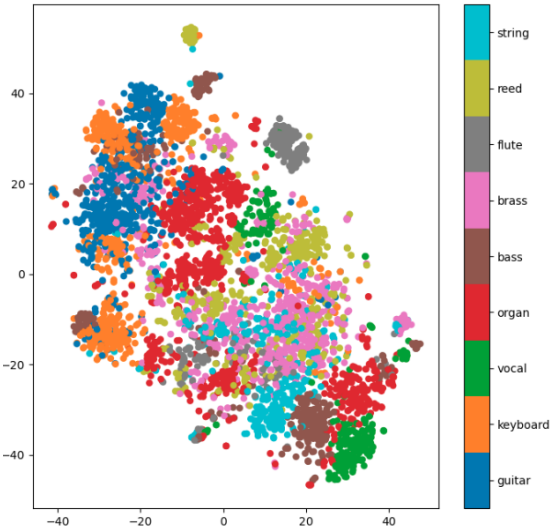


Figure 1: *2D projection of the regularized latent space*

6. ACKNOWLEDGMENTS

This work was funded by Science Foundation Ireland through the SFI Centre for Research Training in Machine Learning (18/CRT/6183)

7. REFERENCES

[1] Jesse Engel, Chenjie Gu, Adam Roberts, et al., "Ddsp: Differentiable digital signal processing," in *International Conference on Learning Representations*, 2020.

[2] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts, "Gansynth: Adversarial neural audio synthesis," in *International Conference on Learning Representations*, 2019.

[3] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[1] https://magenta.tensorflow.org/datasets/nsynth

[2] https://www.tensorflow.org/

[4] Nick Bryan-Kinns, Berker Banar, Corey Ford, Courtney N Reed, Yixiao Zhang, Simon Colton, and Jack Armitage, "Exploring XAI for the arts: Explaining latent space in generative music," 2021.

[5] Anastasia Natsiou, Seán O'Leary, and Luca Longo, "An exploration of the latent space of a convolutional variational autoencoder for the generation of musical instrument tones," in *The 1st World Conference on eXplainable Artificial Intelligence*, 2023.

[6] John M. Grey, "Multidimensional perceptual scaling of musical timbres," *The Journal of the Acoustical Society of America*, vol. 61, no. 5, pp. 1270–1277, May 1977.

[7] Hiroko Terasawa, Malcolm Slaney, and Jonathan Berger, "Perceptual distance in timbre space," p. 8, 2005.

[8] Philippe Esling, Axel Chemla-Romeu-Santos, and Adrien Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces.," in *ISMIR*, 2018, pp. 175–181.

[9] Geoffroy Peeters, Bruno L. Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams, "The timbre toolbox: Extracting audio descriptors from musical signals," vol. 130, no. 5, pp. 2902–2916, 2011.

[10] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[11] Kıvanç Tatar, Daniel Bisig, and Philippe Pasquier, "Latent timbre synthesis: Audio-based variational auto-encoders for music composition and sound design applications," *Neural Computing and Applications*, vol. 33, no. 1, pp. 67–84, 2021.

[12] David Brynjar Franzson, Thor Magnusson, and Victor Shepardsson, "Autocoder: a variational autoencoder for spectral synthesis," 2022.

[13] Yin-Jyun Luo, Kat Agres, and Dorien Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders," *International Society for Music and Information Retrieval Conference (ISMIR)*, 2019.

[14] Ruihan Yang, Dingsu Wang, Ziyu Wang, Tianyao Chen, Junyan Jiang, and Gus Xia, "Deep music analogy via latent representation disentanglement," *International Society for Music and Information Retrieval Conference (ISMIR)*, 2021.

[15] Anastasia Natsiou and Seán O'Leary, "A sinusoidal signal reconstruction method for the inversion of the mel-spectrogram," in *2021 IEEE International Symposium on Multimedia (ISM)*, 2021, pp. 245–248.

[16] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 161–165.

[17] Brian CJ Moore and Brian R Glasberg, "Suggested formulae for calculating auditory-filter bandwidths and excitation patterns," *The journal of the acoustical society of America*, vol. 74, no. 3, pp. 750–753, 1983.

[18] Stephen Handel, "Timbre perception and auditory object identification," in *Hearing*, pp. 425–461. Elsevier, 1995.

[19] Geoffroy Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," *CUIDADO Ist Project Report*, vol. 54, no. 0, pp. 1–25, 2004.

[20] John W. Gordon, "The perceptual attack time of musical tones," vol. 82, no. 1, pp. 88–105, 1987.

[21] Guillaume Lemaitre, Nicolas Grimault, and Clara Suied, "Acoustics and psychoacoustics of sound scenes and events," in *Computational Analysis of Sound Scenes and Events*, Tuomas Virtanen, Mark D. Plumbley, and Dan Ellis, Eds., pp. 41–67. Springer International Publishing, 2018.

[22] Anssi Klapuri and Manuel Davy, "Signal processing methods for music transcription," 2007.

[23] Laurens Van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

VOCAL TIMBRE EFFECTS WITH DIFFERENTIABLE DIGITAL SIGNAL PROCESSING

David Südholt *

Centre for Digital Music
Queen Mary University of London
London, UK
d.sudholt@qmul.ac.uk

Cumhur Erkut

Multisensory Experience Lab
Aalborg University Copenhagen
Copenhagen, Denmark
cer@create.aau.dk

ABSTRACT

We explore two approaches to creatively altering vocal timbre using Differentiable Digital Signal Processing (DDSP). The first approach is inspired by classic cross-synthesis techniques. A pretrained DDSP decoder predicts a filter for a noise source and a harmonic distribution, based on pitch and loudness information extracted from the vocal input. Before synthesis, the harmonic distribution is modified by interpolating between the predicted distribution and the harmonics of the input. We provide a real-time implementation of this approach in the form of a Neutone model.

In the second approach, autoencoder models are trained on datasets consisting of both vocal and instrument training data. To apply the effect, the trained autoencoder attempts to reconstruct the vocal input. We find that there is a desirable "sweet spot" during training, where the model has learned to reconstruct the phonetic content of the input vocals, but is still affected by the timbre of the instrument mixed into the training data. After further training, that effect disappears.

A perceptual evaluation compares the two approaches. We find that the autoencoder in the second approach is able to reconstruct intelligible lyrical content without any explicit phonetic information provided during training.

1. INTRODUCTION

Neural singing voice synthesis has made great progress over recent years. Many efforts are focused on generating natural-sounding voices. The fame of the classic "vocoder" sound however, popularized by artists like Daft Punk or Kraftwerk shows the desire for creative timbre manipulation of vocals, where naturalness is not a desired characteristic.

Differentiable Digital Signal Processing (DDSP) [1] proposes an end-to-end learning approach for neural audio synthesis. Instead of generating signals sample-by-sample in the time domain, or time-varying spectra in the frequency domain, DDSP offers a library of synthesizer components implemented entirely within a framework supporting auto-differentiation. In the case of timbre transfer, an autoencoder model is trained to reconstruct a monophonic sound source based on into pitch and loudness information by generating time-varying control parameters for the synthesizers. The loss is calculated by comparing the spectrogram of the generated audio from the synthesizers to that of the original audio on multiple timescales. The auto-differentiable implementation allows the gradient of the loss to backpropagate through the synthesizers to update the model weights of the autoencoder.

The synthesizers are based on the spectral modeling synthesis (SMS) [2] framework. The harmonic components of the sound are generated by a sum of K sinusoids. The decoder predicts K time-varying amplitudes $A_k(n)$, referred to as the *harmonic distribution*, since the sinusoids are defined to oscillate at integer multiples of the (also time-varying) fundamental frequency $f_0(n)$ extracted by the encoder. Thus, the output of the harmonic component x_h can be formulated as

$$x_h(n) = a(n) \sum_{k=1}^{K} A_k(n) \cdot \sin(\phi_k(n)) \,, \tag{1}$$

where $a(n)$ is a global amplitude, also predicted by the decoder, and $\phi_k(n) = 2\pi \sum_{m=0}^{n} k f_0(m)$ is the instantaneous phase of the k-th harmonic.

Additionally, the decoder predicts the time-varying magnitude responses of a finite impulse response (FIR) filter. The non-harmonic components of the sounds are generated by processing white noise through these FIRs.

Recent approaches extended the DDSP components and source waveforms. Masuda [3] proposed a novel approach to synthesizer sound matching by implementing a basic subtractive synthesizer using differentiable DSP modules. Shan [4] introduced Differentiable Wavetable Synthesis (DWTS), a technique for neural audio synthesis that learns a dictionary of one-period waveforms through end-to-end training. Lee [5] formulated recursive differentiable artificial reverberation components, allowing loss gradients to be back-propagated end-to-end, and implemented these models with finite impulse response (FIR) approximations. Finally, Wu [6] proposed a new vocoder called SawSing for singing voice, which synthesizes the harmonic part of singing voices by filtering a sawtooth source signal with a linear time-variant finite impulse response filter whose coefficients are estimated from the input mel-spectrogram by a neural network.

Despite these achievements, the use of the classical DDSP for a vocal input with intelligible lyrics has not been explored or exploited, except in [7]. In this paper, we propose two methods of adapting DDSP to create vocal effects. We provide a real-time implementation and report perceptual experiments to evaluate our approaches. The structure of this short paper follows our approaches and experiments.

* Work performed as an M.Sc. student in Sound and Music Computing at Aalborg University Copenhagen

2. VOCAL EFFECTS WITH DDSP

Our first approach focuses on altering the predicted synthesizer parameters in a vocoding-inspired manner and will be referred to as the *vocoding approach*. The other makes use of a latent encoding of timbre information and will be referred to as the *latent approach*.

2.1. Vocoding Approach

The vocoding approach uses a model trained for timbre transfer. A decoder solely conditioned on pitch and loudness features is trained on audio recordings of a specific instrument, e.g. a trumpet. After training has completed, extracting pitch and loudness from any input audio can be used to generate the same melody line in the sound of a trumpet by using the trained decoder to predict corresponding synthesizer controls.

To create the effect of a "talking trumpet" from vocal input, we extract pitch and loudness information from the input. Before the synthesis step however, the harmonic distribution A_k is replaced by an altered distribution A_k^{out} by interpolating between the predicted distribution and the harmonics A_k^{in} of the input.

To generate A_k^{out}, a user-supplied interpolation factor $p \in [0, 1]$ is introduced. The modified distribution can then be calculated as

$$A_k^{\text{out}} = \begin{cases} (1-p)A_k + A_k^{\text{in}} & kf_0 < \frac{f_s}{2} \\ 0 & \text{else} \end{cases}, \tag{2}$$

taking care not to include oscillators at frequencies exceeding the Nyquist limit of $f_s/2$. Note that for $p = 0$, $A_k^{\text{out}} = A_k$. In this way, we can create a hybrid harmonic distribution containing both aspects of the timbre of the instrument the decoder was trained on, and the spectral contour of the phonetic content of the vocal input.

A real-time implementation of this approach is made available as a Neutone model at `https://github.com/dsuedholt/ddsp_xsynth`.

2.2. Latent Approach

In the latent approach, the encoder generates a vector z in addition to pitch and loudness information. We use an encoder provided in the DDSP library that calculates mel-frequency cepstral coefficients (MFCCs) of the input audio at every time step, and processes them through a recurrent layer before projecting them to the latent space.

In this approach, no modifications are applied to the decoder output. Instead, the effect is generated through selection of the training datasets. As explored previously [7] and confirmed through preliminary experiments, simply training a VAE on recordings of a singing voice can be sufficient to obtain a model capable of reconstructing a vocal input from a different singing voice in the style of the training data with intelligible lyrics.

The idea behind this approach is to add in other monophonic instruments, such as a trumpet or a synthesizer, to the training data, to influence the timbre of the reconstructed vocals in musically interesting ways.

During the experiments, it became clear that if the model is trained until the training loss converges, a decoder with a sufficient number of parameters learns to distinguish between vocal input and the additional instrument, and is able to reconstruct both accurately. This results in a model that is effectively just performing voice transfer.

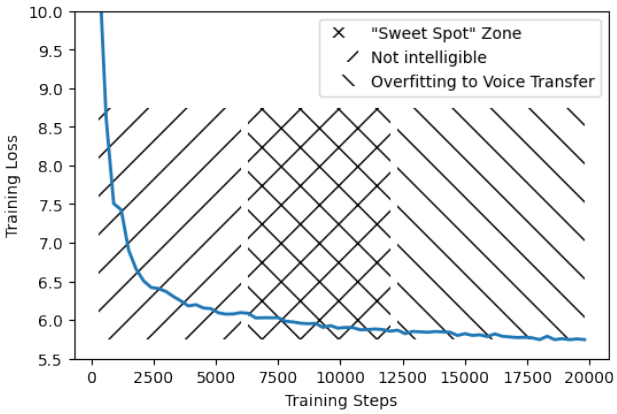


Figure 1: *The training process of a latent encoding model on a combined dataset of vocal performances and brass instruments. Early during training, it cannot reconstruct intelligible lyrics yet. Then it transitions into the "sweet spot" where lyrical content is preserved, but the timbre is affected by the additional instrument. After further training, that effect disappears, and the model performs regular voice transfer.*

However, there appears a "sweet spot" early on in training, where the model is already able to reproduce the lyrical content of the input, but has not yet learned to fully distinguish between the different input sources. At this point, the timbre of the reconstructed vocals is affected noticeably by the additional instrument. This is illustrated in Figure 1.

3. EXPERIMENTS

A dataset of vocal performances was created from the Children's Song Dataset (CSD) [8] and the MUSDB18 dataset [9]; instrument datasets were created by combining respective instrument recordings taken from the University of Rochester Multi-Modal Music Performance dataset (URMP) [10]. Additionally, a synthesizer performance was obtained by processing randomized MIDI at varying velocities and pitches through a software synthesizer.

Sound examples demonstrating the effect of the vocoding approach at various values for p, as well as the "sweet spot" effect of the latent approach, are available at `https://dsuedholt.github.io/ddsp-vocal-effects`.

We performed a perceptual evaluation to compare the two approaches. We trained and used the following four models:

VC-Synth: Timbre transfer model trained on the synth dataset, vocoding approach, $p = 0.7$

VC-Brass: Timbre transfer model trained on the brass dataset, vocoding approach, $p = 0.7$

Z-Vocals: Latent encoding voice transfer model trained on a single singer from the CSD dataset

Z-Mixed: Latent encoding model trained on a mixed dataset from the MUSDB18 medley vocals (multiple singers) and the synth dataset

Two vocal samples, one performed by a male, one by a female singer, were processed by all four models. 15 participants rated the output in a multi-stimulus test under the following three aspects:

1. Perceived audio quality
2. Intelligibility of the lyrics
3. How musically interesting the effect is

The results of the subjective evaluation are shown in Figure 2.

The clearest result can be found in the rating of the lyrical intelligibility aspect on the female input sample, where the latent encoding models clearly outperform the vocoding models. The same trend, although to a lesser degree, is shown in the evaluation of the male input sample. This seems to confirm that the MFCC + RNN encoder is already capable of reproducing intelligible lyrics without any explicit phonetic information.

None of the models are rated particularly favorably under the aspect of perceived audio quality, although the latent encoding models perform slightly better than the vocoding models. This could potentially be improved by working sampling rates greater than 16 kHz.

The highly subjective rating according to "musical interest" shows the highest variance of the ratings, although a slight trend favoring the latent encoding models seems to exist.

4. CONCLUSION

We presented two methods of creating vocal effects that show how the model training and the synthesis stage of the DDSP pipeline can be manipulated for creative effect. We demonstrated that no conditioning on explicit phonetic information is needed to preserve lyrical intelligibility while altering the timbre of the vocal input. These results pave the way towards synthetic "talking" instruments, as well as better understanding of the DDSP training mechanisms and strategies. Still, implementing a unified voice synthesis framework such as NANSY++ [11] remains a future challenge for our field in general.

5. REFERENCES

[1] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts, "DDSP: Differentiable Digital Signal Processing," in *International Conference on Learning Representations*, 2020.

[2] Xavier Serra and Julius O. Smith, "Spectral modeling synthesis. A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[3] Naotake Masuda and Daisuke Saito, "Synthesizer Sound Matching with Differentiable DSP," in *Proc. Intl. Soc. Music Information Retrieval Conf. (ISMIR)*, 2021.

[4] Siyuan Shan, Lamtharn Hantrakul, Jitong Chen, Matt Avent, and David Trevelyan, "Differentiable Wavetable Synthesis," in *Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Proc. (ICASSP)*. IEEE, may 23 2022.

[5] Sungho Lee, Hyeong-Seok Choi, and Kyogu Lee, "Differentiable Artificial Reverberation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2541–2556, 2022.

[6] Da-Yi Wu, Wen-Yi Hsiao, Fu-Rong Yang, Oscar Friedman, Warren Jackson, Scott Bruzenak, Yi-Wen Liu, and Yi-Hsuan Yang, "DDSP-based singing vocoders: A new subtractive-based synthesizer and a comprehensive evaluation," *arXiv*, 2022.

[7] Juan Alonso and Cumhur Erkut, "Explorations of Singing Voice Synthesis using DDSP," in *18th Sound and Music Computing Conference*, 2021, vol. 2021-June, pp. 183–190.

[8] Soonbeom Choi, "Children's Song Dataset for Singing Voice Research," in *International Society for Music Information Retrieval Conference*.

[9] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner, "The MUSDB18 corpus for music separation," Dec. 2017.

[10] Bochen Li, Xinzhao Liu, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma, "Creating a Multitrack Classical Music Performance Dataset for Multimodal Music Analysis: Challenges, Insights, and Applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, Feb. 2019.

[11] Hyeong-Seok Choi, Jinhyeok Yang, Juheon Lee, and Hyeongju Kim, "NANSY++: Unified voice synthesis with neural analysis and synthesis," in *Intl. Conf. Learning Representations (ICLR)*, 2023, Accepted as a poster.

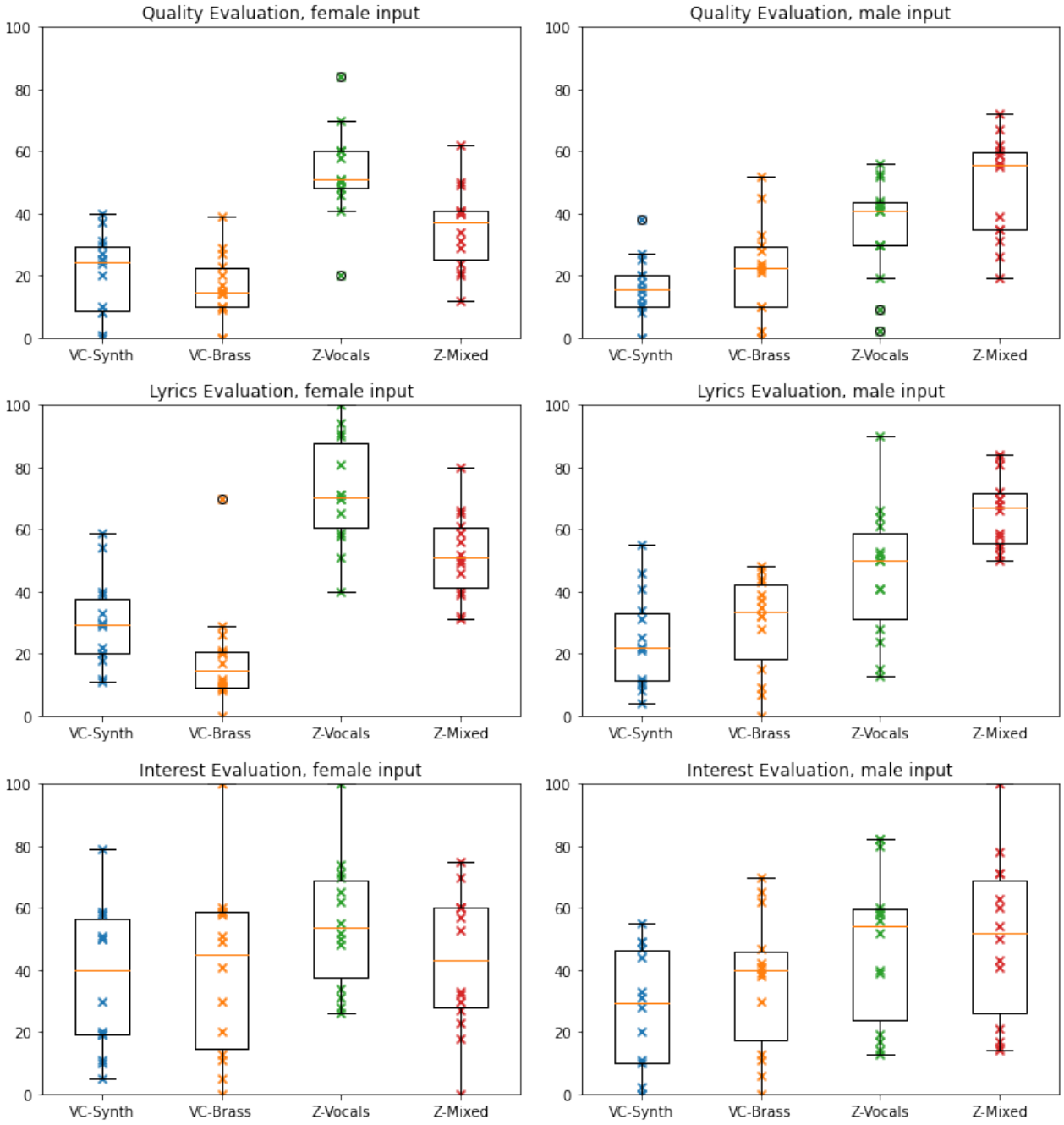


Figure 2: *Results of the perceptual evaluation. All individual ratings are displayed as a scatter plot. A box plot marks the median rating with a horizontal line. The box itself extends from the first to the third quartile of the ratings.*

WHAT YOU HEAR IS WHAT YOU SEE: AUDIO QUALITY FROM IMAGE QUALITY METRICS

Tashi Namgyal, Alexander Hepburn, Raul Santos-Rodriguez

Intelligent Systems Lab
University of Bristol
Bristol, UK
tashi.namgyal@bristol.ac.uk
alex.hepburn@bristol.ac.uk enrsr@bristol.ac.uk

Valero Laparra, Jesus Malo

Image Processing Lab
Universitat de Valencia
Valencia, Spain
valero.laparra@uv.es
jesus.malo@uv.es

ABSTRACT

In this study, we investigate the feasibility of utilizing state-of-the-art perceptual image metrics for evaluating audio signals by representing them as spectrograms. The encouraging outcome of the proposed approach is based on the similarity between the neural mechanisms in the auditory and visual pathways. Furthermore, we customise one of the metrics which has a psychoacoustically plausible architecture to account for the peculiarities of sound signals. We evaluate the effectiveness of our proposed metric and several baseline metrics using a music dataset, with promising results in terms of the correlation between the metrics and the perceived quality of audio as rated by human evaluators.

1. INTRODUCTION

Perceptual assessment of the quality of audio signals has been explored to varying degrees for different kinds of audio content. Whilst there exist several tools to understand speech quality [1], the evaluation of music is rarely explored and comes in the form of software hidden behind commercial licences [2]. More generally, practitioners rely either on traditional physical measures of the audio signal, e.g., signal-to-noise ratio (SNR), or more recent deep learning-based metrics that involve noninterpretable models to capture statistics of the degradation [3]. The picture is quite different in the visual modality, where many more perceptual models have been developed over the years for these purposes – and well-curated datasets are readily available [4, 5].

It is well-known that the auditory and visual processing pathways share similar attributes. For example, *divisive normalisation*, a form of local gain control, is a well explored phenomenon that is encountered when studying neurons in the brain [6, 7]. Specifically in vision, divisive normalisation has been shown to factorise the probability density function of natural images [8]. In audio the same phenomenon has been shown to minimise the dependencies between between natural sound stimuli responses to filters of certain frequencies [7]. Other behaviours such as signal adaptation can also be observed in both modalities [9]. Many of these ideas form the basis of the design of image quality metrics, but, as they are also observed in auditory statistics or psychophysical tests, we argue they should be included in the design of audio quality metrics.

The algorithmic parallelism and interaction between neural pathways implies that audio signals can alter the perception of visual stimuli [10], and examples of this (audio-to-vision) correlation are present even in pop music [11]. Here we take the opposite (vision-to-audio) approach: *what you hear is what you see*.

We draw inspiration from state-of-the-art image quality metrics to bridge the gap with their audio counterparts, which are not so successful at predicting perceived quality, for example, when evaluating neural audio synthesis [12]. Although raw audio takes a very different form to images, well-studied transformations can be used to align the two modalities. For example, spectrograms represent audio signals using image-like 2D matrices, where each column represents a time window and each row is a frequency band. As such, spectrograms encode the audio signal similar to wavelet decompositions that are often used in image metrics [8, 13]. We can then use these representations to exploit the literature on image quality metrics (IQMs) to estimate audio quality. Importantly, whilst the structure and semantics of spectrograms are different to images, the underlying principles are similar, e.g. the importance of amplitude (brightness) and local differences (contrast).

The paper is organised as follows: firstly, we show that popular IQMs can outperform metrics specifically designed for audio. Secondly we show that fine-tuning a traditional IQM based on divisive normalisation, which is also seen in auditory processing, can further improve results. We also provide the intuition behind what this tuned metric is capturing about the properties of audio.

2. QUALITY METRICS

Quality metrics aim to replicate the distance between two examples perceived by a human. This usually involves projecting the raw data to a perceptually meaningful space and computing a distance, or computing and comparing statistical descriptors of the examples. Below we will detail a number of audio and image quality metrics used throughout the paper.

2.1. Image Quality Metrics

Traditional IQMs fall into two categories; *structural similarity*, comparing descriptions of image statistics, and *visibility of errors*, which aims to measure how visible the distortions are to humans. Multi-Scale Structural SIMilarity (MS-SSIM) [14] is based on the former and compares three descriptors (luminance, contrast and structure) at various scales. Normalised Laplacian Pyramid Distance (NLPD) [15], based on the visibility of errors, is inspired by the biological processing in the visual system. Coincidentally, this

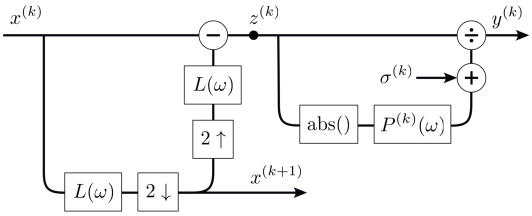


Figure 1: *Architecture for one stage k of the Normalised Laplacian Pyramid model, where $x^{(k)}$ is the input at stage k, $L(\omega)$ is a convolution with a low-pass filter, $[2\downarrow]$ is a downsample of factor two, $[2\uparrow]$ is an upsample of factor two, $x^{(k+1)}$ is the input image at stage $(k+1)$, $P^{(k)}(\omega)$ is s scale-specific filter for normalising the image with respect to the local amplitude, $\sigma^{(k)}$ is scale-specific constant and $y^{(k)}$ is the output at scale k. Figure taken from [15].*

processing is also present in the auditory system and we will use this to fine-tune NLPD to audio (sec. 3).

2.2. Audio Quality Metrics

Audio quality metrics have typically been designed for evaluating audio coding and source separation artifacts [16]. Here, we compare three recent metrics. Fréchet Audio Distance (FAD) [17] is a reference-free metric for evaluating generated audio based on the Fréchet Inception Distance (FID) commonly used in images [18]. FAD uses embeddings from the VGGish model [19] to measure the distance between previously learned clean studio quality music and a given audio clip. Virtual Speech Quality Objective Listener (ViSQOL) [20] is a full-reference metric based on the Neural Similarity Measure (NSIM) [21] between spectrograms. NSIM is similar to SSIM, using the luminance and structure terms but dropping the contrast term. Additionally it uses a support vector regression model to map the NSIM scores more closely to Mean Opinion Scores. The discriminator output of a Generative Adversarial Network (GAN) can also be used to predict perceptual ratings [3].

3. NORMALISED LAPLACIAN PYRAMID DISTANCE

NLPD is our example case for adapting existing image metrics to audio. The Laplacian Pyramid is well known in image coding [22]. The signal is encoded by applying a low-pass filter and then subtracting this from the original image multiple times at various scales, creating low entropy versions of the signal. The Normalised Laplacian Pyramid (NLP) extends this with a local normalisation on the output of each pyramid level [15]. These two steps are similar to the early stages of the visual and auditory systems where linear filtering and local normalisation are present [7, 13, 9]. The distance in this new domain is referred to as NLPD [15, 23], correlates well with human perception, and reduces redundancy in agreement with the efficient coding hypothesis [8].

An overview of the architecture is detailed in Fig. 1. Given two images, x_1 and x_2, we compute the outputs $y_1^{(k)}$ and $y_2^{(k)}$ at every stage of the pyramid k, and sum the differences:

$$\text{NLPD}(x_1, x_2) = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{\sqrt{N_s^{(k)}}}||y_1^{(k)} - y_2^{(k)}||_2 \qquad (1)$$

where N is the number of stages in the pyramid, and $N_s^{(k)}$ is the number of coefficients at stage k.

4. EXPERIMENTS

4.1. Data

We use the Perceived Music Quality Dataset from [3]. It consists of 4-second audio clips across 13 genres, with 5 songs per genre and 3 clips per song, totalling 195 reference clips. These reference clips are degraded in four ways: waveshape distortion, low pass filtering, limiting and additive noise, resulting in 975 clips. We divide this into an 80-20 train-test split, in which the test set contains all 3 clips for the last song in each genre. Each clip has an associated perceptual quality rating on a scale from 1 to 5 These ratings were gathered using Amazon Mechanical Turk using a no-reference paradigm. Each clip was rated by at least 5 participants and the median value was taken.

For the SSIM, NLPD, and Mean Square Error (MSE) metrics the audio clips are downmixed into mono and converted into mel spectrograms. The audio is downsampled from 48kHz to 16050Hz with 512 mel-bands, a window size of 2048, and a hop-length of 64, resulting in spectrograms of size 512×1024. For NLPD we use 6 pyramidal layers, with inputs being halved in size for each layer down to 16×32. The SSIM ratings are calculated using Pytorch MS-SSIM[1]. For ViSQOL and FAD, the clips are downmixed into mono and converted from 32-bit to 16-bit WAV files. Ratings are calculated using the ViSQOL[2] and FAD[3] packages.

Table 1: *Spearman correlation between human ratings and various metrics. NLPD [15] and (MS-)SSIM [14] are image quality metrics, whilst ViSQOL [20], FAD [17] and GAN [3] are audio quality metrics. We report the correlation for each degradation type separately as well as for all degradations simultaneously. GAN results are taken from the respective paper [3].*

Metric	Waveshape	Lowpass	Limiter	Noise	All
MSE	**0.469**	-0.049	**0.378**	0.641	0.483
NLPD	0.468	0.012	0.339	**0.681**	0.633
SSIM	-0.450	-0.175	-0.356	-0.629	**-0.656**
MS-SSIM	-0.468	-0.045	-0.323	-0.654	-0.648
ViSQOL	-0.142	0.191	-0.316	-0.629	-0.232
FAD	0.386	-0.083	0.316	0.550	0.593
GAN*	0.349	**0.222**	0.120	0.359	0.426

4.2. NLPD Optimisation

To fine-tune NLPD to audio we optimise the filters $P^{(k)}(\omega)$ and the constant $\sigma^{(k)}$ in the divisive normalisation stages. There are two possible methods; statistically [15] or perceptually [13]. We use 5x5 filters in both cases.

Optimising statistically consists of calculating average pixel values of the band-passed spectrograms z separately for each layer k. The divisive normalisation filters are learned as weights p_j that transform the weighted sum of pixel values in the neighbourhood

[1] https://github.com/VainF/pytorch-msssim
[2] https://github.com/google/visqol
[3] –/google-research/google-research/tree/master/frechet_audio_distance

surrounding each pixel to approximate the centre pixel, j:

$$f_C^{(k)}\left(\mathbf{z}_{N_i}\right) = \sigma^{(k)} + \sum_{j \in N_i} p_j^{(k)} \left| z_j^{(k)} \right| \tag{2}$$

where N_i defines the neighbourhood (filter size) to be considered. The constant $\sigma^{(k)}$ is the mean absolute value of z for each layer:

$$\sigma^{(k)} = \frac{1}{N_s^{(k)}} \sum_{i=1}^{N_s^{(k)}} \left| z_i^{(k)} \right| \tag{3}$$

where N_s is the number of coefficients at stage k, i.e. dimensions of z. The weights are optimised with Eq. 4. We optimise over the reference spectrograms contained in the training set only, using ADAM optimiser, learning rate 0.01, batch size of 1 for 10 epochs.

$$\hat{\mathbf{p}}^{(k)} = \underset{\mathbf{p}}{\operatorname{argmin}} \sum_{i=1}^{N_s^{(k)}} \left(\left| z_i^{(k)} \right| - f_C\left(\mathbf{z}_{N_i}^{(k)}\right) \right)^2 \tag{4}$$

Optimising perceptually consists of maximising the Pearson's correlation between the NLPD and the human ratings of each reference audio clip and a degraded version of the clip. The filters are initialised to be the image NLPD values, and $\sigma^{(k)}$ is initialised with Eq .3. We use ADAM optimiser to maximise the Pearson correlation with a learning rate of 0.001 for 100 epochs, where each batch only contains one degradation. We use Pearson as the training objective instead of Spearman's, assuming approximately linear rankings, as the sorting operation has undefined gradients.

Table 2: *Spearman correlations for variations of the NLPD. Original: filters fit statistically to natural images [15]. No DN: NLP with no divisive normalisation stage.* $P(\omega) = 1$*: divisive normalisation filters are all ones. Statistical: filters optimised to predict the center pixel given its neighbours. Perceptual: model optimised to maximise correlation with human ratings.*

Metric	Waveshape	Lowpass	Limiter	Noise	All
Original	**0.468**	0.012	0.339	**0.681**	0.633
No DN	0.412	-0.052	0.336	0.670	0.617
$P(\omega) = 1$	0.457	-0.022	**0.380**	0.669	0.629
Statistical	0.432	-0.033	0.356	0.660	0.619
Perceptual	0.430	**0.035**	0.347	0.637	**0.643**

5. RESULTS AND DISCUSSION

5.1. Main findings

Table 1 shows correlations between humans and perceptual quality metrics. Surprisingly, IQMs perform better than AQMs for all degradations other than the low pass filter. However, the limiter and low pass filter had much weaker p-values so these correlations could be due to chance. We think this is partly because the amount of degradation applied in those cases was not high enough compared to the waveshaping and noise degradations. This is indicated by the degraded audio being judged as better quality than the reference in some pairs.

Table 2 shows results for adapting NLPD to audio using five different divisive normalisation strategies: 1. using filters from NLPD optimised statistically on natural images (as in Table 1), 2. with no divisive normalisation, 3. setting the filters in the divisive normalisation to one (for equal contribution of all the neighbours), 4. fitting the model statistically on spectrograms (where no perceptual information is used), and 5. maximising the correlation between spectrograms and the opinion of humans. The perceptually trained divisive normalisation has the highest correlation when all degradations are tested simultaneously, and other strategies trained on spectrograms increase correlation for the low pass and limiter degradations. For waveshape and noise, the forms of divisive normalisation using spectrograms decrease the correlation compared to training on natural images. The relationship between the degradations tested and the form of divisive normalisation used could be further explored as this process may effectively be reducing certain degradations, i.e. enhancing the signal.

Fig. 2 shows the learned divisive normalisation filters at different layers of the NLPD for 3 optimisation strategies. For the first four layers, the statistical spectrogram model focuses almost exclusively within the central frequency band, particularly at the time steps immediately before and after the central bin. This is similar to later layers of the model fit to natural images but different from early layers, where both directions are important. The later layers of the statistical spectrogram model look across bands but only within a single timestep in a manner completely unlike the image model. This may reflect the pattern of repeating harmonics in spectrogram signals. This may only be captured at later layers as early layers have a higher resolution, i.e. there are more frequency bands between harmonics. Using larger filters at early layers may help to capture this. The fact that the model only uses the central timestep at later layers may reflect the way that later layers are effectively averaging across longer time windows. As such, the signal will vary less smoothly between time bins and so will be less predictive of the central value. Larger time filters may start to capture rhythmic information. In contrast, the perceptual spectrogram filters consider both time and frequency simultaneously, like early layers of the image model, with layers exhibiting more smoothing behaviour in general. This may indicate that perceptually trained models may be better at capturing degradations that effect lower energy regions than statistical models.

5.2. Further Work

We have identified a need for a greater number of publicly available datasets of perceptual quality in audio with a larger variety of sounds and degradation types. The scores in the GAN and ViSQOL papers were collected according to the ITU-T P.800 recommendation (for telephone conversations). However, according to ITU-R BS.1534-1 this proved insufficient for evaluating audio signals of broadcast quality. Instead the "MUlti Stimulus test with Hidden Reference and Anchor" is the recommended grading procedure, as is used in [12]. The non-adaptive psychophysical Two Alternate Forced Choice (2AFC) paradigm, as used in IQMs [4, 5] would also be suitable. Preliminary tests should be performed to ensure the range of degradation is similar across degradation types. Tests should also scale degradation amounts to avoid improving the perceptual quality above the reference. A task training procedure or better task descriptions could also improve rating quality, as according to [3], participants were asked "How do you rate the audio quality of this music segment?" where "quality" is left largely up to participants to interpret. Such a dataset would allow for a more reliable comparison with contemporary [24, 25] and future AQMs. We also plan to investigate how divisive normalisation

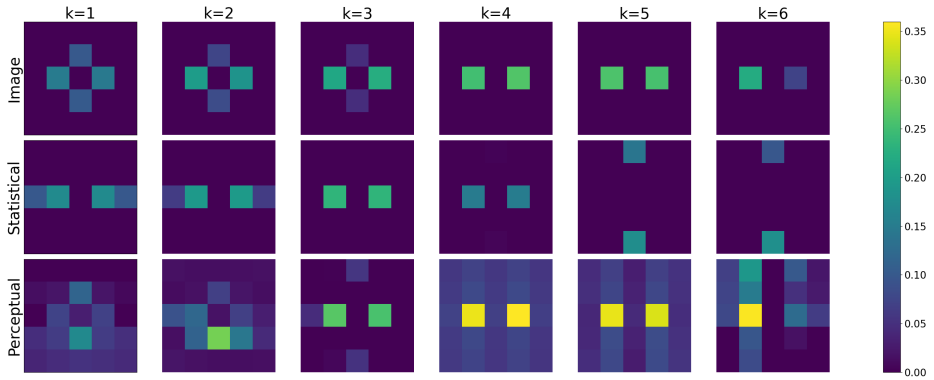


Figure 2: *Divisive normalisation filters learnt different optimisation strategies. Each column is a different layer in the pyramid, k. The top row (image) is the implementation of NLPD used for images, the second row (statistical) is statistically fit to audio spectrograms (Eq. 4), and the final row (perceptual) are filters resulting from fitting NLPD to perceptual judgements on audio.*

may be better tailored to audio, such as by using separate filters for time and frequency. We intend to use these metrics as a loss function in generative modelling, so that such models generate audio samples that sound more realistic with fewer perceived distortions. We also want to investigate the degree to which navigating through latent spaces of models trained with perceptual metrics aligns with human expectations of how the generated audio should change. We believe this should help with explainability, trust and control over the outputs of generative audio models.

6. ACKNOWLEDGMENTS

TN is supported by the UKRI AI CDT (EP/S022937/1). AH and RSR are supported by UKRI Turing AI Fellowship EP/V024817/1. VL and JM are supported by MINCEO and ERDF grants PID2020-118071GB-I00, DPI2017-89867-C2-2-R and GV/2021/074.

7. REFERENCES

[1] H. Gamper et al., "Intrusive and non-intrusive perceptual speech quality using CNNs," *IEEE WASPAA*, pp. 85–9, 2019.

[2] T. Thiede et al., "PEAQ-the ITU standard for measurement of perceived audio quality," *J. Audio-Eng. Soc.*, vol. 48, no. 1/2, pp. 3–29, 2000.

[3] A. Hilmkil, C. Thomé, and A. Arpteg, "Perceiving music quality with GANs," *arXiv:2006.06287*, 2020.

[4] N. Ponomarenko et al., "TID2013: Peculiarities, results and perspectives," *Sig.Proc.Im.Comm.*, vol. 30, pp. 57–77, 2015.

[5] R. Zhang et al., "The unreasonable effectiveness of deep features as a perceptual metric," *IEEE CVPR*, pp. 586–95, 2018.

[6] M. Carandini and D. J. Heeger, "Normalization as a canonical neural computation," *Nat.Rev.Neurosci.*, vol. 13, pp. 51–62, 2012.

[7] O. Schwartz and E. P. Simoncelli, "Natural signal statistics and sensory gain control," *Nat.Neurosci.*, vol. 4, pp. 819–825, 2001.

[8] J. Malo and V. Laparra, "Psychophysically tuned divisive normalization factorizes the PDF of natural images," *Neural Comput.*, vol. 22, no. 12, pp. 3179–3206, 2010.

[9] B. Willmore and A. King, "Adaptation in auditory processing," *Physiol.Rev.*, vol. 103, no. 2, pp. 1025–1058, 2023.

[10] L. Shams, Y. Kamitani, and S. Shimojo, "What you see is what you hear," *Nature*, vol. 408, no. 12, pp. 788, 2000.

[11] W. Mertens, "What you see is what you hear. Usura Records. Belgium," 2006.

[12] A. Vinay and A. Lerch, "Evaluating generative audio systems and their metrics," in *ISMIR*, 2022.

[13] V. Laparra et al., "Divisive normalization image quality metric revisited," *JOSA A*, vol. 27, no. 4, pp. 852–64, 2010.

[14] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality," in *37th IEEE Asilomar Conf. Sig. Syst. Comp.*, 2003, vol. 2, pp. 1398–1402.

[15] V. Laparra, J. Ballé, A. Berardino, and E. P. Simoncelli, "Perceptual image quality assessment using a normalized laplacian pyramid," *Electr.Imag.*, vol. 2016, no. 16, pp. 1–6, 2016.

[16] M. Torcoli et al., "Objective measures of perceptual audio quality reviewed," *IEEE/ACM Trans. Audio, Speech, Lang. Proc.*, vol. 29, pp. 1530–1541, 2021.

[17] K. Kilgour et al., "Frechet audio distance: A reference-free metric for evaluating music enhancem.," *InterSpeech*, 2019.

[18] M. Heusel et al., "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *NeurIPS*, 2017, p. 6626–6637.

[19] S. Hershey et al., "CNN architectures for large-scale audio classification," in *IEEE ICASSP*, 2017, pp. 131–135.

[20] M. Chinen et al., "ViSQOL: An open source production-ready object. speech and audio metric," *IEEE QoMEX*, 2020.

[21] A. Hines and Harte. N, "Speech intelligibility prediction using a neurogram similarity index measure," *Speech Comm.*, vol. 54, no. 2, pp. 306 – 320, 2012.

[22] P. Burt and E. Adelson, "Laplacian pyramid as a compact image code," *IEEE Trans.Comm.*, vol. 31, pp. 532–540, 1983.

[23] V. Laparra et al., "Perceptually optimized image rendering," *JOSA A*, vol. 34, 2017.

[24] P. Manocha et al., "CDPAM: Contrastive learning for perceptual audio similarity," in *ICASSP*, 2021.

[25] C. Gupta et al., "Parameter sensitivity of deep-feature based evaluation metrics for audio textures," in *ISMIR*, 2022.

DESIGNING A LIBRARY FOR GENERATIVE AUDIO IN UNITY

Enrico Dorigatti and Stephen Pearse

School of Creative Technologies
University of Portsmouth
Portsmouth, UK
enrico.dorigatti@port.ac.uk | stephen.pearse@port.ac.uk

ABSTRACT

This paper overviews URALi, a library designed to add generative sound synthesis capabilities to Unity. This project, in particular, is directed towards audiovisual artists keen on working with algorithmic systems in Unity but can not find native solutions for procedural sound synthesis to pair with their visual and control ones. After overviewing the options available in Unity concerning audio, this paper reports on the functioning and architecture of the library, which is an ongoing project.

1. INTRODUCTION

Unity is a game development software used in a range of scenarios by a diverse and wide audience, from enthusiasts to researchers in academia. Besides its capabilities concerning the development of multi-platform games and software—Mac OS, Windows, iOS, and Android being the most popular ones—the wide choice of options and the flexibility it offers makes it the ideal, user-friendly environment for fast prototyping especially when it comes to VR (virtual reality), XR (extended reality), and the production of virtual and simulated environments. Examples of the widespread usage and diverse contexts of application can be found in [1] and [2] (VR); [3] (business); [4] (visualisation of biomedical data); [5] (scholarly research); and [6] (automotive). Furthermore, it is also used within the artistic context. Interestingly, there is a general lack of academic resources reporting on this usage of Unity; however, it is possible to trace it back by visiting the websites of some artists and reading their artistic statements or program notes—although, as the focus is usually on the artistic outcome or message conveyed by an artwork, the tools used in the creative process are often omitted. Some examples, however, are reported below, and, additionally, a dedicated page on the Unity website, reports and highlights the specific features the software offers to artists and designers[1].

Unity is composed of two main components. Firstly, a robust graphics and 3D engine allow one to quickly and easily draft complex scenes and environments using 3D models, lights and shadows, and materials with custom properties, as well as effects such as particle systems, filters, and custom shaders—either written in GLSL or created via the built-in node-based editor. Furthermore, the physics engine allows for the design of interactions between objects and, in general, the different surfaces of the environment, making it possible to create complex behaviours that either mimic real-world physics or introduce randomness and imaginary behaviours.

Within Unity, these two systems work seamlessly, and, together with the possibility to control the behaviour of almost any parameter via custom C# scripts—thus connecting graphic power to computation—makes Unity the ideal environment when it comes to the design of generative art systems based on algorithms, either contemplating human interaction or not. Some examples this way are given by the works of Danish artist Carl Emil Carlsen[2], some software developed by the composer and media artist Giovanni Albini, such as Memoriale[3], and Life[4], a generative artwork developed by the first author and based on the Life algorithm developed by J. H. Conway [7]. However, when it comes to audio, Unity does not natively offer the same level of flexibility one can find in its components dedicated to scripting and visuals, especially when it comes to algorithmically based projects.

Nowadays, the main competitor of Unity is Unreal Engine, a source-available proprietary software which includes Metasound[5], a low-level, sample-accurate node-based system that allows developers to create synthesis and music systems within the engine—one of the most notable additions to the fifth version of the software. However, despite the great potential of Unreal Engine, establishing itself as the leading and reference platform in a wide range of scenarios spacing from architecture and automotive rendering and visualisation to game development, it is a popular opinion (e.g. on dedicated websites[6] or online communities[7]), especially amongst enthusiasts and small or indie developers that it has a steeper learning curve and developing a project from scratch with it requires much more effort—although strategies such as the Blueprints system[8] have been implemented to ease it out. Unity, on the contrary, has established itself as the go-to platform for fast prototyping and creation, especially when it comes to mobile devices. In this perspective, the project presented in this paper could be seen as a rudimental version of Metasound, an attempt to fill the gap concerning procedural audio synthesis in Unity highlighted when developing Life. The goal of URALi is thus to offer user-friendly generative audio capabilities directly from within Unity.

[1] https://unity.com/solutions/artist-designers

[2] https://cec.dk/

[3] https://play.google.com/store/apps/details?id=com.albinigiovanni.memoriale

[4] https://www.enricodorigatti.com/wp-content/uploads/2022/01/Life.mp4

[5] https://docs.unrealengine.com/5.0/en-US/metasounds-in-unreal-engine/

[6] https://gamedevacademy.org/unity-vs-unreal/

[7] https://www.quora.com/Is-Unity-easier-than-Unreal

[8] https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/

2. AUDIO IN UNITY

When it comes to generative audio, Unity does not natively offer solutions matching its possibilities in terms of graphics and physics. However, besides the stock audio filters providing common effects such as delay, echo, equalisation, and 3D spatialisation, different approaches can be followed to deal with sound when designing an application. Such approaches imply choosing different objects and workflows and are highly dependent on the type of project one is developing.

2.1. Audio Clips

The main way to deal with audio is based on the usage of pre-made clips loaded as assets of the application in development and which can be played when necessary—for example, when an action is performed in the case of a game or a button is pressed in the case of a user interface. Some methods of the AudioClip[9] object allow the programmer to retrieve information from the audio file—such as frequency, number of channels, and duration—and perform fairly advanced operations, such as setting the sample data that the clip contains. It is clear, however, that filling a clip with procedurally generated audio data is not a trivial operation and, on average, this method will mostly come in handy for shaping the amplitude of the original data, as shown in the example provided along with the documentation of the method[10].

Therefore, whilst audio clips are necessary in the average scenario—for example in the case of a game, where a finite set of defined actions and thus audio events is repeated over and over and possibly manipulated through the stock filters based on the properties of the action itself such as the composition of the ground for footstep—they are not the ideal solution in the case of generative and algorithmic systems, when the properties of the system itself 'evolves' unpredictably, based on the state of an algorithm. The most common scenario involving the usage of clips sees a scene prepared with an audio listener component—the 'ears' of the user—and different audio clips ready to be triggered when a condition occurs and possibly manipulated and filtered by the spatial properties of the space in which the scene takes place, as well as the distance of the player from the spot where the action happens. Employing AudioClip objects can be seen as the standard and most common way of dealing with audio in Unity and fits the vast majority of use cases—although it offers a limited amount of options and freedom.

2.2. Middleware

Another possible way to deal with audio in Unity is to employ audio middleware such as the popular Wwise[11] or FMOD[12]. Those tools, designed and developed with the video game industry in mind are meant to offer flexibility in this specific context, allowing studios to separate the development of the audio engine from the development of the game engine, keeping them, however, linked and eventually integrating them together. The functioning of middleware is mainly based on the audio clips a sound designer imports, which can however be triggered and manipulated in complex ways based on the calls and parameters the middleware receives from the game engine—an approach granting a high amount of freedom and flexibility.

The drawback of this approach to audio in Unity is that software such as Wwise and FMOD can be complex to master and not everybody, especially in the case of one-man-teams, is likely to learn them from scratch—on the contrary, in medium to big-sized game studios, there are professionals whose job is just to integrate the sounds provided by the sound designers, building all the pipeline and system necessary to make them work smoothly in accordance to what happens in the game engine, thus providing a smooth interaction between the two systems.

2.3. Third-party software

Furthermore, another and more experimental way to deal with audio in Unity contemplates the connection of third-party software providing specific environments, tools, and abstractions for sound and music—the most popular ones being SuperCollider, Pure Data, ChucK, and Max/MSP—through protocols such as the popular OSC [8]. These software are well known in the experimental music and sound design contexts for providing sandboxes in which it is possible to create algorithms for sound synthesis and manipulation and algorithmic music composition.

Offering probably the highest amount of flexibility, however, similar to what happens with middleware, they are different systems and languages from Unity, which means that, once again, one has to learn how to use them from scratch. On top of that, unlike Wwise or FMOD, they are not designed to natively integrate with Unity, as they are primarily conceived as standalone software. Thus, this means that one has to employ tools acting as bridges to port their functionalities within the software developed in Unity, which in turn leads to compatibility issues and the impossibility of using all the functions, classes, and objects available—especially when it comes to export a project for platforms such as Android. Projects like Chunity [9] have been developed to offer a fully-functioning connection between Unity and the ChucK audio programming language [10], allowing one to fully integrate Chuck code within Unity projects. However, this solution does not solve the issues of the different languages, which still forces an artist or developer to learn a different and specific one.

2.4. OnAudioFilterRead

There is one last option to deal with audio in Unity natively by taking advance of the possibility offered by implementing a custom OnAudioFilterRead[13] function. According to the documentation, it fits the audio DSP chain as a custom filter—a filter is each effect within the chain, such as an echo effect or a low-pass filter—manipulating the data flowing from the preceding one. However, if OnAudioFilterRead is the first or sole filter in the chain, it can be used to procedurally generate the audio data. The purpose of OnAudioFilterRead is, to summarise, to provide an 'access window' on the audio data passing through. Beyond that, it is the programmer's choice to decide where to place it and, therefore, whether to use it to generate the data or only access and manipulate the existing ones—as an example, multiplying them by 0.5 will approximately halve the level output level.

Similar to what has been previously said about the AudioClip object, the challenge resides in that generating audio procedurally can be challenging. However, the difference lies in the way the two

[9] https://docs.unity3d.com/ScriptReference/AudioClip.html
[10] https://docs.unity3d.com/ScriptReference/AudioClip.SetData.html
[11] https://www.fmod.com/unity
[12] https://www.fmod.com/unity
[13] https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnAudioFilterRead.html

systems work. Whilst audio clips are especially useful when dealing with fixed sounds that do not change and need to be played only when necessary (e.g. a footstep occurring only when the player moves a step), OnAudioFilterRead, on the contrary, gives the possibility to create, manipulate and, in general, work on constant streams of data in real-time. This, in turn, makes possible the realisation of a flexible framework for the algorithmic generation of the samples—that is, the procedural generation of sounds.

3. URALI

Taking advantage of the OnAudioFilterRead function, URALi is a library designed and developed to provide an easy-to-use framework for procedural audio in Unity without the need for external software such as Wwise or FMOD nor specialised programming languages such as SuperCollider or Pure Data and related bridging software when the connection is not natively possible. URALi is the acronym for Unity Real-Time Audio Library, and, as the name suggests, it offers a collection of functions and classes meant to work as the building blocks for audio algorithms. The functioning of the library recalls the fashion of the aforementioned audio programming languages, where the programmer can use, connecting them together, a set of objects and functions which eventually will compose an algorithm for audio synthesis or manipulation. As it happens in SuperCollider, such objects are defined by strings instead of being represented by visual nodes, characteristics of languages such as Pure Data or Max/MSP.

The development of URALi started with different motivations, as it originally was an exploratory attempt to investigate possible solutions for generative audio in Unity for the artwork described in [7]. However, as the system developed proved worthy, and, most importantly, with potential yet to explore, it was reworked with the goal of building a solid, flexible, and scalable framework designed to offer easy and efficient access and utilisation of the functions and objects of the library, avoiding the drawback of the other audio solutions previously explored [11].

3.1. Design of the Library

The first consideration done while developing the library was that relying on the thread running OnAudioFilterRead for the calculation of the audio samples would lead—especially in the case of complex synthesis algorithms with many different generators, and especially when running on older systems—to slowdowns and data starvation. The official OnAudioFilterRead documentation indeed states that it needs to process chunks of audio data at fixed intervals to provide a smooth stream; should it not be able to do so—for example, due to too many calculations to perform—the data stream would break and, perceptually, this would result in glitches. For this reason, OnAudioFilterRead runs on a separate thread, which incidentally means that many Unity functions can not be used. This considered, as every operation performed within the dedicated thread would reduce the headroom for further processing and increase the risk of occurring in data starvation, it was decided to split the calculation of the data from their retrieval.

The solution implemented consisted of another separated thread continuously computing new audio samples without timing or constraints. The results are stored in a circular buffer accessible both by this dedicated thread and OnAudioFilterRead, and the safety of the data concerning overwriting is guaranteed by a system of flags which pauses the calculation of new samples if there is no more space available for storing them. Whilst paused, through busy waiting [12], the thread periodically checks if some buffer space has been marked as free by the flag system and possibly resumes its activity. The free space is flagged as that by OnAudioFilterRead, which, when necessary, fetches chunks of data from the buffer and, by leveraging the flag system, marks the corresponding slots as overwritable. Retrieved data are then sent through the pipeline, ready to be outputted or modified by any filters natively available in Unity, should they be stacked in the DSP chain.

User side, URALi is designed as a sandbox in which the programmer can work with all the classes and methods contained in the library and beyond. This means that it is possible to integrate and expand an audio chain built with the objects available from within URALi with custom processes and logic, as well as data, from other parts of the program—for example, to control the frequency of an oscillator based on the speed of the player, which is calculated in a separate script. URALi is indeed written in C#, the scripting language used in Unity, and this makes it possible to have a seamless stream of data to and from other scripts and to use functions defined elsewhere.

Concretely, URALi needs the programmer to define a synthesis function in which to build the actual synthesis chain. This function has to be passed to the class implementing the audio engine, and this latter has to be started. Once done, the dedicated audio thread of the library initiates its task, and audio data become available from the circular buffer as they get calculated. As a last step, to fetch these data and have them outputted, the programmer needs to implement a custom OnAudioFilterRead function from which to access, through a specific call to the library, the circular buffer, to retrieve the next chunk of audio data.

Outside the code, in the inspector, URALi, which within the environment is represented by OnAudioFilterRead, is visualised by the default visualisation of the latter, namely a VU meter showing the output level and the processing time it takes. However, due to the library design, this number should always stay in the green zone—provided that no additional operations are performed in OnAudioFilterRead—as all the calculations are done in the dedicated thread. As the OnAudioFilterRead representing the library is a node of the audio chain, it is possible to stack it with all the audio effects shipped with Unity, such as echo, delay, and filters. The possibility to seamlessly integrate URALi within the existing audio ecosystem of Unity makes it possible to streamline its development, avoiding the necessity of implementing, and thus duplicating, audio effects already available natively.

4. CONCLUSIONS

URALi is a project forked from specific technical and artistic research; thus, its advancement in terms of maintenance, optimisation, and implementation of new functions relies solely on the efforts and free time of the first author. This is why, although the project started in 2017 and was initially presented in 2019 [13], an initial version has not been released yet, and the documentation has not been compiled. At the current time, URALi provides access to an envelope generator, granular [14], frequency modulation (FM) [15], and additive synthesisers, waveshaper, LFO (low-frequency oscillator), lookup oscillator with built-in tables and the possibility to load new ones, panners with different pan laws, classic waveform oscillators (sawtooth, triangular, square, sinusoidal), and white noise generator. While implementing these nodes, the problem of aliasing [16] was faced and subsequently addressed by

implementing the technique described by Välimäki et al. [17].

Other functionalities are already listed for implementation and have been selected by also taking inspiration from the palette of well-established software such as Max/MSP[14]. However, unlike this language, it was chosen to not include any object dealing with logic as, given the nature of URALi and the language in which it is written, and based on what has been discussed previously, it would be easier for a programmer to implement their custom conditions and logic to fit their peculiar case rather than understanding and adapting any generic object provided by the library. This is diametrally different from what happens in Max/MSP, where the possibility to use stock objects offering even the most basic logic functions, controls, and tasks improves the readability of the code and prevents the programmer from building large networks of control objects or even being forced to deal with scripting. In general, however, the confrontation with other musicians and artists would be beneficial for the development of URALi as it would help to understand what is missing and what should be prioritised.

Currently, some demos of URALi, built for Windows-based machines, are available and aim to demonstrate some key features of the library: the procedural generation of audio, the seamless integration within the Unity audio ecosystem and the broader environment, and the possibility to control real-time the parameters of the synthesis algorithm. Each demo is a standalone application and showcases one or more of these features. Specifically, each of them employs a different audio generator unit (e.g. waveshaper, noise, FM synthesizer) and the movement of the mouse on both X and Y axes to control relevant parameters. Furthermore, some employ stock Unity audio effects (e.g. a low-pass filter, an echo). The code of these demos was written by linking the library to the Unity project as a .dll file and follows the structure mentioned earlier, as it is composed of the implementation of the synthesis function, the instructions to set up the audio engine, and the code to control the particle system in the middle—which has a merely aesthetic role although shares the data of the mouse position with the audio chain to loosely determine the direction of the particles. The demos will be made open source at release time as example projects; currently, however, the executable files can be sent upon request to the first author, and an audiovisual recording showcasing the functioning of some of them is available as well[15].

5. ACKNOWLEDGMENTS

The first author acknowledges the University of Portsmouth - Faculty of Creative and Cultural Industries for the PhD Scholarship supporting this research.

6. REFERENCES

[1] J. Jerald, P. Giokaris, D. Woodall, A. Hartholt, A. Chandak, and S. Kuntz, "Developing virtual reality applications with unity," in *2014 IEEE Virtual Reality (VR)*, 2014, pp. 1–3.

[2] S. Wang, Z. Mao, C. Zeng, H. Gong, S. Li, and B. Chen, "A new method of virtual reality based on unity3d," in *2010 18th International Conference on Geoinformatics*, 2010, pp. 1–5.

[3] S. Patil, G. Gaikwad, S. Hiran, A. Ikhar, and H. Jadhav, "metaar – ar/xr shopping app using unity," in *2023 International Conference for Advancement in Technology (ICONAT)*, 2023, pp. 1–11.

[4] N. H. Khalifa, Q. V. Nguyen, S. Simoff, and D. Catchpoole, "A visualization system for analyzing biomedical and genomic data sets using unity3d platform," in *Proc. 8th Australasian Workshop on Health Informatics and Knowledge Management*, 2015, pp. 47–53.

[5] F. Fontana, R. Paisa, R. Ranon, and S. Serafin, "Multisensory plucked instrument modeling in unity3d: From keytar to accurate string prototyping," *Applied Sciences*, vol. 10, no. 4, pp. 1452, Feb 2020.

[6] R. Schroeter and M. A. Gerber, "A low-cost vr-based automated driving simulator for rapid automotive ui prototyping," in *Adjunct Proceedings of the 10th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, New York, NY, USA, 2018, AutomotiveUI '18, p. 248–251, Association for Computing Machinery.

[7] E. Dorigatti, "Automating art: A case-study of cellular automata in generative multimedia art," in *Proc. Intl. Computer Music Conf.(ICMC)*, Limerick, Ireland, Jul. 3-9 2022, pp. 175–181.

[8] M. Wright, "Open sound control: an enabling technology for musical networking," *Organised Sound*, vol. 10, no. 3, pp. 193–200, 2005.

[9] J. Atherton and G. Wang, "Chunity: Integrated audiovisual programming in unity.," in *Proc. Intl. Conf. New Interfaces for Musical Expression (NIME)*, Genova, Italy, Jun. 5-7 2018, pp. 102–107.

[10] G. Wang and P. R. Cook, "Chuck: A concurrent, on-the-fly, audio programming language," in *Proc. Intl. Computer Music Conf. (ICMC)*, Singapore, Sep. 29-Oct. 4 2003.

[11] E. Dorigatti, "Interacting with audio in unity [manuscript submitted for publication]," in *Proc. Conf. Dictionary for Multidisciplinary Music Integration (DiMMI)*, Trento, Italy, Nov. 25-26 2022.

[12] F. Corradini, G. Ferrari, and M. Pistore, "Eager, busy-waiting and lazy actions timed computation," *Electronic Notes in Theoretical Computer Science*, vol. 7, pp. 96–114, 1997, EXPRESS'97.

[13] E. Dorigatti, "Urali: a proposal of approach to real-time audio synthesis in unity," in *Proceedings of the 16th Sound & Music Computing Conference*, I. Barbancho, L. J. Tardón, A. Peinado, and A. M. Barbancho, Eds., Malaga, Spain, May 28–31 2019, pp. 86–87.

[14] C. Roads, "Introduction to granular synthesis," *Computer Music Journal*, vol. 12, no. 2, pp. 11–13, 1988.

[15] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *journal of the audio engineering society*, vol. 21, no. 7, pp. 526–534, september 1973.

[16] J. Schimmel, "Audible aliasing distortion in digital audio synthesis.," *Radioengineering*, vol. 21, no. 1, 2012.

[17] V. Välimäki, J. Pekonen, and J. Nam, "Perceptually informed synthesis of bandlimited classical waveforms using integrated polynomial interpolation," *The Journal of the Acoustical Society of America*, vol. 131, no. 1, pp. 974–986, 2012.

[14] https://docs.cycling74.com/max8/vignettes/max_alphabetical

[15] https://www.enricodorigatti.com/wp-content/uploads/2021/12/URALi.mp4

REAL-TIME VISUALISATION OF THE MUSICAL TIMBRE BASED ON THE SPECTRAL ESTIMATES OF THE SNAIL-ANALYSER

Thomas Hélie, Charles Picasso, Robert Piéchaud,

STMS laboratory (IRCAM-CNRS-SU)
1, place Igor Stravinsky
Paris, France
firstname.name@ircam.fr

Michaël Jousserand, Tom Colinot

Buffet Crampon
Mantes-La-Ville, France
firstname.name@buffetcrampon.com

ABSTRACT

This article presents a real-time software solution that allows musicians to visualise the timbre content of their musical tones. The timbre representation is based on the spectral estimates of the Snail-Analyser, for a high frequency precision, and on a harmonic-like representation. After a brief review on the derivation of these estimates, some second-stage estimates and the mapping used for the timbre representation are described. The visual representations in the application have been prototyped using the MAX software and developed with the Juce framework.

1. INTRODUCTION

The Snail-Analyser is a real-time software that allows the spectral analysis of a sound with a high frequency precision [1, 2]. It delivers a visual rendering on a musical scale made of a spiral (one turn is one octave, one angle is one chroma).

The ATRIM project[1] aims to design reliable tools with reactive visual renderings adapted to high precision pitch and timbre analysis of musical wind instruments. It is devoted to provide musicians with accurate and informative feedback in real time for tuning and timbre assessment, during performance and in several contexts such as: expert testing and improvement of manufactured instruments, tuning or comparison of instruments, musical practice by helping musicians adjust their motor control to reach their own or a teacher's target (intonation, timbre, vibrato, glissando, playing effects, etc).

A first tuning application has been designed in the ATRIM project to assess the frequency-deviation from the target pitch, in Hertz [3]. Its technology involves demodulated phase signals, low-pass filtered phase-constancy indicators, and other complementary signals calculated in the spectral estimates of the Snail-Analyser. The visual rendering mimics electro-mechanical strobe-tuners that exploit a stroboscopic technology (see e.g [4, 5] for original work and [3] and references therein for some brief descriptive historical elements and more recent non-mechanical versions). A second version of this application has been designed to assess a more comfortable deviation, in cents [6]. It involves a reactive local-in-time constancy indicator of the demodulated phase.

[1] ATRIM is the French acronym for "Analyseur Temps-Réel haute précision de justesse et de timbre pour Instruments Musicaux" (High precision real-time pitch and timbre analyser for musical instruments). This project is supported by the plan "France Relance" (see acknowledgements at the end of the paper).

This paper addresses the assessment of timbre. This issue has a long history from the point of view of perception, cognition and signal processing (see e.g. [7, 8, 9]) and learning algorithms (see e.g. [10] and references therein for a recent work integrating results on timbre and instrumental playing techniques). A resulting general statement is that timbre is related to spectrotemporal indicators and their modulations, and that it intimately covaries with pitch and amplitude.

The paper is organised as follows. First, Section 2 presents a brief review on some spectral estimates used in the Snail-Analyser, the spiral visual rendering, a tuning spinner and in a rendering with harmonic structure that provides a starting basis to address musical timbre representation. Then, the analyser is complemented by an indicator overbuilt on that of local-in-time constancy of the demodulated phase, that characterises the harmonic synchronicity, in order to provide a set of spectrotemporal timbre indicators adapted to quasi-harmonic signals. Second, Section 3 presents the software prototype and the designed visual rendering based on these indicators. Section 4 ends with conclusive remarks and perspectives.

2. ANALYSIS METHOD

This section first presents a brief review on the principle used in the Snail-Analyser [1] with its spectral estimates and their use in the spiral rendering (in Section 2.1). These estimates are used in a harmonic view (in Section 2.2) of interest to represent timbre. They are complemented by an overbuilt estimate, the harmonic synchronicity (in Section 2.3), to be used in the final visualisation.

2.1. Review of the Snail-Analyser process

The analysis process of the input sampled signal is based on (see [2, Sec. 3.1] for more details) a spectral transform (here, a short-time Fourier transform) for successive frames, the complex values of which are interpolated according to a vector `Freq_v` of tuned frequencies that are exponentially spaced to correspond to equally spaced midi codes, and spectral estimates introduced in [1], some of them being recalled below to make this paper self-contained. Typical parameters are: $F_s = 44.1\,\text{kHz}$ (sampling frequency), $T_{\text{frame}} = 50\,\text{ms}$ (duration of a classic-shape window) associated with the next power of 2 of the corresponding number of samples (number of FFT points), $T = 5\,\text{ms}$ (incremental time step for overlap) and $R_m = 20$ frequency points per half tone (midi resolution) for `Freq_v` over the analysed midi range.

The downstream indicators of the Snail-Analyser are derived from the spectrum modulus and phase. They are presented by steps, denoted **(Si)** below, and defined for all frames starting at time $t \in \{nT \text{ s.t. } n \in \mathbb{N}\}$ and all frequencies f in `Freq_v`.

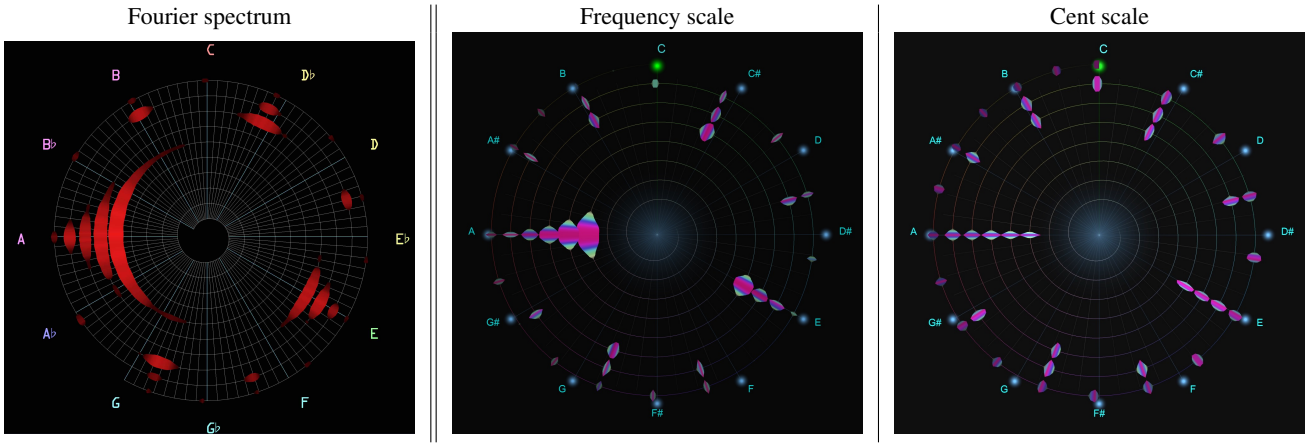


Figure 1: *Spiral renderings of the spectrum of an ideal harmonic signal (harmonics synthetised with uniformly distributed amplitudes). From left to right: Fourier transform (50ms, Hanning), Snail-Analyser (frequency precision:* 6 *Hz), Snail-Analyser (cent precision:* 5 *cents).*

.

(S1) Demodulated phase (ϕ_d): this signal is defined by

$$\phi_d(t,f) = \phi_{Fourier}(t,f) - 2\pi f t, \tag{1}$$

where $\phi_{Fourier}(t,f)$ denotes the spectrum phase.

(S2) Phase constancy indicator (C_{f_c}, set by f_c): this signal is defined as the squared modulus of the output of a low-pass filter (e.g. of Butterworth type) with cutoff frequency f_c, excited by the input complex signal $u_f : t \mapsto \exp\big(i\,\phi_d(t,f)\big)$, namely,

$$u_f(t) = \mathrm{e}^{i\,\phi_d(t,f)} \longrightarrow \boxed{\text{Low-Pass } (f_c)} \longrightarrow \boxed{\,|\cdot|^2\,} \xrightarrow{C_{f_c}(t,f)} \tag{2}$$

This indicator is used as a factor multiplying the spectrum loudness (ISO226:2003) to form the thickness represented on the spiral representation[2]. Compared to the Fourier spectrum (see Fig. 1 left), this improves the frequency accuracy by contracting the lobes to a precision of about $\pm f_c$ ($f_c = 6\,$Hz in center figure and $f_c(f) = 2^{\frac{\kappa}{1200}} f$ with $\kappa = 5\,$cents in the right figure). Due to the low-pass filter, this indicator also rejects time-varying components faster than f_c, which is of interest for tuning tasks in noisy environments.

To capture fast components (glissando, vibrato, etc) with the same precision ($\pm f_c$), a third (reactive) indicator is introduced.

(S3) Local-in-time constancy ($\delta\Phi_d$): this signal is defined by

$$\delta\Phi_d(t,f) = M[\phi_d(t,f) - \phi_d(t-T,f)], \tag{3}$$

where M denotes the 2π-modulo centered on $(-\pi,\pi)$.

This deviation indicator is local-in-time, at the scale of the incremental time step T. It quantifies a deviation-speed indicator that can be converted in Hertz (frequency deviation δF), in cents (cent deviation $\delta\kappa$) and used to contract the spectral lobes (factor $\tilde{C}_{f_c}$) according to the shape inherited from the low-pass filter chosen to compute C_{f_c}, as described below.

[2]The spiral skeleton (+1 round from the center is +1 octave, so that one angle is one chroma) is defined in polar coordinates by $\rho(f) = 1 + \log_2(f/f_{min})$, $\theta(f) = \theta_{ref} + 2\pi\log_2(f/f_{ref})$ where f_{min} and f_{max} are the lowest and highest frequencies to be displayed and f_{ref} is the tuning reference displayed at angle θ_{ref}.

(S4) Frequency deviation (δF): this signal defined by

$$\delta F(t,f) = \delta\Phi_d(t,f)/(2\pi T), \tag{4}$$

estimates a frequency deviation from f to the central frequency $F \approx f + \delta F$ of the spectral lobe of a stationary component (see e.g. [11] for an overview of other estimates used for spectrum reallocation). Its robustness (to perturbations or an unsteady frequency component) is achieved through the averaging of the demodulated phase evolution over the frame duration T_{frame}. Its reliability is limited to the range $\frac{1}{2T}(-1,1)$ of unaliased frequency deviations (100 Hz for $T = 5\,$ms).

(S5) Cent deviation ($\delta\kappa$): this signal is defined by

$$\delta\kappa(t,f) = \mathcal{C}ents\left(\frac{f+\delta F(t,f)}{f}\right), \tag{5a}$$

$$\text{where } \mathcal{C}ents(f_1/f_2) = 1200\log_2(f_1/f_2), \tag{5b}$$

estimates the deviation of f_1 from a reference frequency f_2 in cents. A good approximation of $\delta\kappa$ is $\frac{1200}{\ln 2}\frac{\delta F}{f}$ for $|\delta F/f| \ll 1$. Note that according to the definition of δF, the unaliased range in cents depends on f and is given by $\pm 1200\log_2\left(1+\frac{1}{2Tf}\right)$.

(S6) Contraction factor (factor $\tilde{C}_{f_c}$): this signal defined by

$$\tilde{C}_{f_c}(t,f) = \Big|\, H\big(\,\delta F(t,f)/f_c\,\big)\Big|^2, \tag{6}$$

restores the same shape as C_{f_c} for a stationary sound input signal, choosing $f \mapsto H(f/f_c)$ as the transfer function of the low-pass filter used to compute C_{f_c}.

The indicator ϕ_d mapped to the angle of a spinning pattern is the one used in the Snail-Analyser and in the strobe-tuner application [3], and that $\delta\kappa$ mapped to the angle velocity of the same pattern is the one used in the cent-sensitive version [6]. The contraction factor $\tilde{C}_{f_c}$ can be used in the Snail-Analyser: the combination of its high time reactivity and frequency precision makes it particularly suitable for enhancing real-time spectrogram-like rendering.

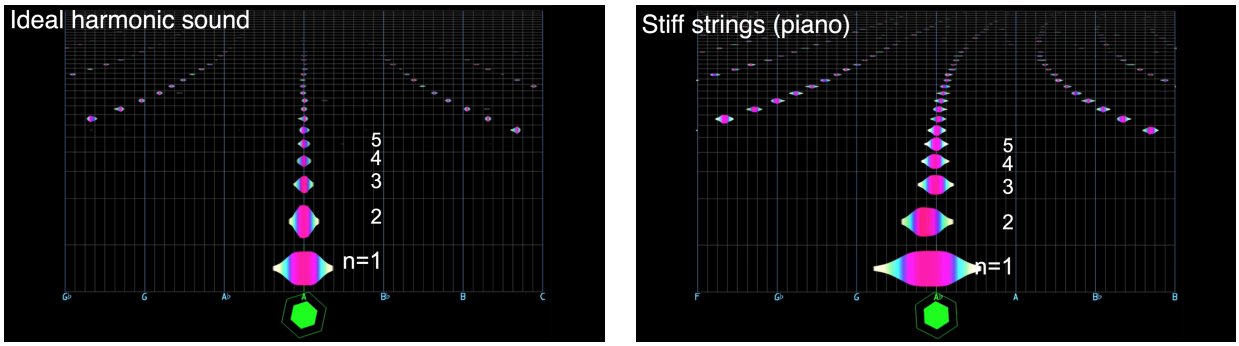


Figure 2: *Harmonic representation ($f_c = 6$ Hz): (left) ideal harmonic signal of Fig. 1; (right) inharmonic string (acoustic upright piano).*

2.2. Harmonic view

A new rendering to be integrated into future Snail-Analyser distributions is a harmonic view that aligns harmonic components (see Fig. 2). Given a fundamental frequency F_1 (e.g. 440 Hz for note A4), each stage n represents a spectral area of harmonics $n \geq 1$ on a cent-scaled axis centered according to the abscissa

$$X^n_{F_1}(f) = \mathcal{C}ents\left(\frac{f}{nF_1}\right) = 1200\log_2\frac{f}{nF_1}. \tag{7}$$

This representation is chosen as one of the bases of the timbre representation described in Section 3.

2.3. Harmonic synchronicity

The harmonic information is complemented by new descriptors of harmonic synchronicity in Hertz (ν_n) or in cents (σ_n), for each harmonic $n \geq 1$, as follows.

(S7) Harmonic frequency synchronicity (ν_n): it is defined by

$$\nu_n(t,f) = \delta F(t,nf) - n\,\delta F(t,f). \tag{8}$$

This descriptor provides a local-in-time estimation of the harmonic synchronicity for nearly harmonic signals in the following sense.

Consider an input signal composed of nearly harmonic partials of frequencies $F_n = n(1+\epsilon_n)F_1$ (with $\epsilon_1 = 0$), where ϵ_n encodes the relative deviation of F_n compared to the expected harmonic frequency nF_1. Then, consider an analysis frequency f and assume that nf is within the frequency range of the spectrum lobe of the partial of frequency F_n. In this case, $\delta F(t,nf) \approx F_n - nf$ estimates the frequency deviation between the ground-truth frequency F_n and the analysis frequency nf, and $\nu_n(t,f) \approx n(1+\epsilon_n)F_1 - nF_1 = n\epsilon_n F_1$ estimates the deviation from the harmonic frequencies, independently from the analysis frequency f.

A conversion of ν_n into cents (relative to nf) is below.

(S8) Cent harmonic synchronicity (σ_n): it is defined by

$$\begin{aligned}\sigma_n(t,f) &= \mathcal{C}ents\left(\frac{nf+\nu_n(t,f)}{nf}\right)\\ &= 1200\log_2\left(1+\frac{\nu_n(t,f)}{nf}\right),\end{aligned} \tag{9}$$

a good approximation of which is given for small frequency deviations by $\sigma_n(t,f) \approx \frac{1200}{\ln 2}\,\frac{\nu_n(t,f)}{nf}$.

Note that a version into cents, relative to an estimate of the target harmonic frequency $nF_1 \approx n\big(f+\delta F(t,f)\big)$, is $\tilde{\sigma}_n(t,f) = 1200\log_2\left(1+\frac{\nu_n(t,f)}{n\left(f+\delta F(t,f)\right)}\right) \approx \frac{1200}{\ln 2}\,\frac{\nu_n(t,f)}{n\left(f+\delta F(t,f)\right)}$.

3. VISUAL RENDERING AND SOFTWARE

3.1. Description of the visualisation

The software is a first prototype integrating a tuning spinner [6] and a harmonic representation inspired from Fig. 2 complemented by spinners cent-sensitive to the harmonic synchronicity (σ_n). The visual rendering is organised in two main parts (see Fig. 3).

The left part displays a vertical representation of the harmonics, with the fundamental (of target pitch F_1) at the bottom of the representation and higher harmonics superimposed in tiers above. Note that this vertical harmonic representation is not the spectrum unrolled on the frequency Y-axis but a superposition of visual windows, centered around these harmonics, extracted from the Snail analysis kernel. Each visual window covers a range of ± 1 semitones around the target harmonic frequency (nF_1), the target being represented by a pink line in the middle of the window.

The right part of each window displays a rotating spinning hexagon, that conveys pitch and harmonic synchronicity information. In the lowest one (pitch window), the rotation speed indicates the pitch deviation from the target F_1. Its color linearly varies from red (50 cents or more deviation) to green (0 cent). In the staged upper windows (harmonics $n \geq 2$), the rotation of the hexagons (of fixed blue color for now) indicates the harmonic synchronicity estimate, with an angular speed proportional to $\nu_n(t,nF_1)$.

The Figure 3 shows a sawtooth wave playing a E3 (normally 164.81Hz) slightly detuned to +20 cents (=166.73Hz). Each F0 spinner on the right has a yellow color due to the +20 cts deviation from the target frequency (E3=164.81Hz).

Consistently, the center of the lobes deviates above the reference pink line due to the slight detuning effect. The figure on the right shows the result using the cents scale processing of the contraction factor $\tilde{C}_{f_c}$ (f_c is set to 5 cents as in Fig. 1-right). This method significantly enhances accuracy and ensures equal sizing of the lobes on a cents scale. This is in contrast to the center figure, which represents the same signal but with $f_c = 6$,Hz (as in Fig. 1-center) and displays unequal lobe sizes.

To compare the precision stemming from the Snail analysis to the Fourier approach, Figure 3-left shows the loudness of the Fourier spectrum of the same signal.

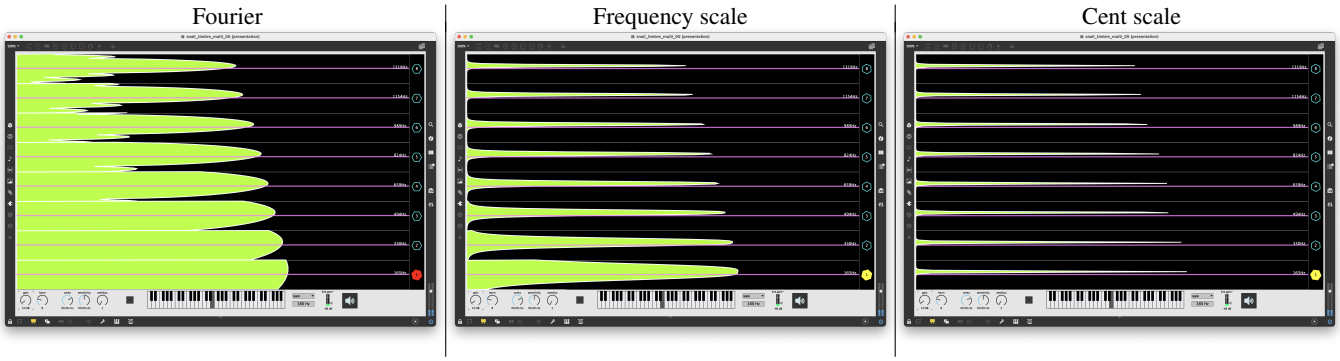


Figure 3: *Harmonicity renderings of a sawtooth wave with a fundamental frequency at 166.73Hz (E3 +20 cents), 8 visible harmonics. From left to right: Fourier (50ms, Hanning), Snail-Analyser (frequency precision: 6 Hz), Snail-Analyser (cent precision: 5 cents).*

3.2. Rapid prototyping with MAX

The prototype Software and the visualisation have been developed using the Max environment [12], a visual programming paradigm for interactive multimedia applications. A Max external object (MXO), loaded as a dynamic library in the environment at startup, encapsulates the Snail kernel library and computes for the signal processing chain. The real-time visualisation is then built using the Javascript for User Interface (or JSUI [13]) in Max, which has internal bindings to the Max graphics engine. Each analysis signal from the Max object is passed on to the JSUI object to build the rendering. This first approach proves reliable enough for the design of proof of concept real-time renderings, running approximately at 25 frames per second and requiring fast development iterations [6]. Thus, other views for future developments are yet to be tested. Once the user interface prototypes reach a final state, a desktop version of the prototype should also be converted to the JUCE environment [14].

4. CONCLUSION AND PERSPECTIVES

The Snail analysis process appears relevant to design applications specifically aimed at real-time representations of the timbre of nearly harmonic sounds. The proposed visual rendering incorporates reactive descriptors pertaining to the amplitude of spectral components, tuning, and harmonic synchronicity. These descriptors ensure accurate frequency representation, making this prototype a reliable initial development. Further work will focus on exploring this tool and refining visualisations in collaboration with expert musicians, integrating new descriptors to enhance both information and legibility, and designing a desktop application.

5. ACKNOWLEDGMENTS

The authors thank the workplan "France Relance" for the financial support of the ATRIM project 21-PRRD-0001-01-11-009. This project is collaborative work between Buffet Crampon and the STMS laboratory.

6. REFERENCES

[1] T. Hélie, "Analyse fréquentielle par démduolation de phase d'un signal acoustique," French Patent App. FR1455456A (2014), Int. Patent App. WO2015/189419 A1 (2015), Assignee: Centre National de la Recherche Scientifique, 2014.

[2] T. Hélie and C. Picasso, "The snail: a real-time software application to visualize sounds," in *Proc. DAFx*, 2017.

[3] T. Hélie, C. Picasso, R. Piéchaud, M. Jousserand, and T. Colinot, "Variations on the Snail-Analyser and its spectral estimates for the accurate tuning of musical sounds: a strobe tuner like display," in *31st ERK*, Portoroz, Slovenia, Sept. 2022.

[4] R. W. Young and L. Allen, "Stroboscope," 1938, US2286030A.

[5] G. E. Arends and R. S. Dobrose, "Strobe tuner," 1997, US5877443A.

[6] T. Hélie, C. Picasso, R. Piéchaud, M. Jousserand, and T. Colinot, "A Real-Time Cent Sensitive Strobe-like Tuning Software Based on Spectral Estimates of the Snail-Analyser," in *Sound and Music Computing Conference*, Stockholm, Sweden, June 2023, vol. 19, pp. 1–7.

[7] J. M. Grey, "Multidimensional perceptual scaling of musical timbres," *the Journal of the Acoustical Society of America*, vol. 61, no. 5, pp. 1270–1277, 1977.

[8] S. McAdams, S. Winsberg, S. Donnadieu, G. De Soete, and J. Krimphoff, "Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes," *Psychological research*, vol. 58, pp. 177–192, 1995.

[9] K. Siedenburg, I. Fujinaga, and S. McAdams, "A comparison of approaches to timbre descriptors in music information retrieval and music psychology," *Journal of New Music Research*, vol. 45, no. 1, pp. 27–41, 2016.

[10] V. Lostanlen, C. El-Hajj, M. Rossignol, G. Lafay, J. Andén, and M. Lagrange, "Time–frequency scattering accurately models auditory similarities between instrumental playing techniques," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2021, no. 1, pp. 1–21, 2021.

[11] F. Auger, P. Flandrin, Y.-T. Lin, S. McLaughlin, S. Meignen, T. Oberlin, and H.-T. Wu, "Time-frequency reassignment and synchrosqueezing: An overview," *IEEE Signal Processing Magazine*, vol. 30, no. 6, pp. 32–41, 2013.

[12] Website Cycling74, "Max home page," https://cycling74.com/products/max.

[13] Website Cycling74, "Max JSUI Home Page," https://docs.cycling74.com/max7/refpages/jsui.

[14] Website, "JUCE Home Page," https://www.juce.com.

PHYSICALLY INSPIRED SIGNAL MODEL FOR HARMONIUM SOUND SYNTHESIS

Ninad Puranik and Gary Scavone

Schulich School of Music, McGill University
Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)
Montreal, Canada
ninad.puranik@mail.mcgill.ca | gary.scavone@mcgill.ca

ABSTRACT

The hand harmonium is arguably the most popular instrument for vocal accompaniment in Hindustani music today. However, it lacks microtonality and the ability to produce controlled pitch glides, which are both important in Hindustani music. A harmonium sound synthesis model with a source-filter structure was previously presented by the authors in which the harmonium reed sound is synthesized using a physical model and the effect of the wooden enclosure is applied by a filter estimated from a recorded note. In this paper, we propose a simplified and perceptually informed signal model capable of real time synthesis with timbre control. In the signal model, the source is constructed as a band-limited waveform matching the spectral characteristics of the source signal in the physical model. Simplifications are suggested to parametrize the filter on the basis of prominent peaks in the filter frequency response. The signal model is implemented as a Pure Data [1] patch for live performance using a standard MIDI keyboard.

1. INTRODUCTION

While Asian free reed instruments such as the sheng, the sho and the khaen have existed for over 2000 years, the development of Western musical instruments using free reeds (e.g. harmonium, accordion, harmonica, etc.) did not occur until the end of 19th century. The harmonium, also known as the reed organ or pump organ, reached the peak of its popularity in the Western world during the latter half of the 20th century. Dwarkanath Ghose is generally credited to have modified the European harmonium to invent the Indian hand-bellowed harmonium (Fig. 1) in 1884 [2]. Although the use of the harmonium in Western music declined, the hand harmonium grew in popularity to become the instrument of choice for vocal accompaniment in North Indian (Hindustani) classical music today.

However, the use of the harmonium in Indian music has attracted a lot of criticism from musicologists [3]. Due to its standard keyboard design, only a discrete set of twelve notes per octave, typically spaced at intervals of one semitone can be played on a harmonium. This is inconsistent with the non-equal temperament system in Hindustani music that uses 22 notes (referred to as shrutis) in one octave. Additionally, the harmonium cannot produce continuous pitch glides from one note to another as would be possible in a string instrument like a violin. Such a lack of capability makes the harmonium unsuitable for producing essential ornaments in Hindustani music such as meend, andolan, and gamaka, which are elements similar to glissando, portamento, and vibrato in Western music.

Figure 1: *An Indian hand harmonium (top view).*

To address these limitations in a real harmonium, the authors recently proposed a physics-based synthesis system for harmonium sounds [4]. The proposed system used a source-filter structure in which a 1D physics-based model of a free reed interacting with the air flow acted as the source and the effect of the wooden enclosure of the instrument was approximated by an all-pole filter whose coefficients were estimated from a recorded harmonium sound.

While the proposed system produced convincing results, we observed some limitations that hindered its use for real-time synthesis in a live-performance context. Consequently, a simplified signal model that perceptually replicated the physics-based synthesis model was developed.

Section 2 provides a short literature review on free reeds while in section 3, we describe in brief the physics-based system for harmonium sound synthesis from our previous work. Section 4 defines the main requirements for an equivalent signal model and describes our implementation of the signal model in Pure Data.

2. RELATED WORK

A comprehensive review of prominent acoustic studies on free reed instruments has been provided by Cottingham [5]. Early acoustic studies on free reeds have focused on understanding how self-sustained oscillations are produced in a free reed. A series of experimental and theoretical studies by St. Hilaire [6, 7, 8] suggests that the inertial effect of the upstream air flow is responsible in setting up self-oscillations in free reeds. Experimental studies by Tarnopolsky [9, 10] and Ricot [11] agree with this theory. Millot and Baumann [12] have described a minimal 1D model and a numerical solution scheme to simulate the sound from any free reed instrument. In previous work [4], we have proposed adaptations

to the Millot-Baumann model to suitably match the physical setup and sound timbre of a Indian hand-harmonium.

3. SOURCE-FILTER BASED PHYSICAL MODEL OF A HAND-HARMONIUM

In the source-filter model used for the synthesis of voiced vowel sounds, a periodic glottal excitation signal determines the fundamental pitch (f0-frequency) while a linear filter representing the vocal tract imparts the timbral characteristics to the sound [13]. The source-filter model can be used effectively in situations where there is only a weak coupling between the source excitation and the resonator [14]. This is the case for the harmonium sound production. Since the metallic harmonium reeds are lightly damped, their frequency response is concentrated in a narrow band around their resonance frequencies. The effect of the wooden enclosure is hence analogous to the vocal tract. The use of a source-filter model to synthesize harmonium sounds is thus justified. The source and filter models that we used are described in the following subsections.

3.1. Physical model of the harmonium free reed

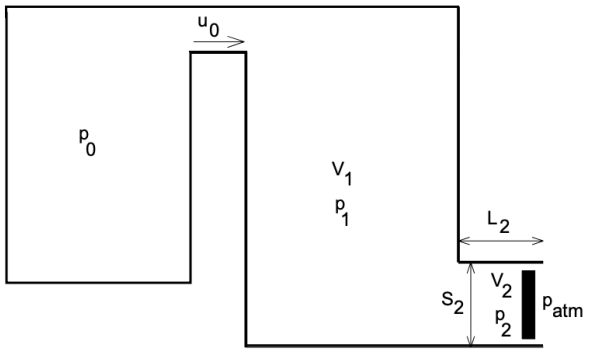
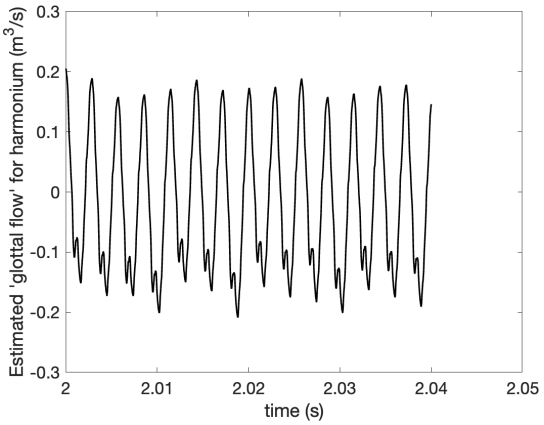


Figure 2: *Configuration of the Physical model.*

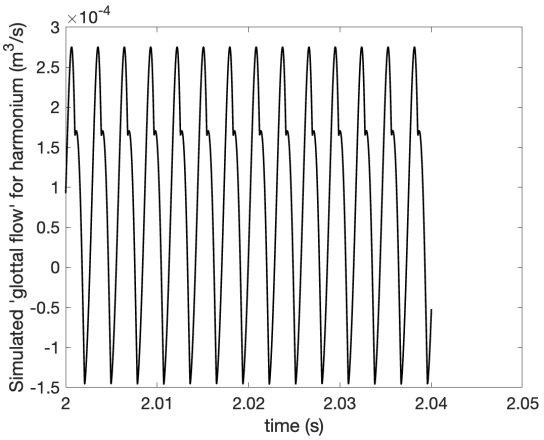
Figure 2 shows a schematic representation of the configuration of the physical model used to simulate the harmonium free-reed. The model assumes the region upstream of the reed (i.e. the reed chamber) to be a large volume V_1 where the pressure p_1 is uniform. The volume V_2, with cross-sectional area S_2 and length L_2, models the region near the reed where the air flows across the reed through a narrow jet. The reed itself was modeled as a sinusoidally oscillating lumped mass-spring-damper whose behavior is predominantly governed by the eigenfrequency (ω_0) and the quality factor (Q) parameters. The region downstream from the reed is exposed to atmospheric pressure. In the original Millot-Baumann model, the system was excited by the volume flow u_0 entering the volume V_1. In our adaptation, however, we added the chamber with pressure p_0 to represent the bellows pressure which indirectly controls the excitation signal u_0. With this change, we observed a better agreement between experimentally measured reed-chamber pressures and the corresponding p_1 values in the synthesis. It also allowed for the use of bellows pressure as a parameter to control the sound produced, like in a real harmonium. The governing equations for the system and the details of the numerical solution scheme are presented in [12, 4]. It should be noted that the bellows pressure p_0 and the reed eigenfrequency ω_0 are the only control parameters that vary while playing the harmonium. The note onset time and the stability of the numerical scheme was observed to be dependent on the other non-playing parameters.

3.2. Estimation of the wooden enclosure filter

The Iterative Adaptive Inverse Filtering (IAIF) algorithm developed by P. Alku [15] and implemented in the COVAREP Toolbox [16] can used to estimate the 'glottal flow' and the 'vocal tract' filter for a speech sound. For a harmonium sound, we expected the algorithm to estimate the analogous 'reed airflow' and 'wooden enclosure' filter respectively.



(a) *Estimated reed airflow.*



(b) *Simulated reed airflow.*

Figure 3: *A comparison of estimated vs simulated reed airflow.*

A comparison of the reed airflow estimated by the IAIF algorithm for a recorded harmonium note with the reed airflow simulated by the physical model is shown in Figs. 3a and 3b, respectively. The two signals display similar features. In particular, they show a discontinuity within each period that is assumed to occur when the reed passes from one side of its support plate to the other. The amplitude of the airflow signal estimated for the recorded sound is varying considerably, which is expected since the bellows pressure cannot be maintained perfectly constant in a real harmonium. Given the similarity, we proposed that the 'vocal

tract' filter estimated by the IAIF can be a good estimate of the 'wooden enclosure' filter.

4. SIGNAL MODEL AND IMPLEMENTATION IN PURE DATA

4.1. Source approximation

As mentioned in the previous section, the numerical stability for the physical model was dependent on the non-playing parameters, such as reed-chamber volume, reed clearance, etc. Hence, in order to change the playing frequency for a note, these parameters had to be changed in addition to the reed eigenfrequency (ω_0) and they had to be determined by manual trial-and-error, which is not possible in a live-performance context. Moreover, it was observed that the filter has a much stronger influence on the synthesized sound timbre than the source. Hence, a simplified signal model capable of approximately generating the reed source signal at the different playing frequencies would produce perceptually similar results while eliminating the problems of numerical stability and computation cost involved in the physical model.

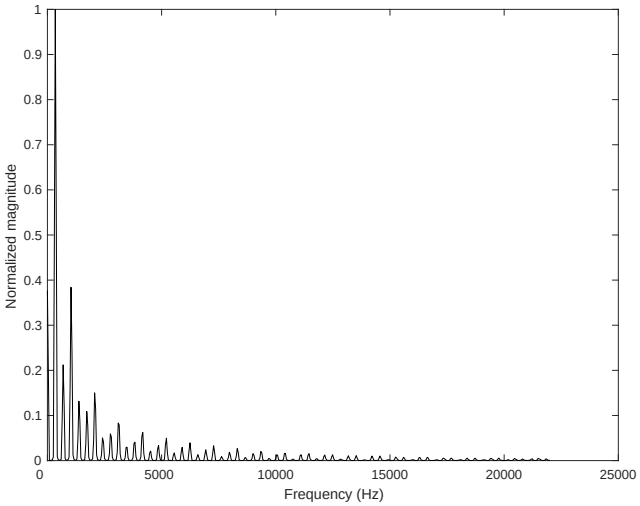


Figure 4: *Normalized LTAS for the reed source signal simulated using physics model.*

The normalized Long Term Average Spectrum (LTAS) for an 'F4' note (349 Hz) simulated by the physical model can be seen in Fig. 4. A band-limited signal with harmonic weights equal to the peak heights in the normalized LTAS was used to model the reed source signal. In our Pure Data implementation (Fig. 6), this was achieved by using a 'sinesum' message with the harmonic weights of the first 37 peaks in the normalized LTAS to populate a wavetable. Notes with different frequencies were synthesized by reading the wavetable at the particular frequencies.

4.2. Filter approximation

The all-pole filter estimated by the IAIF algorithm had 49 coefficients. However, the magnitude filter response for different harmoniums tested was observed to have 8-9 peaks. Hence, a close approximation of the IAIF estimated filter using a series of 10 second order sections (i.e. 10 biquad elements in a cascade in the Pure Data implementation) was deemed feasible. In MATLAB, the 'stmcb' algorithm [17] was used to estimate an approximate filter with 10 poles and 10 zeros and the 'tf2sos' function was then used to determine the second order filter coefficients. Also, a smooth frequency response curve passing through all the peaks and valleys was obtained using the 'pchip' algorithm [18] and the 'yulewalk' algorithm [19] was used to construct a filter with that frequency response. Although there was a larger error in the second approach for the filter approximation, the error was observed to be perceptually insignificant in the synthesis while allowing for the parametrization of the filter on the basis of just 9 peak and 9 valley points. Such a parametrization would be helpful to construct arbitrary frequency responses that can be used to synthesize harmonium sounds with a variety of timbres. The IAIF filter response and its approximations using the two methods described can be seen in Fig. 5.

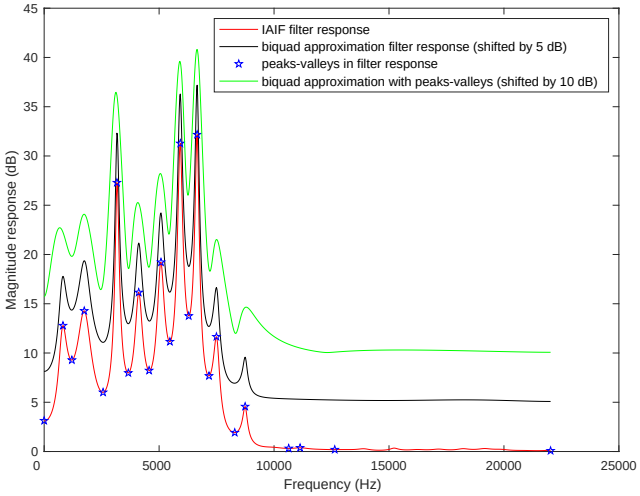


Figure 5: *Frequency response of estimated and approximated wooden enclosure filters for a concert quality harmonium.*

4.3. Effect of bellows pressure

The primary role of the bellows pressure is to modulate the amplitude of the sound. Additionally, in the physical model as well as in the case of recorded harmonium sounds, it is observed that the note fundamental frequency (f0) slightly varies when the bellows pressure is changed. Although the variation is very small (typically less than 1-2 Hz), it is still perceptible and is used by harmonium players to create a perceptual effect of continuous pitch glides through specific bellowing techniques. The most prominent effect of this behaviour is the slight increase in playing frequency as the air leaks out of the harmonium, thus reducing the bellows pressure. In the Pure Data implementation, the bellows pressure is used to control the amplitude envelope of the sound. In addition, it is added to the 'f0' frequency to change the rate at which the wavetable is read to mimic the frequency variation described.

5. CONCLUSIONS

The paper describes a signal model for synthesizing Indian hand harmonium sounds. The signal model uses a source-filter structure similar to a previously developed physical model. The source

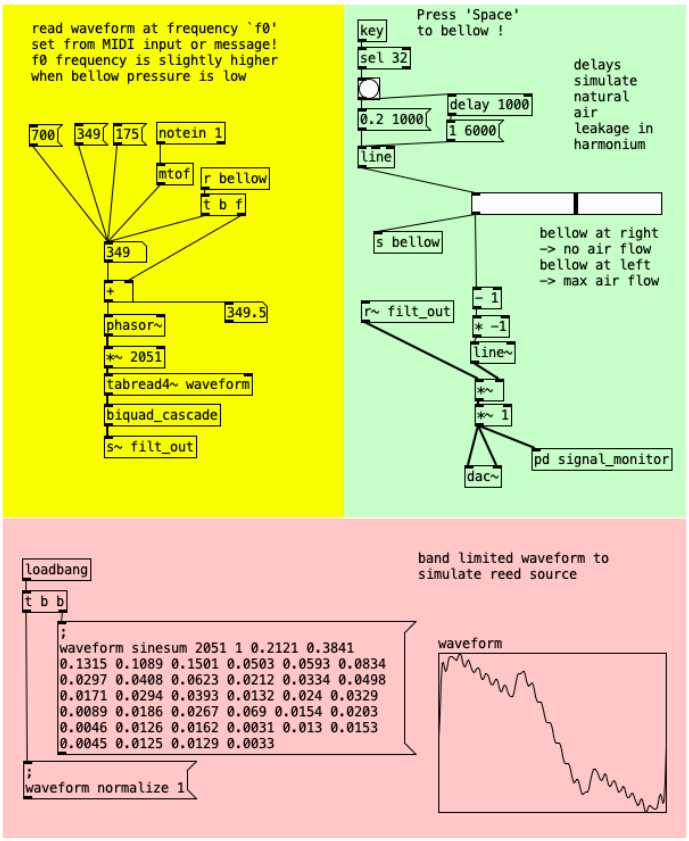


Figure 6: *Pure Data patch to implement the signal model.*

is constructed as a band-limited waveform that has similar spectral characteristics as the physical model. Simplifications are suggested to parametrize the filter on the basis of the prominent peaks in the filter frequency response. Different timbres of the harmonium can be realized by hand-crafting or estimating the filter parameters. The simplified model implemented in Pure Data presents the possibility of real-time synthesis and live performance using the virtual instrument.

6. ACKNOWLEDGMENTS

Financial support for this work was provided by the Schulich School of Music, McGill University, Montreal. Funding for attending the conference was provided by the Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT).

7. REFERENCES

[1] Miller Puckette et al., "Pure data: another integrated computer music environment," *Proceedings of the second intercollege computer music concerts*, pp. 37–41, 1996.

[2] Birgit Abels, *The Harmonium in North Indian Music*, New Age Books, 2010.

[3] Matt Rahaim, "That Ban(e) of Indian Music: Hearing Politics in The Harmonium," *The Journal of Asian Studies*, vol. 70, no. 3, pp. 657–682, Aug. 2011.

[4] Ninad Vijay Puranik and Gary P Scavone, "Physical modelling synthesis of a harmonium," in *Proceedings of Meetings on Acoustics, Fourth Vienna Talk on Music Acoustics*. Acoustical Society of America, 2022, vol. 49, p. 035015.

[5] James Cottingham, "Acoustics of free-reed instruments," *Physics Today*, vol. 64, no. 3, pp. 44–48, Mar. 2011.

[6] Arthur O. St. Hilaire, Theodore A. Wilson, and Gordon S. Beavers, "Aerodynamic excitation of the harmonium reed," *Journal of Fluid Mechanics*, vol. 49, no. 4, pp. 803–816, Oct. 1971.

[7] Arthur O. St. Hilaire and P. G. Vaidya, "Finite amplitude analysis of a flow-structure interaction problem," *Journal of Fluid Mechanics*, vol. 67, no. 2, pp. 377–396, 1975.

[8] Arthur O. St. Hilaire, "Analytical prediction of the non-linear response of a self-excited structure," *Journal of Sound and Vibration*, vol. 47, no. 2, pp. 185–205, 1976.

[9] A. Z. Tarnopolsky, N. H. Fletcher, and J. C. S. Lai, "Oscillating reed valves—An experimental study," *The Journal of the Acoustical Society of America*, vol. 108, no. 1, pp. 400–406, July 2000.

[10] Alex Z Tarnopolsky, JCS Lai, and Neville H Fletcher, "Flow structures generated by pressure-controlled self-oscillating reed valves," *Journal of sound and vibration*, vol. 247, no. 2, pp. 213–226, 2001.

[11] Denis Ricot, René Caussé, and Nicolas Misdariis, "Aerodynamic excitation and sound production of blown-closed free reeds without acoustic coupling: The example of the accordion reed," *The Journal of the Acoustical Society of America*, vol. 117, no. 4, pp. 2279–2290, Apr. 2005.

[12] Laurent Millot and Clément Baumann, "A Proposal for a Minimal Model of Free Reeds," *Acta Acustica united with Acustica*, vol. 93, pp. 122–144, Jan. 2007.

[13] Gunnar Fant, *Acoustic Theory of Speech Production: With Calculations based on X-Ray Studies of Russian Articulations*, De Gruyter, 1971.

[14] Marcelo Caetano and Xavier Rodet, "A source-filter model for musical instrument sound transformation," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2012, pp. 137–140, ISSN: 2379-190X.

[15] Paavo Alku, "Glottal wave analysis with pitch synchronous iterative adaptive inverse filtering," *Speech Communication*, vol. 11, no. 2, pp. 109–118, 1992, Eurospeech '91.

[16] Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer, "Covarep—a collaborative voice analysis repository for speech technologies," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 960–964.

[17] K Steiglitz and L McBride, "A technique for the identification of linear systems," *IEEE Transactions on Automatic Control*, vol. 10, no. 4, pp. 461–464, 1965.

[18] Ralph E Carlson and Frederick N Fritsch, "Monotone piecewise bicubic interpolation," *SIAM journal on numerical analysis*, vol. 22, no. 2, pp. 386–400, 1985.

[19] Benjamin Friedlander and Boaz Porat, "The modified yule-walker method of arma spectral estimation," *IEEE Transactions on Aerospace and Electronic Systems*, , no. 2, pp. 158–173, 1984.

P-RAVE: IMPROVING RAVE THROUGH PITCH CONDITIONING AND MORE WITH APPLICATION TO SINGING VOICE CONVERSION

Shahan Nercessian

iZotope, Inc.
Boston, MA, USA
shahan@izotope.com

ABSTRACT

In this paper, we introduce means of improving fidelity and controllability of the RAVE generative audio model by factorizing pitch and other features. We accomplish this primarily by creating a multi-band excitation signal capturing pitch and/or loudness information, and by using it to FiLM-condition the RAVE generator. To further improve fidelity when applied to a singing voice application explored here, we also consider concatenating a supervised phonetic encoding to its latent representation. An ablation analysis highlights the improved performance of our incremental improvements relative to the baseline RAVE model. As our primary enhancement involves adding a stable pitch conditioning mechanism into the RAVE model, we simply call our method *P-RAVE*.

1. INTRODUCTION

Deep generative audio models aim to reconstruct and/or synthesize novel audio by learning its underlying data distribution. Since the inception of WaveNet [1], models have made considerable gains to improve fidelity, and achieve state-of-the-art realism in many domains. However, they are still largely considered too complex for widespread use, and offer limited controllability to end users.

Recently, the RAVE approach was introduced [2]. This variational autoencoder (VAE) has garnered excitement in the audio community due to its expressive synthesis, stable training procedure, favorable performance, and tractability for streaming applications running on edge devices [3], while modeling audio at sampling rates suitable for music production (i.e. $\geq$ 44.1 kHz). Despite this breakthrough, its baseline formulation can be prone to pitch glitches, especially when applied to out-of-domain input samples. Its latent representation may also still conflate timbral, pitch, and other factors without additional mechanisms to steer their disentanglement, limiting controllability.

Meanwhile, modern voice AI techniques have become emergent in research and pop culture [4]. Singing voice conversion (SVC) is one such application, whose goal is to transform sung material to match the timbre of some target singer while maintaining the source performance. SVC methods such as FastSVC [5] and [6] condition waveform generation on a harmonic excitation signal to counteract the fact that most neural generators lack sufficient pitch stability otherwise [7]. SawSing [8] considered a sawtooth excitation, and in our own work [9], we considered a hybrid end-to-end approach, where a differentiable WORLD synthesizer creates an initial synthesized output from inferred features, which is then further refined via a black-box postnet. Few SVC approaches are amenable to real-time streaming applications [10], so naturally, it is of interest to explore how the RAVE model can be refined for this use case.

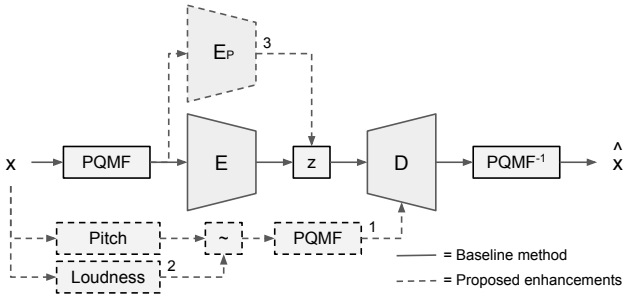


Figure 1: *RAVE model and proposed enhancements in P-RAVE.*

In this work, we offer improvements to the RAVE model. We apply insights from the SVC community to improve tonal signal reconstruction and/or generation in RAVE in a multi-band generator context [11], effectively conditioning its generator on excitation signals capturing pitch and/or loudness information. Accordingly, we call our method *P-RAVE*. Approaches are exemplified for an SVC application, and within this context, we also consider whether the model can benefit from supervised encodings of linguistic content. A byproduct of this work is an efficient phoneme recognizer that learns a feature representation from the time-domain waveform. Our goals are two-fold: we would like to improve the outputs of RAVE while maintaining its advantages, and to disentangle features such as pitch from its latent representation so that they are not conflated in the latent space and/or so that they can be controlled explicitly. In doing so, we are inherently investigating whether and/or how RAVE can be adapted for SVC. We illustrate the benefits of our enhancements relative to a standard RAVE baseline. We organize our paper as follows: Section 2 describes our proposed method, Section 3 reports experimental results, and Section 4 draws conclusions.

2. PROPOSED METHOD

The RAVE model, as well as our proposed additions in P-RAVE, are illustrated in Figure 1. Generally, an input signal x is mapped to a latent encoding z. The decoder aims to invert z, yielding the reconstructed waveform $\hat{x}$. The architecture utilizes a 16-band Pseudo Quadrature Mirror Filterbank (PQMF) [11], which aids model efficiency by allowing the core architecture to operate internally at a fraction (i.e. 1/16th) of the audio system sampling rate f_s. As in [2], we consider f_s = 48 kHz in this work. Signal reconstruction involves generation of an audio waveform from a suit-

able encoding z, along with specifications of any available control signals. Novel signal generation would additionally involve prediction of a relevant latent trajectory (via a second "prior" model). Our focus leans to the former task without restricting the latter.

P-RAVE improves upon RAVE by 1) FiLM conditioning [12] the RAVE generator with a multi-band harmonic excitation signal, 2) incorporating loudness information into said excitation, and 3) appending a supervised phonetic encoding alongside the learned latent representation, considering our particular interest in singing voice applications. We outline the motivation and implementation of each enhancement.

2.1. Harmonic excitation and FiLM conditioning

In order to provide stability and controllability of pitch, we leverage pitch-driven excitations as conditioning signals and adapt them to the multi-band RAVE generator. Combining the excitation generation approaches in [6, 8, 13], we generate the excitation e as

$$e[n] = \begin{cases} \eta[n] & \text{if } f_0[n] = 0 \\ \sum_{k=1}^{K} \frac{1}{k} \sin(\phi_k[n]) & \text{otherwise} \end{cases} \quad (1)$$

$$K = \lfloor \frac{f_s}{2 f_0[n]} \rfloor \quad (2)$$

$$\phi_k[n] = \phi_k[n-1] + 2\pi k \frac{f_0[n]}{f_s} \quad (3)$$

where $\eta \sim \mathcal{N}(0, 1)$ and f_0 is a fundamental frequency contour that can be user-specified or estimated directly from an input signal x. For the latter case, pitch is detected at a specified interval (we arbitrarily use a stride of 128 samples in this work), and upsampled to audio rate using a basic zeroth order interpolation in order to match the input audio waveform dimension. To this end, we leverage the `torchyin` library, which among other conveniences, ensures that the entire pipeline is constructed in PyTorch and can leverage the GPU more effectively during training.

We proceed by creating the multi-band excitation representation $e_{PQMF} = PQMF(e)$. Naturally, these sub-bands still operate at a faster sampling rate than the encoding z (in fact, it would match that of the hypothetical multi-band output signal estimate $\hat{x}_{PQMF}$). Accordingly, we apply FiLM conditioning to layers of the multi-band generator (decoder), as illustrated in Figure 2. We apply successive downsampling layers to e_{PQMF} according to the upsampling factors of each generator layer to ensure that they operate at the same rate. While downsampling could have been accomplished without trainable parameters, we opt to use strided convolutional layers instead, maintaining a constant 16-channel count for each downsampling stage. The outputs of each downsampling stage are subjected to respective 1x1 convolutional layers whose channels equal twice that of the corresponding generator upsampling layer which they are paired with. Output channels are split in half to form the scale and offset terms for FiLM conditioning. For an arbitrary scale γ, offset β, and upsampling layer y, the FiLM-conditioned output is given by

$$y_{FiLM} = \gamma \odot y + \beta \quad (4)$$

When scales and offsets are equal to unity and zero, respectively, FiLM conditioning sites act as pass-throughs. Unlike DDSP [14], note that we are not imposing for the model output to be strictly monophonic per se. The model is still fully capable of generating polyphonic audio, and therefore, we entirely maintain the generality of the RAVE system. We are simply adding a conditioning signal to steer generator upsampling layers, and if the model did not find useful information contained within it, could choose to ignore it. Nonetheless, when applied to intrinsically monophonic applications (e.g. solo voice), we expect that the model would learn to interpret it as an excitation signal (so we continue to refer to it as such), and to non-linearly filter it as a neural source-filter [13].

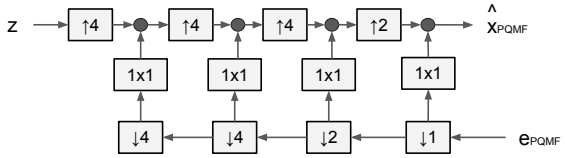


Figure 2: *Proposed multi-band FiLM conditioning applied in the P-RAVE generator. Solid black dots represent conditioning sites.*

2.2. Injection of loudness information into the conditioning

We can also incorporate loudness information into our pitched excitation signal. This is to say that its signal strength can be set to a user-specified value (a constant loudness, amplitude envelope, and/or mapping to MIDI velocity, as in [15]), or to match that of x. In the case of the latter, much like the f_0 computation in Section 2.1, we achieve this by measuring the frame-level root-mean-square (RMS) of x at some notional stride and upsampling it to full resolution, yielding the desired loudness trace L_0. If we similarly extract the RMS of e, yielding L_e, we can embed loudness information into a loudness-adjusted conditioning signal e_L via

$$e_L[n] = \left(\frac{L_0[n] + \epsilon}{L_e[n] + \epsilon} \right) e[n] \quad (5)$$

where $\epsilon = 10^{-5}$ is used for numerical stability. Accordingly, the multi-band excitation signal is then $e_{PQMF} = PQMF(e_L)$.

2.3. Supervised phonetic encoding

When trained for a voice application, the encoder is tasked with capturing not only pitch, loudness, and tone in a global sense, but also timbral changes which vary as a function of the phonetic unit being uttered. As this may prove challenging to accomplish sufficiently in a purely unsupervised manner, the final enhancement considered here concatenates the learned latent representation with a phonetic posteriorgram (PPG) capturing linguistic content.

We train a phonetic encoder on the TIMIT dataset [16] in a supervised manner. One subtlety here is that this dataset is natively sampled at 16 kHz, containing 8 kHz of bandwidth. Therefore, we upsample the data to audio rate (48 kHz in this case), and only consider the lower 5 PQMF sub-bands as input to the phonetic encoder, ideally covering 7.5 kHz with sufficient roll-off by 8 kHz. We use the condensed 40-class phonetic dictionary for the TIMIT dataset (39 phonemes and a silence class) [17], and train the phonetic encoder for 250K training steps before freezing it for the remainder of system training. Maintaining a total encoding size of 128, this leaves 88 latent dimensions to be learned in an unsupervised manner. Input and output sizes aside, the phonetic encoder E_P architecture is identical to that of the unsupervised encoder E. We reduce the number of channels in the two encoders by 50%, such that the number of parameters of their composite is effectively the same as in the encoder of the baseline RAVE model.

3. EXPERIMENTAL RESULTS

For subjective listening, we refer readers to our demo website at https://sites.google.com/izotope.com/prave-demo.

Generally, we observe that the VAE framework offers a sufficient information bottleneck in its latent space [18] for providing the speaker disentanglement needed for SVC. Accordingly, in order to illustrate the effectiveness of our proposed methods, we train models and analyze their robustness in this context. Models were trained on approximately two hours of internal singing voice data of a single target singer. Four different conversion models were considered: a baseline RAVE system, and three P-RAVE systems where we incrementally add our proposed enhancements, as per their enumeration in Section 2. We follow the training strategy outlined in [2], using a batch size of 8, Adam optimizer and its adversarial objective function. In the initial "warm-up" training phase, the encoder and decoder models are optimized jointly for 1M training steps, with the adversarial loss terms omitted. The (latent) encoder is then frozen and the decoder undergoes an additional 2M training steps which attempts to minimize the full objective.

We summarize our quantitative ablation analysis in Table 1, reporting the multi-spectrogram loss (MSL), the number of components needed to summarize 99% of the latent manifold (M_{99}) [2]. We see that our P-RAVE variants considerably improve reconstruction performance relative to the RAVE baseline, as measured by the MSL. Explicit incorporation of loudness information in P-RAVE (1+2) improves upon P-RAVE (1). Meanwhile, P-RAVE (1+2+3) technically sees slightly degraded quantitative performance on the self-reconstruction task relative to P-RAVE (1+2). We attribute this to the fact that the learned latent representation is catered to the target singer (in-domain distribution), while the PPG is a vocalist-independent representation to be leveraged by any source singer (out-of-domain distribution). Therefore, its inclusion still has favorable implications for our ultimate goal of the conversion task. P-RAVE (1) and P-RAVE (1+2) create significantly more compact latent feature representations relative to the baseline RAVE model, as the pitched and/or leveled excitation reduces the burden on the unsupervised encoder to fully model the feature space. The feature space is effectively reduced by one component between P-RAVE (1) and P-RAVE (1+2), hinting that conceivably, there may have indeed been a single latent dimension capturing loudness variations in the data. Interestingly, addition of the PPG in P-RAVE (1+2+3) considerably increases M_{99}. Though seemingly unintuitive, we explain this by noting that the information contained within PPGs arguably reflects the majority of the voice modeling task beyond pitch and loudness. Therefore, by now offloading the unsupervised encoder of its primary modeling "duties", P-RAVE (1+2+3) reduces the posterior collapse effect overall, allowing its unsupervised encoder to concentrate its degrees of freedom to modeling a smaller subspace of the variability in the data in a way that better matches the prior. This can be confirmed by observing that the Kullback-Liebler divergence against the prior for RAVE and P-RAVE (1+2+3) are 12.01 and 1.749, respectively.

Next, we analyze the preservation and controllability of conditioning features across different models for both self-reconstruction (in-domain source vocalist) and conversion (out-of-domain source vocalist), as listed in Table 2. Specifically, we compare conditioning features extracted from source and synthesized performances, measuring their average absolute deviation in fundamental frequency (ΔF in Hz) and loudness (ΔL in dB), and average categorical cross-entropy between source and synthesized PPGs (ΔCE in nats). In the in-domain case, we see that the baseline RAVE model performs decently, though features are better preserved in P-RAVE variants, where P-RAVE (1+2+3) appears to provide the best balance across all features. Within the conversion context, we further discern between whether or not we apply a pitch shift to f_0 so that the converted result is reflective of the register of the vocalist, and report results for both cases. Here, the baseline RAVE model struggles considerably, as it cannot maintain consistent pitch when the source register does not match the target data, and moreover, does not possess an explicit mechanism for applying a pitch shift if it were needed to accomplish a convincing conversion. Again, we see that P-RAVE (1+2+3) is better suited for the application, with outputs roughly achieving their target values across all features.

4. CONCLUSIONS

In this paper, we suggested additions to the RAVE model which improved fidelity and controllability of the generative audio model, and applied it to a singing voice application. In future work, we would like to add further refinements in order to improve fidelity. For example, we may consider a loss term that encourages the model to produce Mel spectrogram-like representations at an intermediate generator layer, or integrate aspects of the very recent developments in [19]. We are also interested to get a better sense of the prominent factors the latent space has learned when it is relieved of modeling pitch, loudness, and phonetic content, and to envision what other forms of transformative audio processing this may be able to unlock. Lastly, we would like to investigate further application of our enhancements in the context of novel tonal content creation.

Table 1: *Quantitative ablation analysis comparing RAVE to our various enhancements in P-RAVE.*

Experiment	MSL	$M_{99\%}$
RAVE	8.568	9
P-RAVE (1)	7.267	4
P-RAVE (1+2)	**7.175**	**3**
P-RAVE (1+2+3)	7.227	14

Table 2: *Comparison of conditioning features between source and synthesized performances.*

	Reconstruction			Conversion (without/with pitch shift)		
Experiment	ΔF (Hz)	ΔL (dB)	ΔCE (nats)	ΔF (Hz)	ΔL (dB)	ΔCE (nats)
RAVE	3.351	3.667	0.0564	61.43 / 103.92	4.918 / 4.923	0.0769 / 0.768
P-RAVE (1)	**0.443**	2.362	0.0556	0.682 / 1.321	4.560 / 2.786	0.0731 / 0.071
P-RAVE (1+2)	0.596	**1.764**	0.0562	**0.237 / 0.791**	**1.922 / 1.958**	0.0706 / 0.0716
P-RAVE (1+2+3)	*0.628*	*1.993*	**0.0515**	*0.277 / 0.823*	*3.3922 / 2.0625*	**0.0696 / 0.0676**

5. REFERENCES

[1] A. van den Oord et al., "WaveNet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[2] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv:2111.05011*, 2021.

[3] A. Caillon and P. Esling, "Streamable neural audio synthesis with non-causal convolutions," in *Int. Conf. on Dig. Aud. Eff. (DAFx)*, 2022, pp. 320–327.

[4] H.S. Choi, J. Yang, J. Lee, and H. Kim, "NANSY++: Unified voice synthesis with neural analysis and synthesis," in *Int. Conf. on Learn. Rep. (ICLR)*, 2023.

[5] S. Liu et al., "FastSVC: Fast cross-domain singing voice conversion with feature-wise linear modulation," in *IEEE Int. Conf. on Mul. and Ex. (ICME)*, 2021, pp. 1–6.

[6] H. Guo, Z. Zhou, F. Meng, and K. Liu, "Improving adversarial waveform generation based singing voice conversion with harmonic signals," in *IEEE Int. Conf. on Ac., Speech and Sig. Proc. (ICASSP)*, 2022, pp. 6657–6661.

[7] B. Di Giorgi, M. Levy, and R. Sharp, "Mel spectrogram inversion with stable pitch," in *Int. Soc. for Mus. Inf. Ret. Conf. (ISMIR)*, 2022.

[8] D.Y. Wu and Others, "DDSP-based singing vocoders: A new subtractive based synthesizer and a comprehensive evaluation," in *Int. Soc. for Mus. Inf. Ret. Conf. (ISMIR)*, 2022.

[9] S. Nercessian, "Differentiable WORLD synthesizer-based neural vocoder with application to end-to-end audio style transfer," in *154th Aud. Eng. Soc. Conv. (AES)*, 2023.

[10] S. Nercessian et al., "Real-time singing voice conversion plug-in," in *Int. Conf. on Dig. Aud. Eff. (DAFx)*, submitted, 2023.

[11] G. Yang et al., "Multi-band Melgan: Faster waveform generation for high-quality text-to-speech," in *IEEE Work. on Spok. Lang. Tech. (SLT)*, 2021, pp. 492–498.

[12] E. Perez et al., "FiLM: Visual reasoning with a general conditioning layer," in *AAAI Conf. on Art. Int.*, 2018.

[13] X. Wang and J. Yamagishi, "Using Cyclic Noise as the Source Signal for Neural Source-Filter-Based Speech Waveform Model," in *Interspeech*, 2020, pp. 1992–1996.

[14] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Int. Conf. on Learn. Rep. (ICLR)*, 2020, pp. 26–30.

[15] L. Renault, R. Mignot, and A. Roebel, "Differentiable piano model for MIDI-to-audio performance synthesis," in *Int. Conf. on Dig. Aud. Eff. (DAFx)*, 2022, pp. 233–239.

[16] J. S. Garapolo et al., *TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1*, Linguistic Data Consortium, Philadelphia, 1993.

[17] T.P. Van, H.N. Thanh, and T.M. Thanh, "Improving phonetic recognition with sequence-length standardized MFCC features and deep bi-directional LSTM," in *NAFOSTED Conf. on Inf. and Comp. Sci. (NICS)*, 2018, pp. 322–325.

[18] K. Qian et al., "AutoVC: Zero-shot voice style transfer with only autoencoder loss," in *Int. Conf. on Mach. Learn.*, 2019.

[19] N. Devis et al., "Continuous descriptor-based control for deep audio synthesis," in *IEEE Int. Conf. on Ac., Speech and Sig. Proc. (ICASSP)*, 2023.