TOWARDS EFFICIENT MODELLING OF STRING DYNAMICS: A COMPARISON OF STATE SPACE AND KOOPMAN BASED DEEP LEARNING METHODS

Rodrigo Diaz *Carlos De La Vega Martin* *Mark Sandler* *

Centre for Digital Music
Queen Mary University of London, UK
r.diazfernandez c.delavegamartin mark.sandler@qmul.ac.uk

ABSTRACT

This paper presents an examination of State Space Models (SSM) and Koopman-based deep learning methods for modelling the dynamics of both linear and non-linear stiff strings. Through experiments with datasets generated under different initial conditions and sample rates, we assess the capacity of these models to accurately model the complex behaviours observed in string dynamics. Our findings indicate that our proposed Koopman-based model performs as well as or better than other existing approaches in non-linear cases for long-sequence modelling.

We inform the design of these architectures with the structure of the problems at hand. Although challenges remain in extending model predictions beyond the training horizon (i.e., extrapolation), the focus of our investigation lies in the models' ability to generalise across different initial conditions within the training time interval. This research contributes insights into the physical modelling of dynamical systems (in particular those addressing musical acoustics) by offering a comparative overview of these and previous methods and introducing innovative strategies for model improvement. Our results highlight the efficacy of these models in simulating non-linear dynamics and emphasise their wide-ranging applicability in accurately modelling dynamical systems over extended sequences.

1. INTRODUCTION

The analysis and modelling of acoustic dynamical systems have been critical areas of scientific research and practical applications for many years. Numerous numerical methods, including traditional approaches such as finite differences and finite elements, have been developed to simulate a broad spectrum of acoustic phenomena within dynamic systems. These methods provide robust frameworks for simulation but often encounter complexities and limitations, particularly with nonlinear systems or those requiring real-time interaction [1].

Learning-based methods are becoming increasingly prevalent due to their potential to model complex dynamics efficiently. Such methods have been proposed to address some of these obstacles in physical modelling. In particular, neural operator-based architectures [2] and and their recurrent variants [3] have been explored for this purpose. However, in such cases, training sequences are limited to dozens or hundreds of steps, with performance degrading rapidly when extrapolating beyond the training domain [4, 5]. These methods face difficulties in training longer sequences, primarily due to gradient instability resulting from vanishing or exploding gradients [6]. Additionally, using nonlinearities in the recursion considerably slows down the training process [7].

In contrast, optimising linear layers is computationally more efficient, as the recursion can be parallelised more effectively [8]. The emergence of deep learning methods that incorporate linear dynamics offers an alternative perspective for modelling problems of this kind. In particular, methods grounded in Koopman theory have shown potential in accurately capturing the nuances of nonlinear dynamic systems [9]. Similarly, deep SSMs have achieved success in efficiently processing very long sequences across various tasks [10].

This preliminary work applies these methods to simulate the dynamics of one-dimensional strings and provides a comprehensive comparison of several related neural architectures to model such dynamics. We introduce a modified Koopman-based model that can capture the nonlinear dynamics of strings with greater accuracy and fewer parameters than previous methods [3]. We also investigate the connections between analytical solutions to these challenges and the architectural decisions behind these deep learning models, providing insights into their effectiveness.

2. BACKGROUND

2.1. The stiff string

The motion of a stiff string defined over the interval $x \in [0, \ell]$ and with no external forcing can be expressed as

$$\rho A \frac{\partial^2 y}{\partial t^2} + EI \frac{\partial^4 y}{\partial x^4} - T(y) \frac{\partial^2 y}{\partial x^2} + d_1 \frac{\partial y}{\partial t} - d_3 \frac{\partial^3 y}{\partial x^2 \partial t} = 0, \quad (1)$$

Where $y(x,t)$ is the transverse displacement of the string, ρ is the mass density, A is the cross-sectional area, E is the Young's modulus, I is the second moment of area, $T(y)$ is the displacement dependent tension, and d_1 and d_3 are the damping coefficients. This PDE, together with a valid set of boundary conditions and initial conditions, defines the Initial-Boundary Value Problem (IBVP) that we are interested in solving. (see Table 1 for the parameter values used).

2.1.1. Linear case

For small displacements, we can take $T(y) = T_0$ to be a constant. Assuming simply-supported boundary conditions, $y(0,t) = y(\ell,t) = 0$ and $y''(0,t) = y''(\ell,t) = 0$, and using separation of variables, we can arrive to a set of spatial basis functions

* This work was funded by UKRI and EPSRC as part of the "UKRI CDT in Artificial Intelligence and Music", under grant EP/S022694/1.

$K_\mu(x) = \sin\left(\frac{\mu\pi x}{\ell}\right)$ for $\mu = 1, 2, \ldots$. These basis functions, which are real in this specific case, diagonalise the system [11, 12]:

$$y(x,t) = \sum_{\mu=1}^{\infty} \frac{\bar{y}_\mu(t) K_\mu(x)}{\|K_\mu(x)\|_2^2}. \tag{2}$$

The spatial modal shapes $K_\mu(x)$ constitute the columns of our transformation kernel $\boldsymbol{K}(x,\mu)$ of the associated Sturm-Liouville transform (SLT).

2.1.2. Tension-modulated nonlinearity

A common assumption when incorporating nonlinear behaviour is to assume spatially uniform tension modulation, leading to the Kirchhoff-Carrier equation [13] with the tension expressed as:

$$\begin{aligned} T_{NL}(y(x,t)) &= T_0 + T_1(y(x,t)) \\ &= T_0 + \frac{EA}{2\ell}\int_0^\ell y'^2(x,t)dx. \end{aligned} \tag{3}$$

Our transformation kernel $\boldsymbol{K}(x,\mu)$ remains the same as in the linear case because our assumption of homogeneous tension still holds [11]. Applying the transformation kernel to the non-linear PDE, we obtain a set of ODEs in modal space:

$$\begin{aligned} &\rho A\ddot{\bar{y}}(\mu,t) + \left(d_3\eta_\mu^2 + d_1\right)\dot{\bar{y}}(\mu,t) + \\ &+ (EI\eta_\mu^4 + T_0\eta_\mu^2)\bar{y}(\mu,t) - \bar{b}(\mu,y,\bar{y}) = 0, \end{aligned} \tag{4}$$

where the non-linear term $\bar{b}(\mu,y,\bar{y})$ in modal space is given by:

$$\bar{b}(\mu,y,\bar{y}) = \eta_\mu^2\left(\frac{EA}{2\ell}\int_0^\ell (y')^2(\xi)d\xi\right)\bar{y}(\mu). \tag{5}$$

These equations can be solved by discretising in time and space and using numerical methods such as the Runge-Kutta method [14].

2.2. Discrete-time dynamical systems

The dynamics of the string can be modelled using the general description of discrete-time dynamical systems

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}(\boldsymbol{x}_k), \tag{6}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ are state variables at discrete time k, and $\boldsymbol{F}$ is a non-linear operator that advances the state. In the linear case, the evolution of the state is given by the state transition matrix $\boldsymbol{A}$, where $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$ and therefore, $\boldsymbol{x}_{k+1} = e^{\boldsymbol{A}\Delta t}\boldsymbol{x}_k$ with a chosen sampling period Δt. The linear operator $\boldsymbol{A}$ for the system can be diagonalised using the similarity transformation

$$\boldsymbol{A} = \boldsymbol{E}^{-1}\tilde{\boldsymbol{A}}\boldsymbol{E}, \tag{7}$$

where $\boldsymbol{E} \in \mathbb{C}^{m\times n}$ and $\boldsymbol{E}^{-1}$ correspond to the inverse and forward SLT [15] kernel. $\tilde{\boldsymbol{A}} \in \mathbb{C}^{m\times m}$ is a diagonal matrix of eigenvalues. The state variables are $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{u} & \boldsymbol{v}\end{bmatrix}^T$, where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the deflection and velocity, respectively. $\boldsymbol{E}$ corresponds to $\boldsymbol{K}(x,\mu)$ of Section 2.1.1 in the case of the linear stiff string.

In the non-linear case, the state evolution is given by the non-linear operator $\boldsymbol{F}$ and, in the general case, it cannot be diagonalised. This motivates the use of deep learning methods. For example, a neural network can be designed to learn the dynamics of the non-linear system by adopting a structure that mirrors this problem formulation.

We consider two methods to learn the dynamics of this system. The first method draws inspiration from Koopman theory and consists of three components: one that transforms the input coordinates (such as position and velocity), a middle component that advances the linearised dynamics through time, and a final component that transforms the states back to the original coordinate system.

The second method relies on deep SSMs. Here, the same three functional components identified in the Koopman-inspired method are integrated into a single layer. By stacking multiple such layers, incorporating nonlinear activations, and adding skip connections, we facilitate the learning of the system's nonlinear dynamics.

2.3. The Koopman operator and the Dynamic Mode Decomposition

Learning the Koopman operator aims to derive a finite approximation to an infinite-dimensional linear operator. This approach essentially transforms the problem of modelling non-linear dynamics into one of linear dynamics in the space of observables [16]. Similarly, Dynamic Mode Decomposition (DMD) can identify and approximate non-linear dynamics using a linear operator [17, 18]. DMD is used to find the best fit, in the least squares sense, for the matrices in Eq. 7 efficiently, even for cases where the state variable vector $\boldsymbol{x}_k$ is very tall. The dynamics under the Koopman operator $\mathcal{K}g = g \circ F$, where the $\circ$ denotes function composition, evolve as:

$$g\left(\boldsymbol{x}_{k+1}\right) = \mathcal{K}g\left(\boldsymbol{x}_k\right). \tag{8}$$

The dynamics found with DMD are equivalent to the Koopman operator when the measurement functions $g(\boldsymbol{x})$ are linear [18]. That is, for systems such as the linear stiff string, it is possible to extract the eigenfunctions $\boldsymbol{K}$ and the eigenvalues in $\tilde{\boldsymbol{A}}$ directly from discrete observations of the system.

2.4. Linear Recurrent Unit and State Space Models

State-space models (SSMs) describe a continuous-time dynamical system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t), \tag{9}$$

where $\boldsymbol{x}(t)$ is the internal state as a function of time, $\boldsymbol{A}$ again is the transition matrix that advances the dynamics of the system. $\boldsymbol{C}$, $\boldsymbol{B}$ transform the input coordinates into state space and viceversa, $\boldsymbol{D}$ is a feed-through matrix (similar to a skip connection) and $\boldsymbol{y}(t)$ is the output. In practice, the SSM is often diagonalised [19], using a similarity transformation (Eq. 7) so that we obtain $\tilde{\boldsymbol{A}}$, in addition to $\tilde{\boldsymbol{B}} = \boldsymbol{E}^{-1}\boldsymbol{B}$ and $\tilde{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{E}$.

During training, the matrices are discretised using bilinear or Zero-Order Hold (ZOH) methods with step parameter Δ. Although these matrices can generally vary over time, deep SSMs [10] and its diagonal variants (DSSMs) [19]) aim to learn the parameters for linear time-invariant systems. More recently, there have been newer techniques to facilitate the optimisation of time-varying parameters [8]. The latter methods (S5) as well as the structured DSSMs initialise these matrices with a predefined structure derived from HiPPO theory [20].

The model then optimises $\mathrm{diag}(\tilde{\boldsymbol{A}}) \in \mathbb{C}^n$ along with the $\tilde{\boldsymbol{B}} \in \mathbb{C}^{m\times n}$, $\tilde{\boldsymbol{C}} \in \mathbb{C}^{n\times m}$ and $\mathrm{diag}(\boldsymbol{D}) \in \mathbb{R}^n$ matrices (the matrix feed-through matrix $\boldsymbol{D}$ is set to be diagonal). Furthermore, the

discretisation step Δ is also learnt. The time-invariant DSSM can be optimised efficiently [21] through the linear convolution of

$$\boldsymbol{y} = \overline{\boldsymbol{K}} * \boldsymbol{u} \tag{10}$$

$$\overline{\boldsymbol{K}} = \left(\overline{\boldsymbol{B}}^{\top} \odot \boldsymbol{C}\right) \cdot \mathcal{V}_L(\overline{\boldsymbol{A}}), \tag{11}$$

where and the matrices $\overline{\boldsymbol{B}}$ and $\overline{\boldsymbol{A}}$ are the discretised versions of $\tilde{\boldsymbol{B}}$ and $\tilde{\boldsymbol{A}}$, $\mathcal{V}_L$ is the Vandermonde matrix of size L, and $\odot$ is the element-wise product. Alternatively, in S5 the diagonal matrices are also efficiently optimised using a parallel scan [22] to parallelise the recursion.

The linear recurrent unit (LRU) [7] follows a very similar formulation but differs in its initialisation strategy. Instead of structured initialisation, the vector $\mathrm{diag}(\tilde{\boldsymbol{A}})$ is initialised with eigenvalues drawn from a uniform distribution within specified bounds in the unit circle.

3. METHOD

To learn the linear and non-linear string dynamics we focus on a Koopman based model with time-varying state.

3.1. Koopman Model

The Koopman-based model leverages an autoencoder framework, in which both the encoder and decoder are designed as multilayer perceptrons (MLPs), to learn the system dynamics in the continuous domain, inspired by previous work [9, 18]. The encoder, denoted by φ, transforms the state of the system into an observable embedding, which is then evolved in time according to the linear operator $\mathcal{K}$:

$$\varphi\left(\mathbf{x}_{k+1}\right) = \mathcal{K}\varphi\left(\mathbf{x}_k\right). \tag{12}$$

Here, the encoder and decoder approximate the Koopman eigenfunctions, which, in the context of a linear stiff string, act as approximate eigenfunctions of the SLT transform.

The main idea for the middle component is to optimise the eigenvalues contained within $\boldsymbol{\Lambda} = \mathrm{diag}(\tilde{\boldsymbol{\Lambda}})$, ensuring that:

$$\mathbf{x}_{k+1} = \varphi^{-1}(\boldsymbol{\Lambda}_{\nu,\theta}\varphi(\mathbf{x}_k)), \tag{13}$$

where the eigenvalues $\boldsymbol{\Lambda} = \left[\lambda_1, \ldots, \lambda_n\right]^{\mathsf{T}}$ are parameterised by their real and imaginary components, $\nu = \Re(\lambda)$ and $\theta = \Im(\lambda)$, in the continuous domain.

Optimisation of the eigenvalues can be approached through recursion or frequency sampling. Learning the linear dynamics of the system in frequency equates to optimising the resonant poles of a parallel filter bank, as per the frequency sampling method [23, 24]. Each filter within this bank acts as a two-pole resonator.

Given the linear dynamics of the model, naive recursion can be markedly slow. Using the parallel scan approach, as employed in the S5 model, can significantly enhance the efficiency of recursion. This technique also facilitates modelling of time-varying dynamics that cannot be captured by static methods such as frequency sampling or convolution. The latter is a helpful feature to capture the time-varying spectra present in systems such as the tension-modulated string [9]. Following this reasoning, we add an MLP that acts on the state of the system, where the input is the square of the real and imaginary components of the state.

For improved optimisation, the parameters are optimised independently, using exponential parameterisation to ensure stability [7]. The magnitude and phase are optimised in logarithmic space, thereby updating the eigenvalue λ_d with

$$\lambda_d := \exp(-\exp(\nu) + i\theta), \tag{14}$$

indicating that optimisation involves $\log(\nu)$ and $\log(\theta)$. This approach mirrors the parameterisation strategies utilised in SSMs and aligns with the design principles adopted for the LRU.

3.2. SSMs with initial conditions

While the formulation of state space models is suited for mapping $\boldsymbol{u} \mapsto \boldsymbol{y}$ (many-to-many), it can adapted to map an initial condition to a sequence $\boldsymbol{u}_0 \mapsto \boldsymbol{u}_{1:L}$ (one-to-many). Thus, the LRU and the S5 model have been adapted to learn a first layer that produces a sequence from initial conditions.

$$\boldsymbol{u_{k+1}} = \overline{\boldsymbol{A}}\boldsymbol{x}_k + \overline{\boldsymbol{B}}\boldsymbol{u}_k\delta_k, \quad \boldsymbol{y}_k = \overline{\boldsymbol{C}}\boldsymbol{x}_k, \tag{15}$$

where $\boldsymbol{u}$ has the initial conditions of the system at $k = 0$ and δ is a unit impulse function. In this setup, the output $\boldsymbol{y}$ must be delayed by one sample to correspond to the samples after the initial condition. Because of this, once trained, it is straightforward to use an arbitrary excitation signal with the SSM model at inference time. Alternatively, the states for the rest of the sequence can be obtained by multiplying the initial condition with the Vandermonde matrix $\boldsymbol{x}_{1:L} = \nu_L(\overline{\boldsymbol{A}})\boldsymbol{x}_0$.

For the linear stiff string, it is necessary, in principle, to optimise only a single layer of such a model. However, the non-linear approximation requires stacking many such layers with non-linear activations and skip connections.

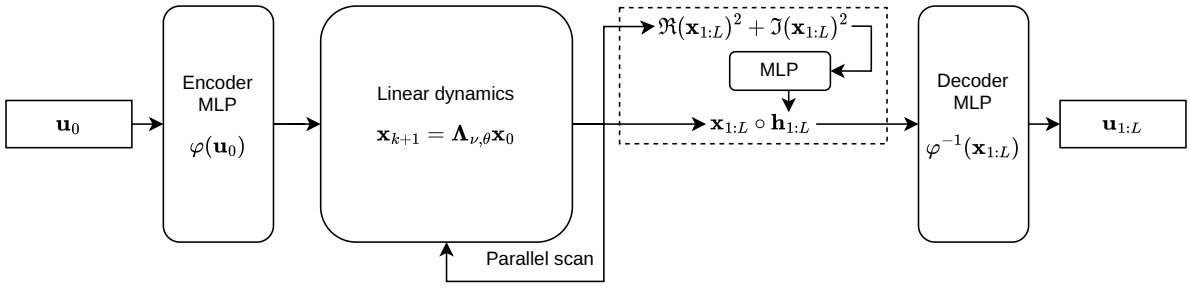
4. EXPERIMENTS

4.1. Dataset

Our experiments were carried out on two types of string models: the linear stiff string and the tension modulated stiff string. We generated a dataset of $L = 4000$ time steps for each model, using sample rates of $4\mathrm{kHz}$ and $16\mathrm{kHz}$. For each sample rate we created 2 sets of 1,000 distinct initial conditions, with either Gaussian or noise-like initial displacement and zero initial velocity. The noise-like initial conditions were randomly generated based on a uniform distribution with the maximum displacements ranging from $1\mathrm{mm}$ to $1\mathrm{cm}$. The Gaussian-like initial conditions were determined by randomly selected means and standard deviations, also with the

Parameter	Value	Unit
A	0.5188×10^{-6}	m^2
I	0.141×10^{-12}	m^4
ρ	1140	$\mathrm{kg\,m^{-3}}$
E	5.4×10^{9}	Pa
d_1	8.0×10^{-5}	$\mathrm{kg\,m^{-1}\,s^{-1}}$
d_3	1.4×10^{-5}	$\mathrm{kg\,m\,s^{-1}}$
T	60.97	N
ℓ	0.65	m

Table 1: Physical parameters of a nylon guitar string, tuned to $\boldsymbol{B}_3$.



(a)

Figure 1: Koopman-based architecture, with optional varying processing for the generated states, as indicated within the dashed box. This involves an MLP, where the eigenvalue radii serve as input, and its output is applied to the state sequence through element-wise multiplication.

same maximum initial displacement range. The dataset was generated by solving the tension-modulated string Eq. 4 numerically using an explicit Runge-Kutta solver of order 8(5,3) (DOP853) [25]. Although both position and velocity were generated for each time step, only the displacement data was used as input for the models. The simulation data was normalised to ensure a standard deviation of 1 prior to training. 80% of the dataset was used for training and another 10% was used for validation.

The selection of these parameters accurately emulates the characteristics of a nylon guitar string tuned to $\boldsymbol{B}_3$ and was in alignment with the experimental setups referenced in previous studies [26, 27]. This approach ensures that our investigations are grounded in realistic parameters, as defined in Table 1.

4.2. Models

We compare six models: an autoencoder without varying state (Koopman), an autoencoder with with varying state (Koopman$_{\text{var}}$), an LRU model [7], an S5 model [8], and the Fourier Neural Operator embedded in the RNN (FRNN) and GRU (FGRU) architectures [3].

The Koopman models and SSMs were trained for 2500 epochs with early stopping, using the AdamW optimiser [28] with a constant learning rate of 0.003. The FNO models were trained for 2000 epochs, using the same optimiser with a 1-cycle learning-rate scheduling ranging from 10^{-4} to 10^{-3}. The hyperparameters for the S5 model were adapted from those used in a speech classification task, with the adjustment of disabling bidirectionality for our specific use case. Similarly, the LRU model employed comparable parameters, but with 8 layers instead of the 6 used for S5. Additionally, the initial bounds for the eigenvalues (for the Koopman and LRU models) are specifically set between 0.99 and 1 within the unit circle. The parameters for the FNO models were the same as those used in previous similar experiments [3], except for the batch size, which was reduced to 256 due to memory limitations caused by the longer sequence (400 time steps instead of the original 120 used for training).

The Koopman model is structured with three dense layers each for the encoder and decoder. The Koopman$_{\text{var}}$ employs a single dense layer for input and output, simplifying the network architecture. Both the SSMs and Koopman-based models internal state representation has size 128, ensuring consistent dimensionality across models.

The models are implemented using JAX. We base our models on the original implementation of S5 [1] and an unofficial implementation of LRU [2]. The FGRU and FRNN models are based on the official PyTorch implementation [3]. An overview of the Koopman model architecture is provided in Fig. 1 and the number of parameters for each model can be found in Table 2.

The training loss for the SSM models is based on the mean square error (MSE) and for the FNO models is the log_{10} MSE, whereas the Koopman models use a composite loss comprising MSE for signal prediction, encoding loss, and consistency loss (also called linear dynamics loss [9]):

$$\mathcal{L}_{consistency} = \sum_{k=1}^{L-1} \|\varphi(\boldsymbol{x}_k) - \Lambda^k \varphi(\boldsymbol{x}_0)\|_2^2 \tag{16}$$

$$\mathcal{L}_{pred} = \sum_{k=1}^{L-1} \|\boldsymbol{x}_k - \varphi^{-1}(\Lambda^k \varphi(\boldsymbol{x}_0))\|_2^2 \tag{17}$$

$$\mathcal{L}_{enc} = \|\boldsymbol{x}_0 - \varphi^{-1}(\varphi(\boldsymbol{x}_0))\|_2^2 \tag{18}$$

$$\mathcal{L} = \alpha_1 \mathcal{L}_{pred} + \alpha_2 \mathcal{L}_{enc} + \alpha_3 \mathcal{L}_{consistency}. \tag{19}$$

The encoding loss ensures that the decoder accurately reverses the encoder's transformation, while the consistency loss enforces that the observed state evolution and the projection operation are commutative. The loss weights were set to $\alpha_1 = 1.0$, $\alpha_2 = 1.0$, and $\alpha_3 = 0.01$.

Model	Trainable Parameters
Koopman	191,077
Koopman$_{\text{var}}$	134,885
LRU	636,221
S5	307,879
FGRU	640,033
FRNN	643,201

Table 2: Number of trainable parameters for each model.

[1] https://github.com/lindermanlab/S5
[2] https://github.com/NicolasZucchet/minimal-LRU
[3] https://github.com/julian-parker/DAFX22_FNO

	Gaussian				Noise			
	4000kHz		16000kHz		4000kHz		16000kHz	
Model	**MSE Rel**	**MAE Rel**	**MSE Rel**	**MAE Rel**	**MSE Rel**	**MAE Rel**	**MSE Rel**	**MAE Rel**
DMD	1.1865(0.2019)	0.8289(0.0857)	1.0938(0.1311)	0.8747(0.0749)	1.6695(0.2687)	1.0420(0.1178)	1.8313(0.2467)	1.0953(0.087)
Koopman	0.1041(0.0144)	0.3271(0.0259)	0.0971(0.0029)	0.3117(0.0044)	0.2070(0.0939)	0.3978(0.1084)	0.0865(0.0155)	0.2596(0.0275)
$\text{Koopman}_{\text{var}}$	**0.0113**(0.0061)	**0.0977**(0.0284)	**0.0094**(0.0046)	**0.0888**(0.0245)	**0.0041**(0.0008)	**0.0527**(0.0081)	**0.0531**(0.0572)	**0.1714**(0.1056)
LRU	0.0260(0.0067)	0.1571(0.0240)	0.0390(0.0187)	0.1870(0.0450)	0.0250(0.0065)	0.1284(0.0077)	0.0476(0.0058)	0.1922(0.0125)
S5	0.1165(0.0260)	0.3318(0.0291)	0.0224(0.0013)	0.1467(0.0053)	0.0507(0.0127)	0.1916(0.0190)	0.0387(0.0062)	0.1493(0.0078)
Models below are trained and evaluated with only 400 time steps								
DMD	0.4519(0.0000)	0.5570(0.0005)	0.4263(0.0000)	0.5415(0.0007)	0.2304(0.0000)	0.3300(0.0009)	0.4192(0.0000)	0.4959(0.0012)
Koopman	0.0031(0.0005)	0.0499(0.0050)	0.0071(0.0002)	0.0946(0.0022)	0.0036(0.0007)	0.0353(0.0052)	0.0393(0.0013)	0.1767(0.0020)
$\text{Koopman}_{\text{var}}$	0.0008(0.0003)	**0.0245(0.0054)**	**0.0003(0.0000)**	**0.0201(0.0006)**	**0.0005(0.0000)**	**0.0132(0.0009)**	**0.0118(0.0009)**	**0.0720(0.0022)**
LRU	**0.0008(0.0000)**	0.0252(0.0011)	0.0013(0.0001)	0.0376(0.0022)	0.0008(0.0001)	0.0189(0.0018)	0.0259(0.0009)	0.1357(0.0031)
S5	0.0124(0.0032)	0.0910(0.0131)	0.0131(0.0033)	0.1145(0.0172)	0.0057(0.0011)	0.0672(0.0090)	0.0357(0.0048)	0.1721(0.0084)
FGRU	0.4534(0.0219)	0.6897(0.0204)	0.0388(0.0151)	0.2164(0.0464)	0.0870(0.0249)	0.2925(0.0444)	0.0583(0.0053)	0.2393(0.0114)
FRNN	0.6927(0.1655)	0.8489(0.0925)	0.7636(0.0883)	0.9311(0.0338)	0.7289(0.0653)	0.8553(0.0495)	0.5611(0.0893)	0.7623(0.0448)

(a) Results for the non-linear dataset.

	Gaussian				Noise			
	4000kHz		16000kHz		4000kHz		16000kHz	
Model	**MSE Rel**	**MAE Rel**	**MSE Rel**	**MAE Rel**	**MSE Rel**	**MAE Rel**	**MSE Rel**	**MAE Rel**
DMD	**0.0000**(0.0000)	**0.0060**(0.0007)	**0.0001**(0.0000)	**0.0097**(0.0006)	**0.0000**(0.0000)	**0.0042**(0.0004)	**0.0001**(0.0000)	**0.0066**(0.0004)
Koopman	0.0110(0.0069)	0.1005(0.0331)	0.0048(0.0009)	0.0692(0.0074)	0.0073(0.0052)	0.0789(0.0322)	0.0019(0.0001)	0.0423(0.0004)
$\text{Koopman}_{\text{var}}$	0.0020(0.0013)	0.0427(0.0145)	0.0013(0.0012)	0.0321(0.0142)	0.0002(0.0001)	0.0134(0.0023)	0.0495(0.0973)	0.1125(0.1785)
LRU	0.0014(0.0005)	0.0373(0.0065)	0.0010(0.0002)	0.0317(0.0034)	0.0005(0.0001)	0.0199(0.0025)	0.0005(0.0002)	0.0217(0.0036)
S5	0.0386(0.0273)	0.1896(0.0597)	0.0156(0.0102)	0.1206(0.0421)	0.0112(0.0029)	0.1023(0.0139)	0.0037(0.0012)	0.0539(0.0119)
Models below are trained and evaluated with only 400 time steps								
DMD	**0.0000(0.0000)**	**0.0013(0.0000)**	**0.0000(0.0000)**	**0.0006(0.0000)**	**0.0000(0.0000)**	**0.0012(0.0000)**	**0.0000(0.0000)**	**0.0005(0.0000)**
Koopman	0.0002(0.0000)	0.0140(0.0014)	0.0004(0.0002)	0.0243(0.0045)	0.0001(0.0000)	0.0088(0.0005)	0.0003(0.0001)	0.0169(0.0021)
$\text{Koopman}_{\text{var}}$	0.0000(0.0000)	0.0058(0.0019)	0.0000(0.0000)	0.0074(0.0018)	0.0000(0.0000)	0.0056(0.0006)	0.0000(0.0000)	0.0055(0.0004)
LRU	0.0002(0.0000)	0.0140(0.0012)	0.0002(0.0000)	0.0171(0.0003)	0.0001(0.0000)	0.0091(0.0005)	0.0002(0.0000)	0.0148(0.0012)
S5	0.0017(0.0005)	0.0425(0.0068)	0.0043(0.0015)	0.0616(0.0130)	0.0017(0.0007)	0.0363(0.0088)	0.0108(0.0009)	0.0944(0.0042)
FGRU	0.4399(0.0258)	0.6825(0.023)	0.0378(0.0162)	0.214(0.0551)	0.0681(0.0198)	0.2618(0.0385)	0.0470(0.0034)	0.2145(0.0082)
FRNN	0.6474(0.1711)	0.8183(0.1198)	0.7242(0.1912)	0.8925(0.1281)	0.5821(0.3888)	0.6870(0.3413)	0.4730(0.3205)	0.6347(0.2836)

(b) Results for the linear dataset.

Table 3: Mean and standard deviation (in parentheses) for 5 seeds of a) non-linear and b) linear validation data across different models and sampling rates, under Gaussian and noise-like initial conditions. We use 4000 steps for both 4kHz and 16kHz. Additionally, the models are also trained and evaluated with 400 time steps to compare with the FRNN and FGRU methods. For DMD, the standard deviation of MSE and MAE across the validation dataset is included as it does not depend on a seed.

4.3. Results

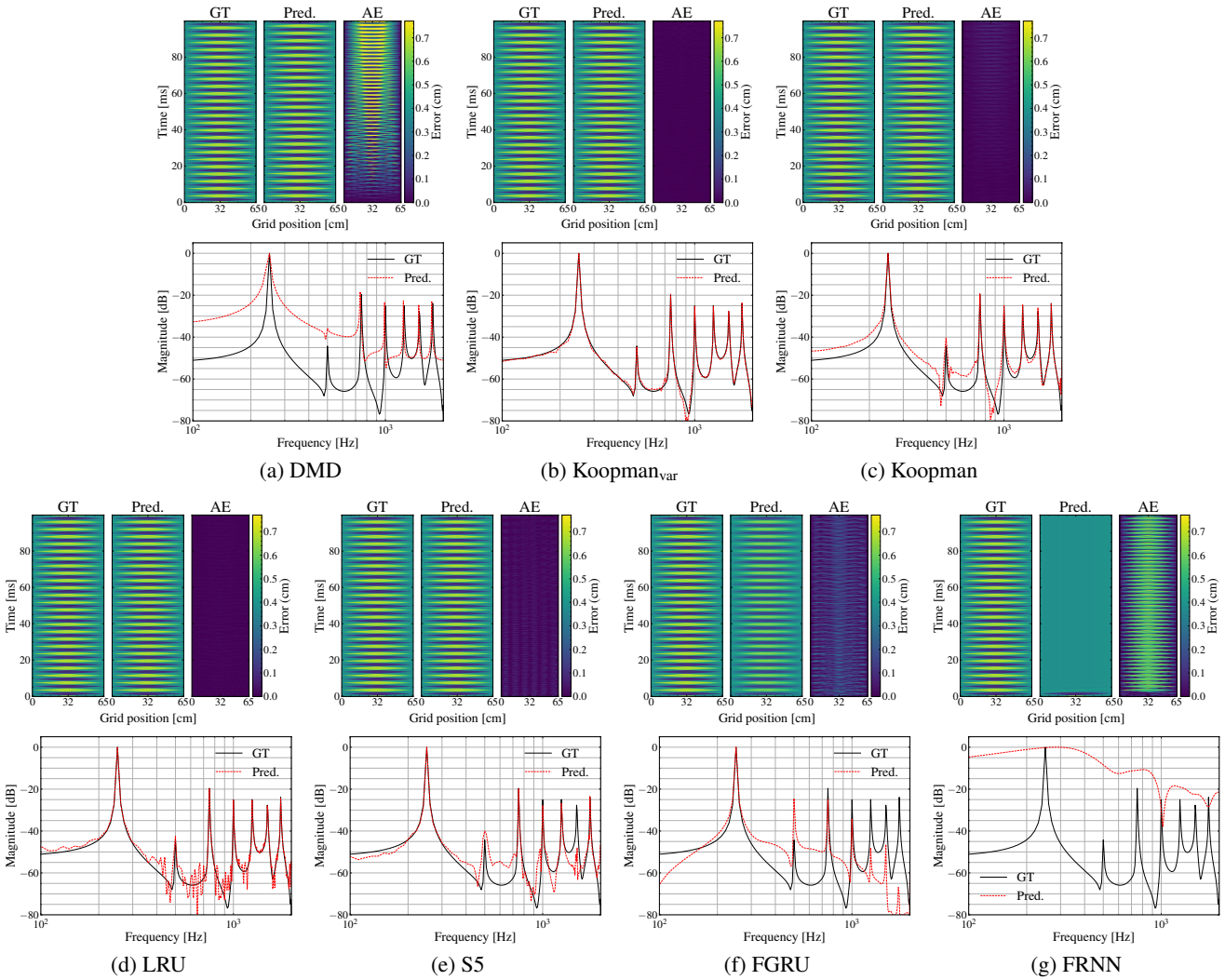
The results of our experiments are summarised in Tables 3a for nonlinear datasets and 3b for linear datasets, showing the relative mean absolute error (MAE) and the relative mean square error (MSE). Additional results and visualisations can be found in the accompanying website [4]. We use relative metrics, normalised with respect to the norm (squared in the case of MSE) of the string displacement trajectory.

In addition, we present metrics for the DMD results as a baseline. It is important to note that DMD struggles to capture the dynamics of systems such as ours, characterised by standing waves, due to the problem of linear consistency across consecutive time steps [17]. To address this limitation, we augment the input data with a Hankel matrix constructed from two lags of the input [29]. Consequently, this comparison might seem skewed, since the other methods effectively work with fewer data, making the evaluation somewhat unequal. Additionally, the DMD only fits a single sequence. To ensure a reasonable fit, we use a fixed amplitude of 0.0055 which is the mean of the variation in the amplitude of the training data and set the singular value rank to $r = 50$. We observe that using $r > 50$ singular value degrades the accuracy of the DMD prediction.

Our findings indicate that all models, except for the FRNN model, demonstrate the ability to capture the system's dynamics, though their accuracy varies. In particular, the $\text{Koopman}_{\text{var}}$ model recorded the lowest relative MAE in non-linear scenarios. The predicted displacement evolution of the model closely matches the ground truth data, as shown in Figures 2 and 3. This accuracy is also reflected in the frequency content of the sequence. In the linear case, DMD performs better than the neural models. This is expected, as the optimal solution to the linear case can be achieved through the linear fitting of the DMD.

The FNO models perform worse than the other models (trained with 400 time steps), with the FRNN being significantly unstable during training and inference. Figure 2 shows the inference of the

[4] https://rodrigodzf.github.io/physmodjax/results.html



(a) DMD (b) Koopman$_{\text{var}}$ (c) Koopman

(d) LRU (e) S5 (f) FGRU (g) FRNN

Figure 2: Non-linear dynamics for an unseen uniform noise-like initial condition in the range 0 to 1cm. The top row for each model shows the displacement evolution along the string for 400 time steps (100ms). The bottom row for each model displays the spectrum of the same section at a single point ($\approx$ 24cm). All models were trained on 400 time steps from the same dataset.

FNO models at their last stable checkpoint with the lowest validation MAE (approximately 500 epochs for the FRNN model).

4.3.1. Extrapolation in Time

Although the models can generalise across different initial conditions, their capability in extrapolating beyond the predefined training horizon of 4000 time steps is limited. As depicted in Figure 4, the models' predictions start to significantly deviate from the ground truth beyond the final training step. As expected, an exception exists with the DMD's performance in linear scenarios, where it provides the optimal linear solution. The predictions made by the Koopman$_{\text{var}}$ model begin to diverge from the actual data more rapidly than those made by the SSM models, which demonstrate slightly better accuracy for a marginally longer duration. In the case of the S5 model, we did not clip the eigenvalues during training, resulting in more unstable long-term inference. The divergence in the Koopman$_{\text{var}}$ model is mostly present in the higher frequencies and generally manifests as a shorter decay, as shown in Figure 3.

This limitation in temporal extrapolation likely stems from the inherent non-linearities within the models and the exponential parameterisation used for their stability. These features are beneficial for learning dynamics within the observed timeframe and for enforcing stability, but they become problematic for predictions that extend beyond the trained interval. Unlike the FGRU or FRNN models, the Koopman-based models and SSMs remain mostly stable in the extrapolated region.

It is important to note that the models were trained exclusively on displacement data; velocity information was not provided as input in our experiments. Representing the state of the system solely through displacement or its embedding might be insufficient for an unequivocal state description.

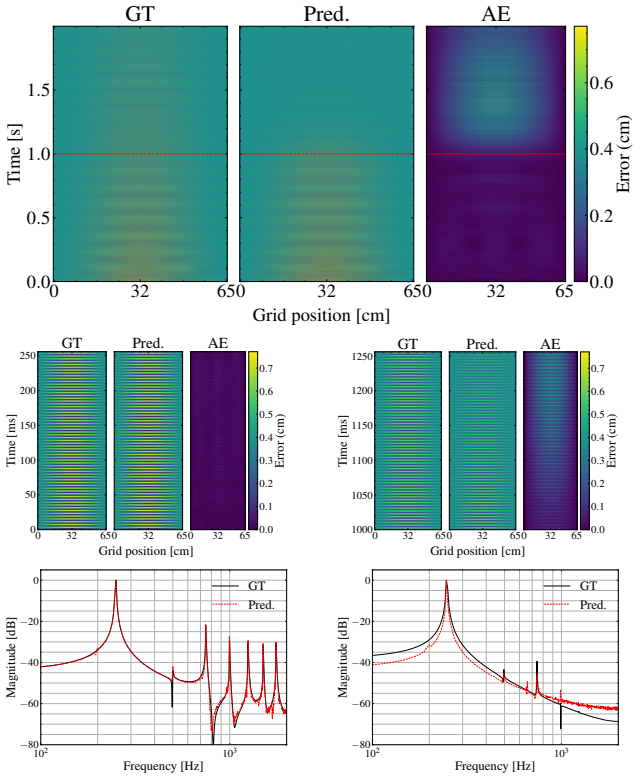


Figure 3: Evolution of the predicted displacement and extrapolation beyond the training horizon for a uniform noise-like initial condition (0 to 1cm). The top row shows the $\text{Koopman}_{\text{var}}$ extrapolation with absolute error (in centimetres). The middle plots zoom in on the first 0.25 seconds (left) and 0.25 seconds after the training horizon (right). The bottom plots show the corresponding spectrum at a single position on the string ($\approx$ 24 cm) for the time spans covered in the middle plots.

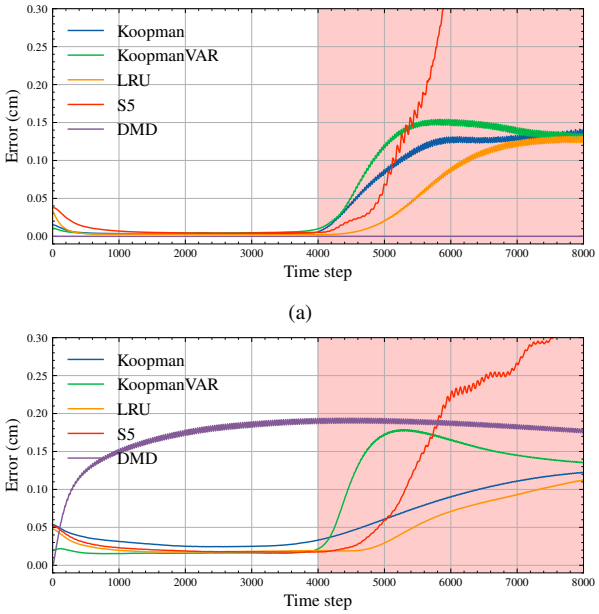


Figure 4: Mean absolute error in centimetres per timestep across 100 unseen test trajectories for each architecture for 4000 time steps. Results are shown for a random noise-like initial condition (0 to 1cm) (a) for a linear string and (b) for a tension-modulated string at a single position on the string ($\approx$ 24 cm). The regions shaded in red represent the predictions after the training time step horizon. The results for the FGRU and FRNN models are not included as these could not be trained with more than 400 steps. The error curves are smoothed for easier visualisation.

5. CONCLUSIONS

We have explored SSM and Koopman-based methods for modelling of the dynamics of linear and tension-modulated strings, providing a comprehensive comparison with similar methods. Our experimental results highlight the viability of Koopman-based methods in effectively modelling these systems over extended time horizons, though challenges remain.

A significant finding from our research is the better or at least comparable performance of the state-varying Koopman method compared to the SSM and FNO models in predicting the dynamics of the string. This finding underscores the potential efficiency of Koopman-based models, which, despite having fewer parameters 1, may be better suited to such dynamic systems.

Looking ahead, our findings open up several avenues for further investigation. We believe that it might be possible to mitigate the issue with extrapolation in time either by augmenting the input (e.g., by providing two concatenated time steps at each step, similar to the Hankel DMD approach [29]) or by incorporating initial velocity information. The latter strategy, in particular, could equip the models with the necessary information to initiate dynamic evolution from a later state, akin to starting from a new initial condition. The absence of velocity data means missing essential phase information needed for long-term state predictions from subsequent starting points.

Another challenge to address is adapting our models to different physical parameters. Currently, the model learns the trajectory for a single set of physical parameters. However, the Koopman and SSM models could be adapted with an additional layer that encodes not only the initial condition but also the corresponding physical parameters. Additionally, we intend to explore the interpretability of the learned weights in the Koopman model. It may also be possible to directly manipulate these weights for more creative uses. Expanding the application of these modelling techniques to two and three dimensional domains represents another possible extension of our method. Such efforts could test the adaptability of our methods to more complex problems in multidimensional systems.

In summary, our exploration of SSMs and Koopman-based modelling techniques reveals a promising landscape for future research in simulating the dynamics of acoustical systems. By addressing the identified limitations and exploring proposed future directions, we can continue to expand the boundaries of what is possible in physical modelling with neural networks.

6. ACKNOWLEDGMENTS

This research utilised Queen Mary's Apocrita HPC facility, supported by QMUL Research-IT. http://doi.org/10.5281/zenodo.438045

7. REFERENCES

[1] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, Wiley, 1 edition, Oct. 2009.

[2] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier Neural Operator for Parametric Partial Differential Equations," in *International Conference on Learning Representations*, Oct. 2020.

[3] J. Parker, S. Schlecht, M. Schäfer, and R. Rabenstein, "Physical Modeling Using Recurrent Neural Networks with Fast Convolutional Layers," in *Proceedings of the 25th International Conference on Digital Audio Effects (DAFx20in22)*. 2022, pp. 138–145, DAFx.

[4] C. De La Vega Martin and M. Sandler, "Physical Modelling of Stiff Membrane Vibration using Neural Networks with Spectral Convolution Layers," in *Forum Acusticum 2023*, 2023.

[5] K. Michałowska, S. Goswami, G. E. Karniadakis, and S. Riemer-Sørensen, "Neural Operator Learning for Long-Time Integration in Dynamical Systems with Recurrent Neural Networks," May 2023.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[7] A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De, "Resurrecting recurrent neural networks for long sequences," in *Proceedings of the 40th International Conference on Machine Learning*. July 2023, vol. 202 of *ICML'23*, pp. 26670–26698, JMLR.org.

[8] J. T. H. Smith, A. Warrington, and S. Linderman, "Simplified State Space Layers for Sequence Modeling," in *The Eleventh International Conference on Learning Representations*, Sept. 2022.

[9] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, pp. 4950, 2018.

[10] A. Gu, K. Goel, and C. Re, "Efficiently Modeling Long Sequences with Structured State Spaces," in *International Conference on Learning Representations*, Mar. 2022.

[11] L. Trautmann and R. Rabenstein, "Sound Synthesis with Tension Modulated Nonlinearities Based on Functional Transformations," in *Proceedings of the Conference on Acoustics and Music: Theory and Applications*, Montego Bay, Jamaica, Dec. 2000.

[12] F. Avanzini, R. Marogna, and B. Bank, "Efficient synthesis of tension modulation in strings and membranes based on energy estimation," *The Journal of the Acoustical Society of America*, vol. 131, no. 1, pp. 897–906, Jan. 2012.

[13] G. F. Carrier, "On the non-linear vibration problem of the elastic string," *Quarterly of Applied Mathematics*, vol. 3, no. 2, pp. 157–165, Jan. 1945.

[14] E. W. Cheney and D. R. Kincaid, *Numerical Mathematics and Computing*, Cengage Learning, 7th edition edition, May 2012.

[15] M. Schäfer, *Simulation of Distributed Parameter Systems by Transfer Function Models*, Ph.D. thesis, Friedrich-Alexander-Universtität Erlangen-Nürnberg (FAU), Feb. 2020.

[16] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, Dec. 2015.

[17] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *Journal of Computational Dynamics*, vol. 1, no. 2, pp. 391–421, 2014.

[18] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, "Modern Koopman Theory for Dynamical Systems," *SIAM Review*, vol. 64, no. 2, pp. 229–340, 2022.

[19] A. Gupta, A. Gu, and J. Berant, "Diagonal State Spaces are as Effective as Structured State Spaces," in *Advances in Neural Information Processing Systems*, May 2022.

[20] A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Re, "How to Train your HIPPO: State Space Models with Generalized Orthogonal Basis Projections," in *International Conference on Learning Representations*, Sept. 2022.

[21] A. Gu, K. Goel, A. Gupta, and C. Ré, "On the Parameterization and Initialization of Diagonal State Space Models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 35971–35983, Dec. 2022.

[22] G. E. Blelloch, "Prefix sums and their applications," Tech. Rep., School of Computer Science, Carnegie Mellon University, 1990.

[23] 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 *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2022, pp. 3104–3108.

[24] R. Diaz, B. Hayes, C. Saitis, G. Fazekas, and M. Sandler, "Rigid-Body Sound Synthesis with Differentiable Modal Resonators," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2023, pp. 1–5.

[25] E. Hairer, G. Wanner, and S. P. Nørsett, "Runge-Kutta and Extrapolation Methods," in *Solving Ordinary Differential Equations I: Nonstiff Problems*, pp. 129–353. Springer, Berlin, Heidelberg, 1993.

[26] M. Schäfer, S. Schlecht, and R. Rabenstein, "A String In A Room: Mixed-Dimensional Transfer Function Models For Sound Synthesis," in *International Conference on Digital Audio Effects*, 2020, pp. 187–194.

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

[28] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations*, Sept. 2018.

[29] H. Arbabi and I. Mezić, "Ergodic Theory, Dynamic Mode Decomposition, and Computation of Spectral Properties of the Koopman Operator," *SIAM Journal on Applied Dynamical Systems*, vol. 16, no. 4, pp. 2096–2126, Jan. 2017.