OPEN ACCESS Freely available online

CC BY

R. Diaz, C. De La Vega Martin and M. Sandler, "Data-Driven Modeling of String and Membrane Dynamics Using Interpretable Neural Architectures," *J. Audio Eng. Soc.*, vol. 73, no. 11, pp. 734–746 (2025 Nov.).
https://doi.org/10.17743/jaes.2022.0234.

Data-Driven Modeling of String and Membrane Dynamics Using Interpretable Neural Architectures

RODRIGO DIAZ,* CARLOS DE LA VEGA MARTIN, AND MARK SANDLER, *AES Fellow*
(r.diazfernandez@qmul.ac.uk) (c.delavegamartin@qmul.ac.uk) (mark.sandler@qmul.ac.uk)

Centre for Digital Music, Queen Mary University of London, London, UK

Simulating the dynamics of strings and membranes is a central challenge in sound synthesis that is traditionally tackled with numerical methods like modal synthesis, which are efficient and accurate when the governing partial differential equations and parameters are known. However, when such information is unavailable, data-driven approaches become essential. Neural networks offer a viable alternative, with recurrent architectures providing real-time synthesis but often suffering from training instability and limited scalability. State-space models address these issues by combining linear dynamics with nonlinear transformations across sequences, enabling parallelization and greater efficiency. This paper introduces a data-driven framework for modeling vibrational dynamics, focusing on interpretability, efficiency, and accuracy. The proposed approach combines a linear time-invariant process with a neural modulation network. The linear time-invariant process captures global dynamics via optimized eigenvalues, allowing direct control over vibrational characteristics such as frequency and decay. The modulation network refines these dynamics by introducing local nonlinear adjustments, ensuring adaptability to complex behaviors. This architecture, therefore, unifies efficient learning, interpretable control, and accurate reconstruction of nonlinear vibrational dynamics.

0 INTRODUCTION

Modeling dynamics of strings and membranes is a fundamental problem in sound synthesis, particularly for musical applications such as virtual instrument design and sound design. The complex vibrational behavior of these systems forms the basis of the sounds generated by musical instruments such as guitars, violins, drums, and cymbals. Modeling these behaviors not only enables realistic sound reproduction, but also empowers sound designers and musicians to experiment with novel sonic possibilities, such as altering material properties, tuning, or decay characteristics.

Traditional approaches to modeling such instruments often rely on physical formulations, such as partial differential equations (PDEs), which describe the behavior of vibrating structures. Although in many cases linearity can be assumed, this does not always provide an adequate model; hence, nonlinear variations of these PDEs are solved by numerical integration methods such as finite difference or finite element techniques.

Modal-based methods, for example, use the eigenmodes of the system to simulate dynamics efficiently [1]. These approaches enable computational optimizations, such as evaluating spatial transformations only at points of interest and performing temporal integration via efficient filter structures [2]. Such techniques are well suited when the system's geometry, boundary conditions, and material properties are fully known. However, in many practical scenarios, this information is either unavailable or only partially known.

In these cases, data-driven neural approaches offer a compelling alternative by learning system dynamics directly from observed trajectories. Several families of neural models have been developed for this purpose. Among these, neural operators and, in particular, the Fourier neural operators [3] have shown promise in capturing nonlinear behavior in strings and membranes [4]. Another neural approach, and more closely related to the one used here, draws on Koopman theory [5]. Koopman-inspired networks have been used to linearize nonlinear dynamics using linear filters coupled with nonlinear lifting blocks for modeling time-varying audio effects [6]. The authors' previous work [7] also leveraged these types of networks to model the dynamics of tension-modulated strings.

Other notable approaches, include physics-informed neural networks [8] and neural ordinary differential equa-

*To whom correspondence should be addressed, email: r.diazfernandez@qmul.ac.uk.

tions (ODEs) [9]. Physics-informed neural networks incorporate physical priors into the learning process but require explicit knowledge of the governing PDEs and therefore are not directly comparable with fully data-driven methods. Neural ODEs offer greater flexibility in this regard but rely on explicit ODE solvers that can become computationally prohibitive when applied to high-dimensional systems or high sampling rates, particularly in the absence of model linearization or compression.

The goal of this paper is to bridge the gap between traditional physics-based models and modern data-driven techniques by proposing a hybrid framework that combines the interpretability of classical methods with the flexibility and scalability of neural network approaches.[1] This work builds upon the authors' earlier model [7], extending its scope and generality while refining the architectural design.

This paper is structured as follows. SEC. 1 provides an overview of Koopman-based methods and state-space models (SSMs), with a brief discussion of classical approaches to establish context. SEC. 2 introduces the proposed autoencoder framework focusing on eigenvalue optimization and its application to string and membrane dynamics. SEC. 3 outlines the data sets, architecture configurations, and training details used in this study's experiments. SEC. 4 evaluates the models, comparing their performance, interpretability, and robustness through experiments and ablation studies. Finally, SEC. 5 summarizes the contributions and highlights potential future directions.

1 BACKGROUND

Modeling the dynamics of vibrating systems such as strings and membranes requires tools capable of capturing both linear and nonlinear behaviors. Traditional techniques, such as time-invariant and short-time invariant autoregressive models and Prony-based methods (e.g., MUSIC, ESPRIT) [10, 11], have been widely used to identify dominant modes in linear and nonlinear systems. Modal analysis, a classical approach, assumes that the dynamics of a system can be represented as a linear combination of modes, each associated with a specific frequency and decay rate.

Although these methods excel for systems with well-behaved linear dynamics, they are limited in the presence of strong nonlinearities or time-varying behaviors. For example, tension-modulated strings or struck membranes exhibit complex interactions between modes that require more sophisticated techniques.

1.1 Koopman-Based Methods

Koopman theory provides a powerful framework for representing nonlinear dynamical systems as linear systems in an infinite-dimensional space of observables [12]. This allows nonlinear dynamics to be studied using tools developed for linear systems, such as eigenvalue and eigenfunction analysis. However, practical applications require finite-dimensional approximations of the Koopman operator, which introduce challenges in balancing accuracy and tractability.

This theoretical foundation has inspired data-driven techniques, such as dynamic mode decomposition (DMD), which approximates the Koopman operator from time series data [13]. Extensions of DMD, such as extended DMD [14] and Hankel DMD [15], aim to capture nonlinear behaviors by expanding the observable space or leveraging time-delayed snapshots, respectively. Extended DMD augments the state with nonlinear transformations, allowing for a more expressive approximation of the Koopman operator. Hankel DMD constructs structured matrices that incorporate temporal correlations, enabling the extraction of coherent modes over time.

Although Koopman-based methods provide a rigorous theoretical foundation, their practical application is often limited by the linear assumptions inherent in the extended space. For example, methods such as online DMD [16] assume local linearity within fixed blocks of data. If these blocks are too large, fast-changing nonlinearities cannot be captured effectively, particularly for systems with highly localized initial conditions. These limitations underscore the need for more flexible approaches.

1.2 Deep Neural State-Space Models

Neural networks offer an appealing solution for modeling nonlinear vibrational systems. Recurrent neural networks have been extensively explored for this purpose due to their ability to perform low-latency, sample-by-sample synthesis, which is essential for high-resolution audio signals (e.g., above 16 kHz). However, recurrent neural networks suffer from training difficulties caused by nonlinearities introduced at each time step that can lead to instability and poor scalability for long sequences [17, 18].

Deep neural SSMs, particularly diagonal SSMs (DSSMs), offer a complementary approach to modeling dynamical systems. These models, such as linear recurrent units (LRUs) [19] and S5 [20], use linear time-invariant (LTI) blocks to describe system dynamics combined with nonlinear transformations applied across entire sequences. By applying nonlinearities along the depth of the network, DSSMs can capture complex, layered representations of system dynamics while maintaining the computational efficiency of linear state-space formulations.

DSSMs are particularly appealing for high-dimensional and long-sequence data because they allow for parallelization during both training and inference. Their design also enables them to model systems with complex time-varying interactions without requiring explicit assumptions about local linearity.

Koopman-based methods and DSSMs address complementary aspects of the modeling problem. Koopman-inspired techniques provide a theoretical foundation for understanding and interpreting dynamical systems, whereas DSSMs leverage the representational power of neural networks to model more complex behaviors. The proposed method bridges these approaches by combining the inter-

[1] Additional examples and results are available at https://rodrigodzf.com/physmodjax/JAES.html.

pretability of eigenvalue-based dynamics with the flexibility of neural architectures, offering a robust and lightweight solution for capturing both nonlinear vibrational dynamics.

1.3 Approach

The approach proposed here is centered around an autoencoder architecture, which learns interpretable eigenvalues and approximates the dynamics of vibrational systems in a latent space. This approach balances interpretability, flexibility, and computational efficiency, enabling accurate modeling of both linear and nonlinear dynamics.

To demonstrate the proposed framework, the authors focus on the vibration of a tension-modulated string and a membrane. These systems serve as simple representative examples of vibrational dynamics, exhibiting rich behaviors that are challenging to model. String and membrane vibrations are governed by a PDE that includes terms for stiffness, damping, and tension among other factors. In the linear case, the system can be decomposed into modes where the spatial and temporal dependencies are decoupled and the dynamics reduce to a system of ODEs describing the temporal evolution of the modal amplitudes. In the nonlinear case, the tension includes a term that depends on the global state of the system, introducing coupling between the modes. By projecting the PDE into modal coordinates, the system can be described as a set of ODEs with a nonlinear term, which can then be solved numerically to obtain the time evolution of the modal amplitudes. These solutions allow for the reconstruction of the displacement and velocity over time. A more detailed derivation is provided in APPENDIX A.1.

By applying this data-driven approach to this simpler problem, the authors aim to demonstrate how the proposed method can overcome the limitations of existing approaches, providing an interpretable and flexible framework for modeling vibrational systems. This evaluation highlights the proposed method's ability to capture both the modal structure and the complex nonlinear interactions inherent in these systems.

2 METHOD

2.1 Autoencoder Architecture

The autoencoder consists of three main components: an encoder, a decoder, and a dynamics block. The encoder $\varphi : \mathbb{R}^{2N} \mapsto \mathbb{C}^M$ maps the input data $\boldsymbol{u}_k \in \mathbb{R}^{2N}$, with position and velocity at N grid points, to M complex-valued coordinates $\boldsymbol{x}_k \in \mathbb{C}^M$. The dynamics block advances the sequence in time and applies nonlinear processing. The decoder $\psi : \mathbb{C}^M \mapsto \mathbb{R}^{2N}$ then transforms the evolved sequence back into position and velocity. Together, these components identify a set of basis functions and their inverses that can approximate and, under certain assumptions, reproduce the solutions obtained by other methods (e.g., DMD). The dimensionality of the latent space can be higher or lower than that of the input space, and APPENDIX A.3 shows the impact of this.

When the encoder is implemented as a linear projection, such as a dense layer, and the latent space is constrained to linearity, its columns may correspond to the Koopman modes $\boldsymbol{x}_k$ such that $\boldsymbol{g}(\boldsymbol{u}) = \sum_i \varphi_i(\boldsymbol{u})\boldsymbol{x}_k$. In the case here, observable functions $\boldsymbol{g}(\boldsymbol{u})$ are direct measurements of positions and velocities such that $\boldsymbol{g}(\boldsymbol{u}) = \boldsymbol{u}$. In this simple case, the Koopman eigenfunctions become mode shapes, and the Koopman modes become modal amplitudes and resemble the results obtained with the DMD and analytical modes (if they are obtainable) obtained by the eigendecomposition of the spatial operator.

For systems with higher complexity, where the dynamics and the corresponding eigenfunctions are nonlinear, the encoder can employ a nonlinear transformation, such as a neural network, to approximate the eigenfunctions directly. In this case, the output of the encoder represents the values of the Koopman eigenfunctions evaluated at the state $\boldsymbol{u}$, and the decoder reconstructs the input based on the evolved latent representation.

This encoder and decoder framework provides a flexible mechanism for capturing the dynamics of a system. By using either linear projections or nonlinear transformations, it enables the approximation of Koopman eigenfunctions and their relationship to the observable space, facilitating a comprehensive understanding of both the spatial and temporal aspects of the system. The specific characteristics of the latent space and its evolution will be addressed in subsequent sections.

2.2 Dynamics

The evolution of the system dynamics in the approach here occurs in two stages: an *LTI process* that advances the latent state sequence and a *parallel post-processing network* that refines the trajectory. This design shares similarities with DSSMs [20] and approaches inspired by Koopman theory, where a linear operator governs the global dynamics.

The latent representation $\boldsymbol{x}_k$ is defined over a sequence of length L, where the index k ranges from 0 to L. It evolves through an LTI system defined by

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Lambda}\boldsymbol{x}_k, \quad \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M) \in \mathbb{C}^{M\times M}, \tag{1}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix containing complex eigenvalues λ_i. Because the system is diagonal, the closed-form solution for each mode is $x_k^{(i)} = \lambda_i^k \cdot x_0^{(i)}$, where $x_k^{(i)}$ represents the i-th mode of $\boldsymbol{x}_k$, and λ_i is the corresponding optimized eigenvalue during training. This linear formulation ensures that the latent dynamics are both interpretable and computationally efficient to train. Similar to S5 and LRU, the sequence is computed using the *parallel prefix sum* algorithm [21], which enables efficient and scalable computation. Stability is enforced by constraining the eigenvalues to lie within the unit circle; this is achieved using an exponential nonlinearity or by applying clipping to the radii during optimization (see SEC. 3.1 and APPENDIX A.5).

The resulting sequence $\boldsymbol{x}_{1:L}$, generated by the LTI process, captures the global evolution of the system's dynamics over L time steps. This stage mirrors the behavior of com-

plex resonant filters that are often used in modal synthesis [22].

After the LTI process, the sequence $\boldsymbol{x}_{1:L}$ is refined in parallel using a neural network that introduces local nonlinear modulation. Specifically, the modulation network $\varsigma_{\text{mlp}} : \mathbb{C}^M \mapsto \mathbb{C}^M$ acts on the element-wise square of the real and imaginary components of the latent state. This model is referred to as LTI–multilayer perceptron (MLP). Additionally, the authors propose an adaptation to the modulation network $\varsigma_{\text{siren}} : \mathbb{R} \times \mathbb{R}^{2N} \mapsto \mathbb{C}^M$, which involves using a temporal grid along with a modulation vector. This vector consists of the flattened initial displacement and velocity of the system across all grid points $\boldsymbol{u}_0$, leading to a complex modulation factor $\hat{\boldsymbol{x}}_k \in \mathbb{C}^M$. This model will be called LTI-SIREN because it employs two modulated Siren networks [23]. The final output including the modulation at each time step k is

$$\hat{\boldsymbol{x}}_k = \varsigma(t, \boldsymbol{u}_0) \tag{2}$$

$$\boldsymbol{x}_k := \hat{\boldsymbol{x}}_k \odot \boldsymbol{x}_k, \tag{3}$$

where $t \in [0, 1]$ is the corresponding temporal coordinate.

The inclusion of the temporal grid leverages the prior knowledge of the exponential decay inherent in the LTI process. Combined with the modulation vector, this enables the network to adjust the phase and amplitude of $\boldsymbol{x}_k$ at each time step while preserving the global frequency and decay structure captured by the eigenvalues.

This formulation closely resembles the behavior of DSSMs, where linear evolution governs the global dynamics, and nonlinear transformations refine the output. Similarly, the optimization of complex values in $\boldsymbol{\Lambda}$ are conceptually related to Koopman-based methods, where dynamics are expressed through eigenfunctions of a linear operator.

3 EXPERIMENTS

In this paper, the authors compare six different models to evaluate the proposed method. The evaluation is conducted in two settings: predicting the entire trajectory and predicting only a part of the trajectory. In the latter case, performance in the *autoregressive setting* is also assessed, where the model predicts a single segment and then uses the last time step of the predicted block as the initial condition for the next block. This process continues iteratively, allowing for comparison with the full sequence prediction.

3.1 Architectures

The six models include the autoencoder in an earlier paper [7] LTI-MLP, a modified version LTI-SIREN, an LRU model [19], an S5 model [20], and a baseline DMD model. An overview of the proposed architectures is shown in Fig. 1. The network parameters and computational costs are detailed in APPENDIX A.2.

As outlined in SEC. 2.2, the autoencoder architecture used in this paper builds on earlier work [7], with two key modifications aimed at improving both flexibility and performance. These changes concern the design of the modulation network and the parameterization and constraints applied to the eigenvalues during optimization.

The first change, described in SEC. 2.2, involves the introduction of an improved modulation network. The second change addresses the way eigenvalues are parameterized and constrained. In the original autoencoder, the eigenvalues were optimized in log-space with an additional exponential nonlinearity to ensure that their radius remained within the unit circle. Specifically, each eigenvalue λ was updated using

$$\lambda := \exp(-\exp(\nu_{\log}) + i \exp(\theta_{\log})). \tag{4}$$

In the current work, the eigenvalues are instead optimized using a clipped polar parameterization

$$\lambda := \text{clip}(r) \cdot i \exp(\theta), \tag{5}$$

where the radius r is clipped to remain in the range [0, 1]. The motivation for this change and a discussion of its effects on training convergence is provided in APPENDIX A.5.

3.2 Data Set

The experiments here were carried out on two types of model: a tension-modulated string and a membrane. Unlike the data set used in the previous paper [7], this data set includes both position and velocity data for each time step. The models were tested in two different settings. First, sequences of 16,000 time steps sampled at 16 kHz were fitted, representing 1 s of dynamics. Second, the authors fit 4,000 time steps randomly selected from each trajectory, providing a diverse set of samples from the data set. For the membrane, the authors only train on a randomly selected block of 400 time steps and only evaluate in the autoregressive setting.

For each model, data sets were generated using only Gaussian-shaped initial conditions. In this context, *Gaussian initialization* refers to the initial displacement (velocity) of the string (or membrane) having the shape of a bell (Gaussian) curve. The mean and SD of the Gaussian were randomly selected for each trajectory, creating a variety of smooth bump-like initial conditions with different centers and spreads. This setup approximates real-world excitation profiles, such as plucking (string) or striking (membrane) at a specific point. For the string, maximum displacements range from 1 mm to 1 cm, and the velocity was set to zero for all initial conditions. For the membrane, the maximum velocity range from 5 ms^{-1} to 25 ms^{-1} and the displacement was set to zero for all initial conditions. The data sets were generated by numerically solving the governing equations (see APPENDIX A.1) using an explicit Runge-Kutta solver of order 8(5,3) (DOP853) [24]. The PDE parameters can be found in Tables 3 and 4.

A string data set with 1,000 trajectories and a membrane data set with 100 trajectories was used. Before training, the position and velocity data for both data sets were first normalized to an SD of 1 and subsequently split into training (80%), validation (10%), and test (10%) sets. This setup ensures consistency with previous experiments and allows

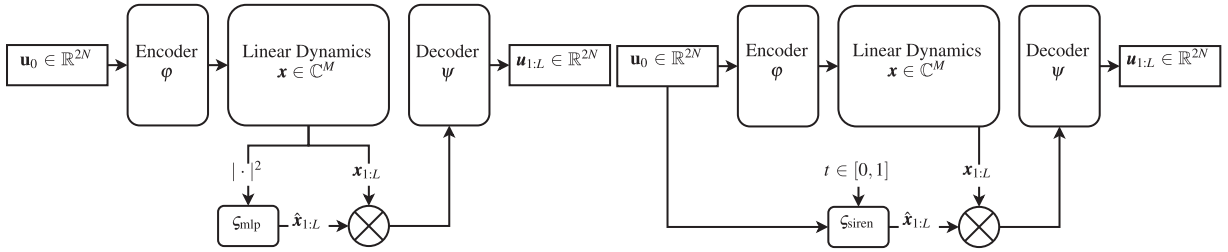


Fig. 1. Architecture of LTI-MLP showing linear dynamics postprocessing, involving an MLP using the square of eigenvalues radii as input and providing output for element-wise multiplication with the state sequence. On the right, an LTI-SIREN is shown, a similar architecture that features a modulated MLP with sinusoidal activations.

for the exploration of the impact of utilizing both position and velocity in the updated network architecture. Focusing on Gaussian initial conditions ensures more controlled and representative initial conditions than noise-like initializations [7].

3.3 Optimization Settings

The models are trained using the AdaBelief optimizer [25] with a learning rate of 0.0001. Training runs are for a maximum of 8,000 epochs, with early stopping based on validation performance to prevent overfitting. A batch size of 25 is used for training with the entire trajectory.

The primary objective function is the reconstruction loss, which measures the squared error between the predicted and actual trajectories. For experiments with global initial conditions (full trajectory), the loss is computed over the entire sequence. In the autoregressive setting, where a block is randomly sampled from the full trajectory, the loss is calculated independently for each block.

To compare with previous work [7], the LTI-MLP model is optimized with and without the linear *consistency loss*. The losses are defined as

$$\mathcal{L}_{cons} = \sum_{k=1}^{L-1} \left\| \varphi(\boldsymbol{u}_k) - \varsigma\left(\boldsymbol{\Lambda}^k \varphi(\boldsymbol{u}_0)\right) \boldsymbol{\Lambda}^k \varphi(\boldsymbol{u}_0) \right\|_2^2, \quad (6)$$

$$\mathcal{L}_{recon} = \sum_{k=1}^{L-1} \left\| \boldsymbol{u}_k - \psi\left[\varsigma\left(\boldsymbol{\Lambda}^k \varphi(\boldsymbol{u}_0)\right) \boldsymbol{\Lambda}^k \varphi(\boldsymbol{u}_0)\right] \right\|_2^2, \quad (7)$$

where φ represents the encoder, ψ is the decoder, and Λ denotes the linear operator and ς the modulation network.

In contrast to the authors' previous work, $\mathcal{L}_{cons}$ is not enforced on the output of the modulation network for the LTI-SIREN model. This is because the encoding, decoding, and dynamics evolution processes are tightly coupled, making the explicit enforcement of consistency less relevant in this updated architecture.

Table 1. Comparison of models for the tension modulated string and membrane in two evaluation settings: full trajectory, which reports the mean squared error (MSE Rel) and mean absolute error (MAE Rel) metrics over the full trajectory (16,000 time steps), and autoregressive setting, which evaluates MSE Rel and MAE Rel for autoregressive predictions (four blocks of 4,000 time steps in the case of the string and four blocks of 400 time steps in the case of the membrane).

	Full Trajectory		Autoregressive Setting	
Model	MSE Rel	MAE Rel	MSE Rel	MAE Rel
DMD	1.4803	1.0226	...	...
LRU(GELU)	0.1889 (0.0788)	0.2741 (0.0474)	0.5479 (0.4401)	0.5941 (0.2482)
LRU(GLU)	0.0212 (0.0030)	0.0828 (0.0086)	0.8917 (0.1155)	0.7886 (0.0498)
S5(GELU)	0.2668 (0.0057)	0.3013 (0.0031)	0.7159 (0.3697)	0.7763 (0.2659)
S5(GLU)	0.0122 (0.0035)	0.0500 (0.0128)	0.2039 (0.0323)	0.3408 (0.0276)
LTI-MLP	**0.0045 (0.0030)**	**0.0423 (0.0062)**	0.0459 (0.0189)	**0.1162 (0.0232)**
LTI-MLP(w. $\mathcal{L}_{cons}$)	0.0050 (0.0005)	0.0508 (0.0026)	**0.0424 (0.0075)**	0.1307 (0.0128)
LTI-SIREN	0.0051 (0.0006)	0.0505 (0.0015)	0.0800 (0.0051)	0.1747 (0.0096)
Models below are trained on the membrane				
LRU(GELU)	...	...	0.2936 (0.0162)	0.4120 (0.0148)
LRU(GLU)	...	...	**0.0075 (0.0009)**	**0.0628 (0.0032)**
S5(GELU)	...	...	0.2492 (0.0569)	0.3544 (0.0347)
S5(GLU)	...	...	0.0178 (0.0030)	0.0976 (0.0061)
LTI-MLP	...	...	0.0417 (0.0111)	0.1505 (0.0188)
LTI-MLP(w. $\mathcal{L}_{cons}$)	...	...	0.0315 (0.0017)	0.1232 (0.0064)
LTI-SIREN	...	...	0.0820 (0.0051)	0.2070 (0.0017)

Note. Results are averaged over three seeds, with SDs in parentheses where applicable. For models such as DMD, which lack stochastic components, a single value is reported. Bolded numbers denote the lowest error.

Table 2. Results for different combinations of initialization parameters: minimum radius (**min r**) and maximum phase (**max** θ) within the unit circle.

		LTI-SIREN		LTI-MLP	
min r	max θ	MSE Rel	MAE Rel	MSE Rel	MAE Rel
0.50	0.01	0.0268 (0.0019)	0.1191 (0.0030)	0.8402 (0.0199)	0.9457 (0.0100)
0.50	3.14	**0.0038 (0.0004)**	**0.0465 (0.0022)**	0.0340 (0.0085)	0.1293 (0.0053)
0.50	6.28	0.0065 (0.0032)	0.0669 (0.0219)	0.0562 (0.0310)	0.2152 (0.0837)
0.80	0.01	0.0268 (0.0050)	0.1190 (0.0182)	0.8255 (0.0273)	0.9398 (0.0105)
0.80	3.14	0.0045 (0.0003)	0.0524 (0.0033)	0.0247 (0.0167)	0.1048 (0.0207)
0.80	6.28	0.0044 (0.0002)	0.0494 (0.0041)	0.0393 (0.0152)	0.1641 (0.0558)
0.99	0.01	0.0318 (0.0016)	0.1272 (0.0044)	0.4821 (0.0549)	0.6944 (0.0646)
0.99	3.14	0.0046 (0.0017)	0.0488 (0.0054)	**0.0044 (0.0017)**	**0.0446 (0.0043)**
0.99	6.28	0.0145 (0.0098)	0.0580 (0.0112)	0.0102 (0.0089)	0.0603 (0.0043)

Note. The table reports the mean absolute error (MAE Rel) and mean squared error (MSE Rel) for each model, averaged over three seeds. The mean values are presented with their SD in parentheses. Bolded numbers denote the lowest error.

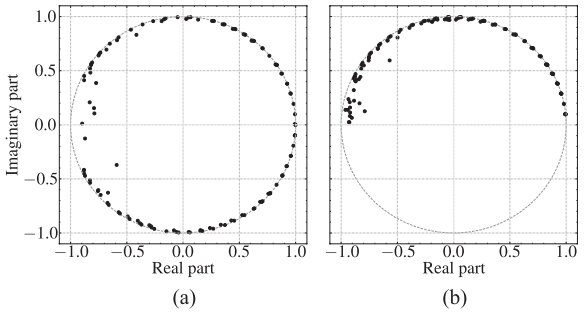


Fig. 2. Optimized eigenvalues for different angle and and magnitude initializations. (a) Eigenvalues are initialized with $r_{min} = 0.99$ and $\theta_{max} = 2\pi$. (b) Eigenvalues are initialized with $r_{min} = 0.5$ and $\theta_{max} = \pi$.

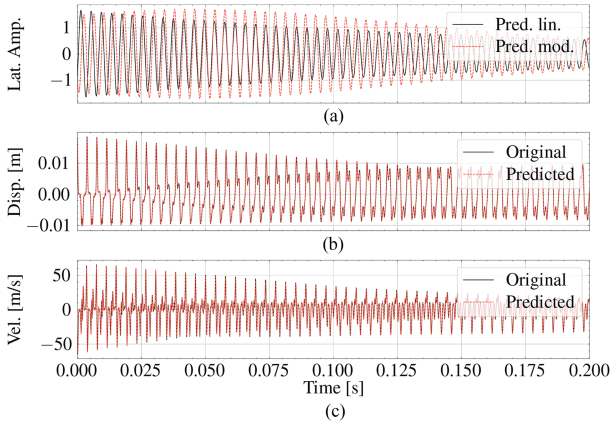


Fig. 3. Modulation of a single eigenvalue path within the network and reconstruction for an unseen initial condition. (a) presents a detailed view of the linear dynamics block output (refer to Fig. 1) in black, and the result after the modulation block in red. (b) shows the displacement prediction at a point (≈28 cm), and (c) depicts the velocity prediction at the same location.

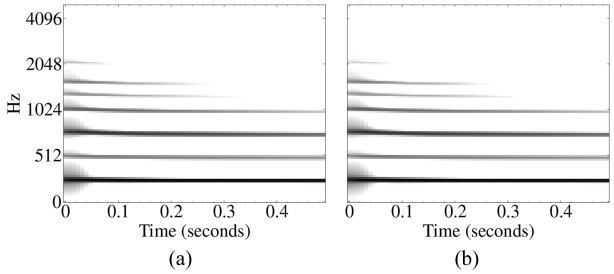


Fig. 4. Mel spectrogram of the displacement at ≈28 cm for the initial condition in Fig. 3, comparing the ground truth (a) and network prediction (b).

3.4 Effect of Eigenvalue Initialization

The initialization of the eigenvalues in the network plays a critical role in determining the dynamics and stability of the models under consideration. The initialization is influenced by two key factors: the phase (angle) and the magnitude (radius) of the eigenvalues.

The results in Table 2 show that the newer model is significantly more robust to variations in initialization. In particular, initializing the phase *in a line*, meaning with a very limited range of values, leads to poor performance. The newer model exhibits strong performance across a wider range of phase initializations, highlighting its robustness.

Additionally, the radius initialization has contrasting effects on the two models. Smaller radius values tend to improve performance because they likely encourage smoother and more stable dynamics in the discrete z-plane. However, these same small radius initializations degrade performance in the previous model, likely due to its parameterization in the Laplace domain, where the logarithmic optimization can amplify sensitivity to radius changes. As shown in Fig. 2, the impact of two distinct initializations is evident. Despite similar reconstruction precision, the eigenvalues settle in different regions of the unit circle.

4 DISCUSSION

Now, the performance of the proposed model is analyzed based on the results presented in Table 1. For the autoregressive setting, the reported values correspond to the reconstruction over the full sequence rather than a single block. Figs. 3 and 4 provide visual comparison of the predicted dynamics for a test initial condition.

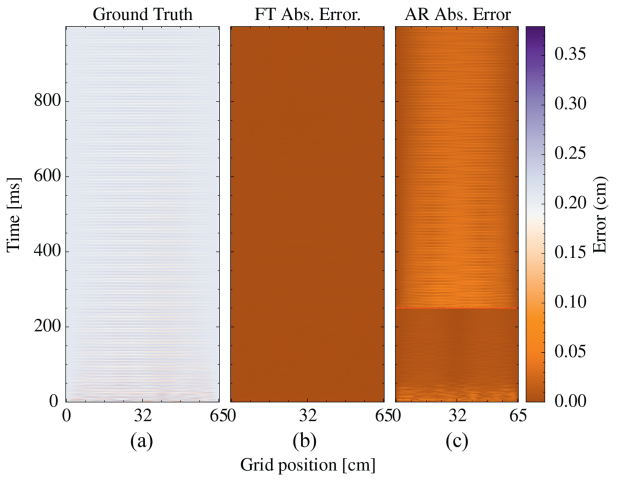


Fig. 5. Absolute error for the prediction along the string over 16,000 time steps (1 s) for an unseen initial condition. (a) Shows the ground truth displacement evolution. (b) displays the prediction error when the model is trained on the full trajectory, and (c) shows the error for the autoregressive setting. The red line indicates the end of the first block in the autoregressive setup.

4.1 Generalization to Initial Conditions

The ability to generalize to unseen initial conditions is a critical feature of the proposed model because it ensures the model has learned the dynamics depending on the variability of the initial condition. In practical settings, initial conditions, such as the initial displacement and velocity of a vibrating string or membrane, vary significantly in terms of their spatial distribution, amplitude, and energy. For instance, in musical instruments like drums or strings, excitation conditions (e.g., a pluck or strike) are rarely identical and can range from highly localized impulses to more spread-out displacements. A robust model must handle this variability without requiring retraining for each new excitation. The performance of networks are evaluated when the center, scale, and spread of the initial displacement are varied.

The results in Table 1 show that the LTI-MLP model generalizes well across varying Gaussian initial conditions when trained on the full trajectory, with the LTI-SIREN model performing comparably. The S5 model in this setting also achieves similar performance.

4.2 Autoregressive Trajectory Reconstruction

The performance of the model on long sequences was evaluated in two settings: reconstructing full trajectories when trained on complete sequences of 16,000 time steps, and predicting entire sequences through an autoregressive approach, where the model is trained on blocks and uses the final predicted sample of each block as the initial condition for the subsequent block. This strategy enables the model to predict over long time horizons effectively, as shown in Figs. 5 and 6.

In this setting, the LRU and S5 models outperform the others. Results are reported using both gated linear unit (GLU) and Gaussian error linear unit (GELU) nonlinearities for these models because performance varies considerably depending on the nonlinearity used (see APPENDIX A.4). If only the GELU case is considered, the LTI models yield better results.

Although errors accumulate in the autoregressive setting, they primarily affect amplitude and phase rather than frequency content. This indicates that the model effectively retains the core oscillatory characteristics of the dynamics, even over extended time horizons. The preservation of frequency content ensures that the perceptual quality of the synthesized sound remains consistent, which is critical for applications in sound synthesis. Readers are encouraged to explore the provided audio examples, where the frequency remains close to the ground truth despite the accumulation of small prediction errors over time.

Compared with methods such as online DMD [16] and the recent neural network-based approach [26], which approximate nonlinear or time-varying systems as locally linear within fixed blocks, the proposed model avoids assumptions about local linearity. In block-based methods, the choice of block size is critical: overly large blocks fail to capture fast nonlinearities, especially for localized initial conditions with sharp energy concentrations. In contrast, this approach decouples global and local dynamics, optimizing global dynamics via an LTI process across the entire trajectory and introducing local adjustments at each time step through a neural network. This separation enhances robustness to fast-changing dynamics and improves handling of localized excitations.

Regarding computational time complexity, the method proposed in [26] achieves efficiency by representing eigenvalues linearly and ensuring phase continuity across blocks using a cumulative sum. This operation, efficiently implemented with parallel algorithms such as the prefix sum [21], shares the same theoretical time complexity as the approach proposed here. However, their method partially relies on the linear solution of the system during training, which leverages prior knowledge of the governing PDE. This reliance on explicit physics-based formulations contrasts with the authors' fully data-driven approach, which operates without requiring detailed prior information, offering greater flexibility and broader applicability.

4.3 Interpretability and Eigenvalue Modulation

A significant advantage of the proposed method lies in the interpretability of the learned eigenvalues that govern the dynamics. The eigenvalues provide a direct connection between the latent representation and physical properties, such as oscillation frequencies and decay rates. This interpretability enables meaningful and precise manipulations of the system behavior. For example, adjusting the eigenvalues allows for control over the vibrational characteristics such as pitch shifting or modifying decay rates. This is conceptually similar to modifying the dispersion relationship of the system, effectively changing its vibrational properties without requiring explicit material parameters or retraining. Such capabilities make the model a powerful tool for sound synthesis and sound design.

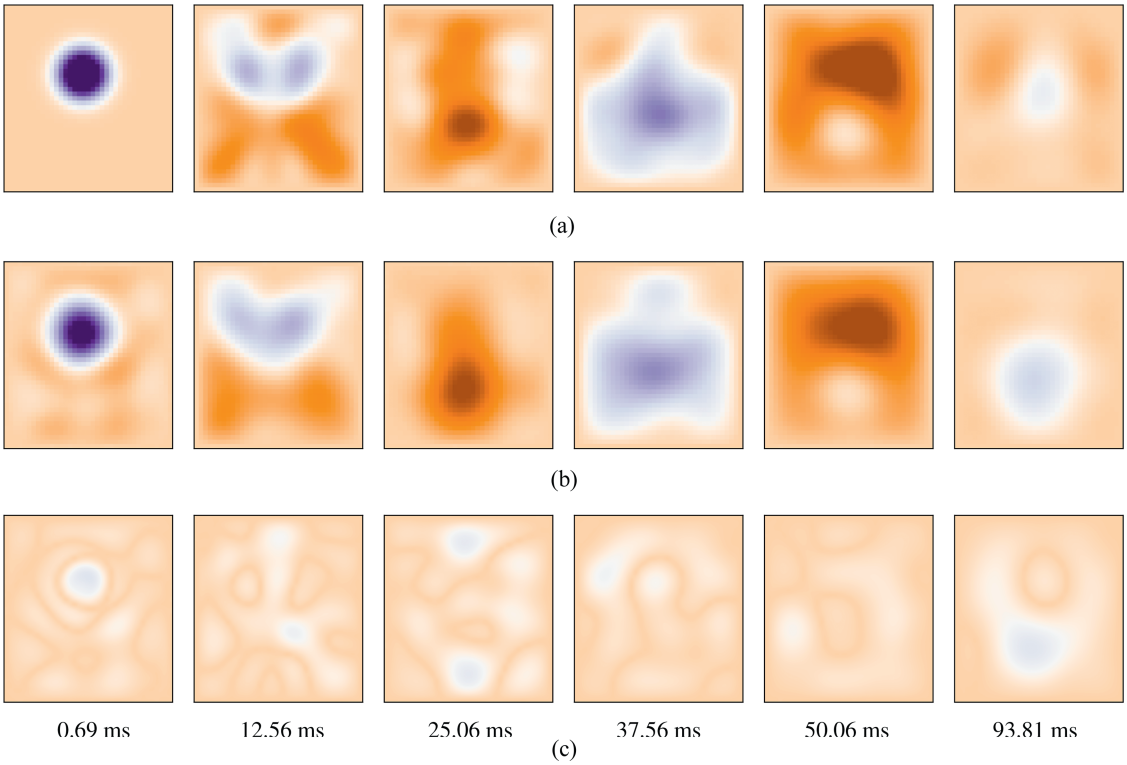


Fig. 6. Displacement evolution on a membrane for 1,600 time steps (100 ms) for an unseen initial condition. (a) Shows the sequence of ground truth snapshots. (b) Shows the predicted snapshots, and (c) shows their absolute error. The model is trained on a single block of 400 time steps (25 ms), and the depicted snapshots in time are generated in an autoregressive setting, where each subsequent block is predicted using the last sample of the preceding block.

To some extent, the network has also learned to separate global dynamics from modulation, which depends on the initial conditions. This separation is not perfectly clear in all cases because certain combinations of eigenvalues and modulation inputs can produce similar outcomes. However, as illustrated in Fig. 3, when analyzing the modulation of a single eigenvalue (manually selected near the apparent fundamental frequency), the output of the network shows a clear modulation of frequency and amplitude. This behavior suggests the model has learned to differentiate between global and local dynamics in a meaningful way, offering insight into the underlying structure of the system.

Importantly, these modifications are applied in a controlled manner via the modulation network, ensuring that the globally learned frequencies and decay characteristics remain consistent. This balance between flexibility and stability underscores the robustness of the proposed approach.

5 CONCLUSION

In this paper, the authors presented a data-driven neural framework for simulating the dynamics of strings and membranes using a two-stage approach. First, an LTI process was employed, where the dynamics evolve through a diagonal matrix whose eigenvalues are optimized directly. This stage captures the global frequency and decay structure efficiently using a cumulative product, implemented with the parallel prefix sum algorithm. Second, a parallel postprocessing neural network was introduced, specifically either an MLP or a modulated SIREN network, to refine the global dynamics by introducing local nonlinear modulation. The combination of these stages enables this method to model complex oscillatory systems effectively while retaining computational efficiency. Experimental results demonstrated the framework's ability to generalize to unseen initial conditions and handle vibrational dynamics in both 1D and 2D settings.

The trained model can be deployed for inference in applications where low-latency simulation is critical. For instance, it could serve as a computationally efficient, real-time synthesizer for virtual strings and membranes with complex nonlinearities. The interpretability of the model also offers opportunities for interactive sound design. For example, the learned eigenvalues can be manipulated to modify the perceived pitch and decay.

Despite the strengths of the proposed approach, there are several limitations that must be addressed. First, the presence of noise was observed in the predicted outputs, primarily because the current model training relies exclusively on a time-domain loss. Although frequency-domain losses can help mitigate this issue for 1D systems, extending such losses to 2D becomes computationally expensive due to the high-dimensional Fourier transforms involved. Efficiently incorporating frequency-domain constraints in the 2D setting is a nontrivial task that has yet to be fully addressed.

Second, the experiments in this study were conducted on clean, synthetic data with well-defined initial condi-

tions. However, real-world data sets often contain noise, which can affect the model's robustness. Incorporating noise into the training process would allow for better evaluation of the performance of the model under realistic conditions and improve its ability to generalize to less ideal data.

Third, as the spatial dimension of the input grows, particularly in the 2D or 3D cases, the joint optimization of the encoder, decoder, and dynamics becomes increasingly challenging. This tight coupling can hinder scalability and lead to suboptimal training. To address this, future work will explore the decoupling of the encoder-decoder from the dynamics evolution. Specifically, the aim is to adopt a pretraining strategy in which the autoencoder is first trained independently to learn spatial representations, followed by a separate stage to optimize the temporal dynamics. This approach has been successfully applied in other contexts [27–29] and may improve the scalability and performance of higher-dimensional systems.

6 ACKNOWLEDGMENT

This work has been funded by UK Research and Innovation (UKRI) and the Engineering and Physical Sciences Research Council (EPSRC) as part of the UKRI Centre for Doctoral Training (CDT) in Artificial Intelligence and Music under Grant EP/S022694/1. This research utilized Queen Mary's Apocrita High-Performance Computing (HPC) facility, supported by Queen Mary University of London (QMUL) Research-IT. http://doi.org/10.5281/zenodo.438045.

7 REFERENCES

[1] F. Avanzini, R. Marogna, and B. Bank, "Efficient Synthesis of Tension Modulation in Strings and Membranes Based on Energy Estimation," *J. Acoust. Soc. Am.*, vol. 131, no. 1, pp. 897–906 (2012 Jan.). https://doi.org/10.1121/1.3651097.

[2] R. Rabenstein and L. Trautmann, "Multidimensional Transfer Function Models," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 49, no. 6, pp. 852–861 (2002 Jun.). https://doi.org/10.1109/TCSI.2002.1010040.

[3] N. Kovachki, Z. Li, B. Liu, et al., "Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs," *J. Mach. Learn. Res.*, vol. 24, no. 89, pp. 1–97 (2022 Dec.).

[4] J. Parker, S. Schlecht, M. Schäfer, and R. Rabenstein, "Physical Modeling Using Recurrent Neural Networks With Fast Convolutional Layers," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pp. 138–145 (Vienna, Austria) (2022 Sep.).

[5] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep Learning for Universal Linear Embeddings of Nonlinear Dynamics," *Nat. Commun.*, vol. 9, p. 4950 (2018 Nov.). https://doi.org/10.1038/s41467-018-07210-0.

[6] V. Huhtala, L. Juvela, and S. J. Schlecht, "KLANN: Linearising Long-Term Dynamics in Nonlinear Audio Effects Using Koopman Networks," *IEEE Signal Process. Lett.*, vol. 31, pp. 1169–1173 (2024 Apr.). https://doi.org/10.1109/LSP.2024.3389465.

[7] R. Diaz, C. De La Vega Martin, and M. Sandler, "Towards Efficient Modelling of String Dynamics: A Comparison of State Space and Koopman Based Deep Learning Methods," in *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx)* (Guildford, UK) (2024 Sep.).

[8] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *J. Comput. Phys.*, vol. 378, pp. 686–707 (2019 Feb.). https://doi.org/10.1016/j.jcp.2018.10.045.

[9] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural Ordinary Differential Equations," in *Proceedings of the Neural Information Processing Systems (NeurIPS)*, vol. 31 (Montreal, Canada) (2018 Dec.).

[10] W. Steedly, C.-H. Ying, and R. Moses, "Statistical Analysis of SVD-based Prony Techniques," in *Proceedings of the Conference Record of the Twenty-Fifth Asilomar Conference on Signals, Systems & Computers*, vol. 1, pp. 232–236 (Pacific Grove, CA) (1991 Nov.). https://doi.org/10.1109/ACSSC.1991.186447.

[11] Y. Susuki and I. Mezić, "A Prony Approximation of Koopman Mode Decomposition," in *Proceedings of the 54th IEEE Conference on Decision and Control (CDC)*, pp. 7022–7027 (Osaka, Japan) (2015 Dec.). https://doi.org/10.1109/CDC.2015.7403326.

[12] I. Mezić, "Spectral Properties of Dynamical Systems, Model Reduction and Decompositions," *Nonlinear Dyn.*, vol. 41, no. 1, pp. 309–325 (2005 Aug.). https://doi.org/10.1007/s11071-005-2824-x.

[13] P. J. Schmid, "Dynamic Mode Decomposition and Its Variants," *Annu. Rev. Fluid Mech.*, vol. 54, no. 1, pp. 225–254 (2022 Jan.). https://doi.org/10.1146/annurev-fluid-030121-015835.

[14] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis, "A Kernel-Based Method for Data-Driven Koopman Spectral Analysis," *J. Comput. Dyn.*, vol. 2, no. 2, pp. 247–265 (2015 Jun.). https://doi.org/10.3934/jcd.2015005.

[15] H. Arbabi and I. Mezić, "Ergodic Theory, Dynamic Mode Decomposition, and Computation of Spectral Properties of the Koopman Operator," *SIAM J. Appl. Dyn. Syst.*, vol. 16, no. 4, pp. 2096–2126 (2017 Jan.). https://doi.org/10.1137/17M1125236.

[16] H. Zhang, C. W. Rowley, E. A. Deem, and L. N. Cattafesta, "Online Dynamic Mode Decomposition for Time-Varying Systems," *SIAM J. Appl. Dyn. Syst.*, vol. 18, no. 3, pp. 1586–1609 (2019 Jan.). https://doi.org/10.1137/18M1192329.

[17] R. Pascanu, T. Mikolov, and Y. Bengio, "On the Difficulty of Training Recurrent Neural Networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning (ICML)*, vol. 28, pp. 1310–1318 (Atlanta, GA) (2013 Jun.).

[18] I. Sutskever, *Training Recurrent Neural Networks*, Ph.D. thesis, University of Toronto, Toronto, Canada (2013

Aug.).

[19] A. Orvieto, S. L. Smith, A. Gu, et al., "Resurrecting Recurrent Neural Networks for Long Sequences," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, vol. 202, pp. 26670–26698 (Honolulu, HI) (2023 Jul.).

[20] J. T. H. Smith, A. Warrington, and S. Linderman, "Simplified State Space Layers for Sequence Modeling," presented at the *The Eleventh International Conference on Learning Representations* (Kigali, Rwanda) (2023 May).

[21] G. E. Blelloch, "Prefix Sums and Their Applications," Tech. Rep. CMU-CS-90-1903 (1990 Nov.). https://www.cs.cmu.edu/∼guyb/papers/Ble93.pdf.

[22] M. Mathews and J. O. Smith, "Methods for Synthesizing Very High Q Parametrically Well Behaved Two Pole Filters," in *Proceedings of the Stockholm Musical Acoustics Conference (SMAC)* (Stockholm, Sweden) (2003 Aug.).

[23] I. Mehta, M. Gharbi, C. Barnes, E. Shechtman, R. Ramamoorthi, and M. Chandraker, "Modulated Periodic Activations for Generalizable Local Functional Representations," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14194–14203 (Montreal, Canada) (2021 Oct.). https://doi.org/10.1109/ICCV48922.2021.01395.

[24] E. Hairer, G. Wanner, and S. P. Nørsett, "Runge-Kutta and Extrapolation Methods," in *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer Series in Computational Mathematics, pp. 129–353 (Springer, Berlin, Germany, 1993). https://doi.org/10.1007/978-3-540-78862-1_2.

[25] J. Zhuang, T. Tang, Y. Ding, et al., "AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 33, pp. 18795–18806 (Virtual) (2020 Dec.).

[26] J. W. Lee, J. Park, M. J. Choi, and K. Lee, "Differentiable Modal Synthesis for Physical Modeling of Planar String Sound and Motion Simulation," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 37, pp. 1058–1081 (Vancouver, Canada) (2024 Dec.).

[27] L. Serrano, L. Le Boudec, A. K. Koupaï, et al., "Operator Learning With Neural Fields: Tackling PDEs on General Geometries," in *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS)*, vol. 36, pp. 70581–70611 (New Orleans, LA) (2023 Dec.).

[28] Y. Yin, M. Kirchmeyer, J.-Y. Franceschi, A. Rakotomamonjy, and P. Gallinari, "Continuous PDE Dynamics Forecasting With Implicit Neural Representations," in *Proceedings of the Eleventh International Conference on Learning Representations* (Kigali, Rwanda) (2023 May).

[29] P. Y. Chen, J. Xiang, D. H. Cho, et al., "CROM: Continuous Reduced-Order Modeling of PDEs Using Implicit Neural Representations," in *Proceedings of the Eleventh International Conference on Learning Representations* (Kigali, Rwanda) (2023 May).

[30] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, vol. 30, pp. 972–981 (Long Beach, CA) (2017 Dec.).

APPENDIX

A.1 VIBRATION OF STRINGS AND MEMBRANES

Let $u(\boldsymbol{x}, t)$ be the displacement of a point $\boldsymbol{x} \in S \subset \mathbb{R}^d$ at time t. This function represents the transverse vibration of a string ($d = 1$) or a membrane ($d = 2$), and its dynamics are described by the partial differential equation (PDE):

$$D\nabla^4 u + \rho_d \frac{\partial^2 u}{\partial t^2} - T(u)\nabla^2 u + d_1 \frac{\partial u}{\partial t} + d_3 \frac{\partial \nabla^2 u}{\partial t} = f_{\text{ext}} \tag{8}$$

where ∇^2 represents the Laplace operator, and ∇^4 denotes the biharmonic operator. The term ρ_d indicates the material density in (kg/m^d), D refers to the bending stiffness ($N \times \text{m}^{3-d}$), and T is the tension given in (N/m^{d-1}). The parameter d_1 corresponds to the frequency-dependent damping (kg/m^{d}s), and d_3 is the frequency-independent damping (kg $\times$ m^d/s). The external force on the system is denoted by $f^{(ext)}$ and is measured in (N/m^d).

The tension can be split into a linear part and a nonlinear correction $T(u) = T_0 + T_{NL}(u)$. Assuming that the tension remains homogeneous, and therefore it only depends on the global state

$$T_{NL}(u) = C_{NL} \frac{S(u) - S_0}{S_0} \simeq \frac{1}{2} \frac{C_{NL}}{S_0} \int_S \|\nabla u\|^2 d\boldsymbol{x}, \tag{9}$$

where $S(u)$ is the extension (length of the string or area of the membrane) and C_{NL} is a coefficient that depends on the geometry of the problem [1].

Table 3. Physical parameters of a nylon guitar string tuned to $\boldsymbol{B}_3$.

Parameter	Value		Unit
ρ_1	5.91	$\times 10^{-4}$	kgm^{-1}
D	1.29		Nm^2
T_0	60.97		N
d_1	8.00	$\times 10^{-5}$	kgm^{-1}s^{-1}
d_3	1.40	$\times 10^{-5}$	kgms^{-1}
ℓ	0.65		m

Table 4. Physical parameters of a rectangular mylar membrane tuned to a fundamental frequency of approximately 100 Hz.

Parameter	Value		Unit
ρ_2	0.26		kg/ m^2
D	2.20	$\times 10^{-3}$	Nm
T_0	800.00		Nm^{-1}
d_1	0.50		kg/ m^2 s^{-1}
d_3	0.01		kg m^2 s^{-1}
ℓ_x	0.40		m
ℓ_y	0.36		m

In the linear case, $T_{NL} = 0$, the system can be expressed as a sum of modes, where the spatial and temporal dependencies have been decoupled,

$$u(\boldsymbol{x}, t) = \sum_{\eta} \frac{\bar{u}_\eta(t) K_\eta(\boldsymbol{x})}{\|K_\eta(\boldsymbol{x})\|_2^2}, \eta \in \mathbb{N}^d, \tag{10}$$

and the modal shapes obey

$$\nabla^2 K_\eta(\boldsymbol{x}) = -\lambda_\eta K_\eta(\boldsymbol{x}), \tag{11}$$

then the PDE can be expressed as a system of ODEs

$$\rho_d \ddot{\bar{u}}_\eta + \left(d_3 \lambda_\eta + d_1\right) \dot{\bar{u}}_\eta + \left(\lambda_\eta \left(\lambda_\eta D + T_0\right)\right) \bar{u}_\eta = \bar{f}_\eta^{(tm)}. \tag{12}$$

The nonlinear part of the tension can be considered as a forcing term and, assuming that the modal shapes remain the same, project into modal coordinates

$$f_\eta^{(tm)}(u, \bar{u}) = -\lambda_\eta T_{NL}(u) \bar{u}_\eta(t) \tag{13}$$

$$= -\lambda_\eta \frac{1}{2} \frac{C_{NL}}{S_0} \left[\sum_{\tilde{\eta}} \frac{\lambda_{\tilde{\eta}} \bar{u}_{\tilde{\eta}}^2(t)}{\|K_{\tilde{\eta}}\|_2^2} \right] \bar{u}_\eta(t). \tag{14}$$

Given a set of initial conditions u_0, $\dot{u}_0$, these ODEs can then be solved numerically using various methods, such as the Runge-Kutta method (RK4), to obtain the time evolution of the amplitudes $u_\eta(t)$. The resulting solution can be used to reconstruct the displacement $u(\boldsymbol{x}, t)$ and velocity $\dot{u}(\boldsymbol{x}, t)$.

A.2 NETWORK PARAMETERS

LTI-MLP and LTI-SIREN use MLP-based encoders and decoders, each consisting of three layers. In the 1D case, these layers have 128 units, whereas in the 2D case, they are expanded to 1,024 units. The encoder outputs 256 dimensions in the 1D case (representing 128 complex latent amplitudes as real components) and 2,048 dimensions in the 2D case (representing 1,024 complex latent amplitudes). The decoder produces 202 dimensions for 1D (corresponding to 101 positions and velocities) and $41 \times 37 \times 2$ dimensions for 2D (representing width, height, position, and velocity).

In the original LTI-MLP implementation, the modulation network is a shallow MLP with scaled exponential linear unit activations [30], which uses $|\boldsymbol{x}_k|^2$ as input. In contrast, LTI-SIREN employs sinusoidal activations. This updated version uses two separate modulation networks: one for predicting magnitude and another for phase.

The LRU model consists of five layers, each with 128 complex-valued units in 1D (1,024 in 2D), using GELU or GLU activations. The S5 model follows a similar structure, comprising four layers of equivalent dimensionality and nonlinear blocks. However, a distinction is that S5 employs conjugate-symmetric eigenvalues, which reduce the number of independently optimized complex parameters by half, thereby lowering the overall model size.

Although the total parameter count remains moderate, the space complexity scales linearly with both sequence length and hidden dimensionality. Additionally, the use of nonlinear blocks, particularly in the LRU(GLU) and S5(GLU) variants, introduces considerable computational overhead, resulting in significantly longer training times.

Table 5. Comparison of model architectures by approximate FLOPs per single forward pass, number of parameters, and size in memory of the models.

Model	FLOPs	Parameter	Memory (MB)
LTI-MLP	5,294,022	266,826	1.1
LTI-SIREN	3,092,059	472,266	1.9
LRU(GELU)	2,556,135	628,100	2.5
LRU(GLU)	2,653,090	1,038,160	4.2
S5(GELU)	1,877,967	314,656	1.3
S5(GLU)	1,954,722	726,736	2.9
2D Models			
LTI-MLP	183,351,296	18,769,920	75.1
LTI-SIREN	60,854,424	40,565,760	162.3
LRU(GELU)	152,597,760	62,191,122	248.8
LRU(GLU)	154,114,784	135,856,642	543.4
S5(GELU)	79,739,976	31,115,282	124.5
S5(GLU)	81,317,656	104,780,802	419.1

Table 5 summarizes the approximate floating point operations, parameters, and memory usage for each model, illustrating the trade-offs between computational efficiency and model complexity across 1D and 2D settings.

A.3 ABLATION ON THE SIZE OF THE LATENT COORDINATES

The effect of the number of latent coordinates M on model performance is investigated. As shown in Table 6, increasing M improves reconstruction accuracy as measured by both mean squared error (MSE Rel) and mean absolute error (MAE Rel). This is expected because a higher latent dimensionality might allow the model to represent a richer set of modes and capture more of the dynamics.

However, this increase in accuracy comes at greater computational cost. A larger M leads to a proportional increase in the number of parameters, memory usage, and training time. In particular, the size of the linear operator $\boldsymbol{\Lambda}$ and the downstream modulation network scale with M, affecting both the LTI dynamics and the nonlinear processing stages.

A.4 ACTIVATIONS AND NONLINEAR BLOCKS IN THE STATE-SPACE MODELS

Table 6. Effect of latent size M on model performance.

M	MSE Rel	MAE Rel
64	0.0330 (0.0041)	0.1563 (0.0171)
128	0.0106 (0.0095)	0.0606 (0.0264)
256	**0.0045 (0.0031)**	**0.0478 (0.0090)**

Note. Metrics are averaged over three runs with SD in parentheses. Bolded numbers denote the lowest error.

Table 7 compares the impact of different nonlinear components following each linear block within the SSM architectures. Specifically, three types of nonlinear transformations are evaluated: the GELU, the GLU, and a three-layer MLP. The GELU is defined as $\mathrm{GELU}(x) = x \cdot \Phi(x)$, where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. The GLU applies a gating mechanism on the GELU-activated input: $\mathrm{GLU}(x) = (W_1 \cdot \mathrm{GELU}(x) + b_1) \cdot \sigma(W_2 \cdot \mathrm{GELU}(x) + b_2)$, where σ is the sigmoid function. The MLP variant uses a three-layer architecture with GELU activations typical in transformers. As shown in the table, the choice of nonlinearity significantly affects model performance, with the GLU yielding the best results for the LRU model and the MLP performing slightly better for the S5 variant.

Table 7. MAE Rel for the LRU and S5 models with different activation and nonlinear blocks.

Model	GELU	GLU	MLP
LRU	0.3852 (0.0106)	**0.0972 (0.0108)**	0.1599 (0.0294)
S5	0.3806 (0.0060)	0.0508 (0.0061)	**0.0398 (0.0092)**

Note. Bolded numbers denote the lowest error.

A.5 EXPONENTIAL PARAMETERIZATION AND CONSTRAINTS

It was found that optimizing eigenvalues using an exponential parameterization in log space can be problematic, particularly when targeting resonant poles with minimal damping. Although this parameterization ensures the radius r remains positive and less than one, the logarithmic mapping stretches values close to the unit circle. This results in a flattened gradient landscape, making it difficult for the optimizer to reach poles with radii near one.

To address this, a clipping operation[2] is applied during training. This modification retains the advantages of the exponential parameterization, namely axis-aligned optimization of magnitude and angle, whereas reducing the distortion caused by the log transformation near the unit circle.

Fig. 7 shows loss histories for a toy experiment adapted from [19, SEC. E.2]. The task is to minimize the loss

$$L(\hat{\lambda}) = \frac{1}{2}\left|\hat{\lambda}^k - \left(\lambda^*\right)^k\right|^2,$$

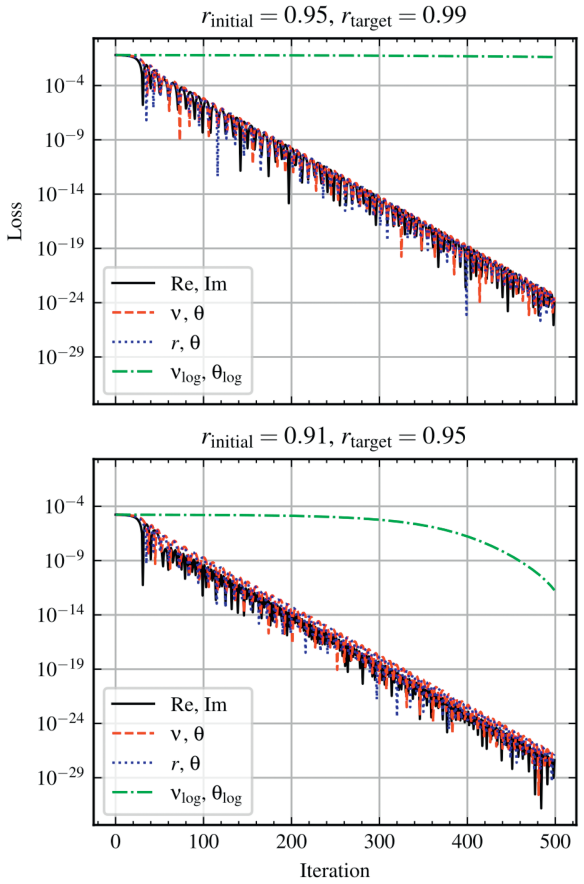


Fig. 7. Loss histories for different parameterizations of the eigenvalue λ. The black curve corresponds to direct optimization of the real and imaginary parts $\lambda = \lambda_{\text{real}} + i\lambda_{\text{imag}}$. The red curve shows the log radius and angle parameterization $\lambda = \exp(-\nu + i\theta)$. The blue curve represents the direct optimization of radius and angle $\lambda = r\exp(i\theta)$. The solid green curve corresponds to $\lambda = \exp(-\exp(\nu) + i\theta)$, and the dashed green curve shows the result for $\lambda = \exp(-\exp(\nu) + i\exp(\theta))$. The Adam optimizer with a learning rate of 10^{-3} was used for all runs.

where λ^* is the target eigenvalue, $\hat{\lambda}$ is the optimized eigenvalue, and k is the number of time steps. Here, the authors initialize the radius away from the target value while setting the initial angle to the target angle. Most parameterization converges to similar final loss values, with the exception of the exponential parameterization with the exponential nonlinearity. In particular, convergence is slower when the initial radius is close to the unit circle, highlighting the sensitivity of this parameterization in low-damping regimes.

[2] A JAX implementation of the gradient-corrected clipping based on https://github.com/ben-hayes/magic-clamp was used.

THE AUTHORS



Rodrigo Diaz



Carlos De La Vega Martin



Mark Sandler

Rodrigo Diaz is a Ph.D. candidate in artificial intelligence and music at Queen Mary University of London under the supervision of Prof. Mark Sandler and Dr. Charalampos Saitis. His research focuses on real-time audio synthesis using neural networks and physics-based modeling. His work has been presented at conferences across the audio and computer vision communities, including Computer Vision and Pattern Recognition (CVPR), International Conference on Acoustics, Speech, and Signal Processing (ICASSP), International Conference on 3D Imaging (IC3D), International Conference on Digital Audio Effects (DAFx), and Audio Engineering Society (AES). Before his Ph.D., he worked as a researcher at the Fraunhofer HHI Institute in Berlin exploring volumetric reconstruction from images using neural networks.

•

Carlos De La Vega Martin is a Ph.D. candidate in artificial intelligence and music at Queen Mary University of London under the supervision of Prof. Mark Sandler and Dr. Johan Pauwels. He received his B.Sc. degree from Durham University in 2016 and his M.Sc. degree from the University of Edinburgh in 2017. His research focuses on neural solvers for spatiotemporal PDEs, with an special interest in both acoustics and nonlinear dynamics. His work has been presented at conferences such as the International Conference on Machine Learning (ICML), Forum Acusticum, and the International Conference on Digital Audio Effects (DAFx). Before starting his Ph.D., he worked in photonic and electronic integrated circuit (IC) design.

•

Mark Sandler, FREng (Fellow of the Royal Academy of Engineering), is Director of the Centre for Digital Music at Queen Mary University of London, where he is a Professor of Signal Processing. He has been active in digital audio and music research since the late 1970s, when he started his Ph.D. in digital audio power amplifiers. Since then, he has published well over 500 papers across many topics in digital audio and computer music. Sound synthesis has been a particular interest since the 1980s, when he worked with the pioneering British drum synthesizer company, Simmons Electronics. He has been active in artificial intelligence (AI) for music and audio since about 2014. He has published more than 500 papers and supervised more than 50 Ph.D. and 20 postdoctoral research assistants (RAs). He is also a Fellow of the AES, IEEE, and IET.