LEARNING NONLINEAR DYNAMICS IN PHYSICAL MODELLING SYNTHESIS USING NEURAL ORDINARY DIFFERENTIAL EQUATIONS

Victor Zheleznov [*], *Stefan Bilbao, Alec Wright*

Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
{v.zheleznov, s.bilbao, alec.wright}@ed.ac.uk

Simon King

Centre for Speech Technology Research
University of Edinburgh
Edinburgh, UK
simon.king@ed.ac.uk

ABSTRACT

Modal synthesis methods are a long-standing approach for modelling distributed musical systems. In some cases extensions are possible in order to handle geometric nonlinearities. One such case is the high-amplitude vibration of a string, where geometric nonlinear effects lead to perceptually important effects including pitch glides and a dependence of brightness on striking amplitude. A modal decomposition leads to a coupled nonlinear system of ordinary differential equations. Recent work in applied machine learning approaches (in particular neural ordinary differential equations) has been used to model lumped dynamic systems such as electronic circuits automatically from data. In this work, we examine how modal decomposition can be combined with neural ordinary differential equations for modelling distributed musical systems. The proposed model leverages the analytical solution for linear vibration of system's modes and employs a neural network to account for nonlinear dynamic behaviour. Physical parameters of a system remain easily accessible after the training without the need for a parameter encoder in the network architecture. As an initial proof of concept, we generate synthetic data for a nonlinear transverse string and show that the model can be trained to reproduce the nonlinear dynamics of the system. Sound examples are presented.

1. INTRODUCTION

Research into physical modelling synthesis has a long history. Various simulation techniques have been employed, including finite-difference time-domain methods [1], modal synthesis [2] and port-Hamiltonian methods [3]. These approaches all rely on the numerical solution of a set of ordinary or partial differential equations (ODEs/PDEs) that describe the dynamics of a system and are accompanied by initial and boundary conditions, and external forces or input signals. In contrast, machine learning approaches generally construct musical systems automatically from data, often in a black-box manner, and recently have gained popularity in audio research, especially in virtual-analog modelling [4]. Some data-driven approaches — such as neural ordinary differential equations (NODEs), in which a derivative of system's state vector is parameterised by a neural network [5] — incorporate existing knowledge about the underlying system into their training [6]. As electronic circuits can generally be viewed as finite-dimensional systems, they are well-modelled using NODEs [7].

However, modelling of distributed musical systems such as strings, plates, etc. using machine learning approaches has seen limited attention in the literature. Parker et al. [8] have presented recurrent neural networks for physical modelling based on a Fourier neural operator [9]. These recurrent structures are trained from data spanning a few initial time steps of around 2 ms. Extrapolation in time is tested over intervals spanning up to 10 times that seen during training and the best performing model shows degradation towards the end of the simulation. Diaz et al. [10] have introduced a Koopman-based model that addresses some discrepancies compared with recurrent architectures such as solution accuracy but extrapolation in time remains a challenge. Lee et al. [11] have employed differentiable digital signal processing techniques for the simulation of a nonlinear string. Amplitudes and frequencies of oscillators, corresponding to modes of a string, are adjusted using multilayer perceptron (MLP) blocks, which are trained to capture nonlinear effects.

One common drawback of these approaches is that initial conditions and physical parameters of a system, affecting pitch, timbre and other sonic characteristics, can not be modified after training, or the network architecture requires a parameter encoder to condition the solution, leading to more trainable parameters and the requirement of a larger dataset containing ground truth data for desired configurations of a system. In addition, system excitation follows from different choices of initial conditions which does not correspond to a realistic playing scenario, where external forcing terms are always present [1]. In this work, we aim to more tightly integrate the physics of a distributed system into a machine learning framework. In particular, we use modal decomposition to construct a system of finite dimension and separate the linear and nonlinear parts of the problem. Then, we replace only a dimensionless memoryless nonlinearity (that describes coupling between the modes) with a neural network and train NODEs to obtain the resulting model. Consequently, physical parameters remain easily accessible and the model generalises to physical parameters, sampling rates and time scales not seen during training. Compared to a fully physical model, we show that the proposed approach is more computationally efficient for the case of nonlinear transverse string vibration.

The paper is organised as follows. A simple model of nonlinear transverse string vibration is described in Section 2. Section 3 derives modal equations for a string, which are discretised in time for computer simulation in Section 4. Section 5 outlines the proposed learning algorithm, which is evaluated in Section 6 across several case studies. Sound examples are available on the accom-

[*] This work was supported by the SGSAH AHRC Doctoral Training Partnership [grant number AH/R012717/1]; and the University of Edinburgh.

panying page[1].

2. NONLINEAR TRANSVERSE STRING VIBRATION

The general equation of motion describing a transverse vibration of a nonlinear string in a single polarisation is:

$$\mathcal{L}u = \mathcal{F} + \mathcal{F}_\mathrm{e}. \tag{1}$$

Here $u = u(x,t)$: $[0, L] \times \mathbb{R}^+ \to \mathbb{R}$ denotes the transverse displacement of a string of length L and depends on spatial coordinate x in m and time t in sec. Initial conditions are assumed to be zero. The string is assumed to be simply supported at both ends, implying the following boundary conditions:

$$u(0,t) = \partial_x^2 u(0,t) = u(L,t) = \partial_x^2 u(L,t) = 0, \quad \forall t \in \mathbb{R}^+.$$

Output is assumed to be drawn directly from the string displacement at position x_o as $w(t) = u(x_\mathrm{o}, t)$.

2.1. Linear Vibration

The linear part of the string vibration is encapsulated in the operator $\mathcal{L}$, defined as:

$$\mathcal{L} = \rho A \partial_t^2 - T\partial_x^2 + EI\partial_x^4 + 2\sigma_0\rho A\partial_t - 2\sigma_1\rho A\partial_t\partial_x^2, \tag{2}$$

where ∂_t and ∂_x represent partial derivatives with respect to x and t, respectively. Physical parameters that appear in (2) are: the material density ρ in $\mathrm{kg}{\cdot}\mathrm{m}^{-3}$; the string cross-sectional area $A = \pi r^2$ in m^2 for a string of radius r; the tension T in N; Young's modulus E in $\mathrm{N}{\cdot}\mathrm{m}^{-2}$; and moment of inertia $I = \frac{1}{4}\pi r^4$ in m^4. Frequency-independent and dependent loss is characterised by parameters $\sigma_0 \geq 0$ and $\sigma_1 \geq 0$, respectively. See [1] for more on this term in the context of linear string vibration.

2.2. Nonlinearity

Nonlinear dynamics of the string are described in a force density $\mathcal{F}$. A general model for $\mathcal{F}$ is given by Morse and Ingard [12] and includes both longitudinal and transverse motion of a string in two polarisations. In this work we neglect the longitudinal motion and one of the two polarisations, leading to the following force density after taking a Taylor series expansion on the force potential [13]:

$$\mathcal{F}(x,t) = \partial_x\left(\frac{EA - T}{2}\xi^3\right), \quad \xi \triangleq \partial_x u. \tag{3}$$

For the force density (3) to be conservative we assume $EA \geq T$, which is true in the case of musical strings. Compared to the Kirchhoff-Carrier model [14, 15], which adequately reproduces only the pitch glide effect, model (3) is capable of capturing other perceptually important effects such as phantom partials [1].

2.3. Plucking Excitation

The string is excited by a pointwise external force $\mathcal{F}_\mathrm{e}$, which can be modelled as:

$$\mathcal{F}_\mathrm{e}(x,t) = \delta(x - x_\mathrm{e}) f_\mathrm{e}(t),$$

where $\delta(x - x_\mathrm{e})$ is the Dirac delta function at the excitation position x_e. Function $f_\mathrm{e}(t)$ resembles a pluck of a string and is of the following form [16]:

$$f_\mathrm{e}(t) = \begin{cases} \frac{1}{2} f_\mathrm{amp}\left[1 - \cos\left(\frac{\pi t}{T_\mathrm{e}}\right)\right], & t \in [0, T_\mathrm{e}] \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Here f_amp is the excitation amplitude in N and T_e is the excitation duration in sec. The excitation starting time is assumed to be zero.

2.4. Equation Scaling

In view of using the string model (1) for dataset generation, it is useful to reduce the number of physical parameters to the smallest possible set. Firstly, we employ spatial scaling by introducing a normalised position variable $x' = \frac{x}{L} \in [0, 1]$:

$$\begin{aligned} \partial_t^2 u = \gamma^2\partial_{x'}^2 u - \kappa^2\partial_{x'}^4 u - 2\sigma_0\partial_t u + 2\sigma_1'\partial_t\partial_{x'}^2 u + \\ + \gamma^2\partial_{x'}\left(\frac{1}{2L^2}\left(\frac{EA}{T} - 1\right)(\partial_{x'}u)^3\right) + \frac{1}{\rho AL}\delta(x' - x_\mathrm{e}')f_\mathrm{e}(t), \end{aligned}$$

where $\gamma = \frac{1}{L}\sqrt{\frac{T}{\rho A}}$, $\kappa = \frac{1}{L^2}\sqrt{\frac{EI}{\rho A}}$ and $\sigma_1' = \frac{\sigma_1}{L^2}$.

Secondly, we scale the displacement by $u' = u{\cdot}u_0$ and obtain:

$$\begin{aligned} \partial_t^2 u' = \gamma^2\partial_{x'}^2 u' - \kappa^2\partial_{x'}^4 u' - 2\sigma_0\partial_t u' + 2\sigma_1'\partial_t\partial_{x'}^2 u' + \\ + \gamma^2\partial_{x'}(\xi'^3) + \delta(x' - x_\mathrm{e}')f_\mathrm{e}'(t), \end{aligned} \tag{5}$$

where $u_0 = \frac{1}{L}\sqrt{\frac{1}{2}(\frac{EA}{T} - 1)}$, $\xi' \triangleq \partial_{x'}u'$ and $f_\mathrm{e}'(t) = \frac{u_0}{\rho AL}f_\mathrm{e}(t)$.

Thus, we have reduced a set of physical parameters $\{L, \rho, A, T, E, I, \sigma_0, \sigma_1\}$ to a set of only four parameters $\{\gamma, \kappa, \sigma_0, \sigma_1'\}$. In the following sections we omit the prime while referring to the scaled string model (5).

3. MODAL DECOMPOSITION

The solution to equation (5) can be decomposed into a set of modes, yielding a finite-dimensional system when truncated to finite order M. The transverse displacement u is rewritten as a superposition of modal displacements $\boldsymbol{q}(t) = [q_1(t), \ldots, q_M(t)]^\mathrm{T}$:

$$u(x,t) = \sum_{m=1}^{M} \Phi_m(x)q_m(t) = \boldsymbol{\Phi}^\mathrm{T}(x)\boldsymbol{q}(t). \tag{6}$$

Here modal shapes $\Phi_m(x) = \sqrt{2}\sin(m\pi x),\ m = 1, \ldots, M$ correspond to the solution of the eigenvalue problem for a stiff string under simply supported boundary conditions [1].

Substituting (6) into (5), left-multiplying by $\boldsymbol{\Phi}$ and taking an L^2 inner product over the interval $[0, 1]$, we obtain the following second-order system of ODEs:

$$\ddot{\boldsymbol{q}} + 2\mathbf{S}\dot{\boldsymbol{q}} + \boldsymbol{\Omega}^2\boldsymbol{q} = \gamma^2\boldsymbol{f}(\boldsymbol{q}) + \boldsymbol{\Phi}(x_\mathrm{e})f_\mathrm{e}(t), \tag{7}$$

where $\mathbf{S}$ and $\boldsymbol{\Omega}$ are $M \times M$ diagonal matrices, defined using modal wavenumbers $\beta_m = m\pi,\ m = 1, \ldots, M$ as:

$$\mathrm{S}_{m,m} = \sigma_0 + \sigma_1\beta_m^2, \quad \Omega_{m,m} = \sqrt{\gamma^2\beta_m^2 + \kappa^2\beta_m^4}.$$

[1] https://victorzheleznov.github.io/dafx25

3.1. Nonlinearity

To obtain a closed-form expression for dimensionless nonlinearity $\boldsymbol{f}(\boldsymbol{q})$: $\mathbb{R}^M \to \mathbb{R}^M$, we firstly rewrite the partial derivative ξ in the modal form:

$$\xi = \partial_x \left(\sum_{i_1=1}^{M} \Phi_{i_1} q_{i_1} \right) = \sqrt{2}\pi \sum_{i_1=1}^{M} i_1 \underbrace{\cos(i_1 \pi x)}_{\triangleq c_{i_1}} q_{i_1}. \tag{8}$$

Raising (8) to the power of 3 and taking a partial derivative with respect to x, we get:

$$\begin{aligned}\partial_x\big(\xi^3\big) = -2\sqrt{2}\pi^4 \sum_{i_1=1}^{M}\sum_{i_2=1}^{M}\sum_{i_3=1}^{M} \Big(& i_1^2 i_2 i_3 s_{i_1} c_{i_2} c_{i_3} + \\ &+ i_1 i_2^2 i_3 c_{i_1} s_{i_2} c_{i_3} + i_1 i_2 i_3^2 c_{i_1} c_{i_2} s_{i_3}\Big) q_{i_1} q_{i_2} q_{i_3},\end{aligned} \tag{9}$$

where $s_{i_k} \triangleq \sin(i_k \pi x),\ k = 1, 2, 3$. Multiplying (9) by Φ_m and taking an L^2 inner product over the interval $[0, 1]$, we find:

$$f_m(\boldsymbol{q}) = \int_0^1 \Phi_m \partial_x\big(\xi^3\big)\, dx = -\sum_{i_1=1}^{M}\sum_{i_2=1}^{M}\sum_{i_3=1}^{M} A^m_{i_1,i_2,i_3} q_{i_1} q_{i_2} q_{i_3}, \tag{10}$$

where:

$$\begin{aligned} A^m_{i_1,i_2,i_3} &= \frac{\pi^4}{2}\Big[B^{i_1,m}_{i_2,i_3} + B^{i_2,m}_{i_3,i_1} + B^{i_3,m}_{i_1,i_2}\Big], \\ B^{k,m}_{i,j} &= ijk^2\big[\ \delta_{k+i,m+j} - \delta_{k+i,-(m+j)} + \\ &\quad + \delta_{k+i,m-j} - \delta_{k+i,-(m-j)} + \\ &\quad + \delta_{k-i,m+j} - \delta_{k-i,-(m+j)} + \\ &\quad + \delta_{k-i,m-j} - \delta_{k-i,-(m-j)}\big]. \end{aligned} \tag{11}$$

Here $\delta_{i,j}$ denotes the Kronecker delta function. Thus, the dimensionless nonlinearity $\boldsymbol{f}(\boldsymbol{q})$ is defined by a sparse tensor $A^m_{i_1,i_2,i_3}$ (11) which is symmetric with respect to lower indices i_1, i_2, i_3. Depending on the derivation, one could obtain other equivalent closed-form expressions for the tensor $A^m_{i_1,i_2,i_3}$.

4. TIME DISCRETISATION

For this initial study, we are using the Störmer-Verlet method [17] as it is an explicit and efficient numerical method, although does not guarantee stability. We choose a time step k in sec, yielding a sampling rate $f_\mathrm{s} = \frac{1}{k}$, and define a vector time series $\boldsymbol{q}^n = \boldsymbol{q}(nk),\ n = 0, \ldots, N-1$ on a uniform time grid.

To arrive at a one-step update, we rewrite (7) as a first-order system by introducing modal velocities $\boldsymbol{p} = [p_1(t), \ldots, p_M(t)]^\mathrm{T}$:

$$\begin{cases} \dot{\boldsymbol{q}} = \boldsymbol{p} \\ \dot{\boldsymbol{p}} = -2\mathbf{S}\boldsymbol{p} - \boldsymbol{\Omega}^2\boldsymbol{q} + \gamma^2 \boldsymbol{f}(\boldsymbol{q}) + \boldsymbol{\Phi}(x_\mathrm{e}) f_\mathrm{e}(t) \end{cases} \tag{12}$$

Using centred approximation for time derivatives, we get:

$$\begin{cases} \frac{\boldsymbol{q}^{n+1} - \boldsymbol{q}^n}{k} = \boldsymbol{p}^{n+\frac{1}{2}} \\ \frac{\boldsymbol{p}^{n+\frac{1}{2}} - \boldsymbol{p}^{n-\frac{1}{2}}}{k} = -2\mathbf{S}\boldsymbol{p}^n - \boldsymbol{\Omega}^2\boldsymbol{q}^n + \gamma^2 \boldsymbol{f}(\boldsymbol{q}^n) + \boldsymbol{\Phi}(x_\mathrm{e}) f_\mathrm{e}^n \end{cases}$$

where $\boldsymbol{p}^{n+\frac{1}{2}}$ is an interleaved sequence of modal velocities and $f_\mathrm{e}^n = f_\mathrm{e}(nk)$.

Following [17], this numerical scheme can be written in an explicit form to produce an update $(\boldsymbol{q}^n, \boldsymbol{p}^n) \to (\boldsymbol{q}^{n+1}, \boldsymbol{p}^{n+1})$:

$$\begin{cases} \boldsymbol{p}^{n+\frac{1}{2}} = \boldsymbol{p}^n + \frac{k}{2}\big[-2\mathbf{S}\boldsymbol{p}^n - \boldsymbol{\Omega}^2\boldsymbol{q}^n + \gamma^2 \boldsymbol{f}(\boldsymbol{q}^n) + \boldsymbol{\Phi}(x_\mathrm{e}) f_\mathrm{e}^n\big] \\ \boldsymbol{q}^{n+1} = \boldsymbol{q}^n + k\boldsymbol{p}^{n+\frac{1}{2}} \\ \boldsymbol{p}^{n+1} = \big(\mathbf{I} + k\mathbf{S}\big)^{-1}\big[\boldsymbol{p}^{n+\frac{1}{2}} + \\ \qquad + \frac{k}{2}\big(-\boldsymbol{\Omega}^2\boldsymbol{q}^{n+1} + \gamma^2 \boldsymbol{f}(\boldsymbol{q}^{n+1}) + \boldsymbol{\Phi}(x_\mathrm{e}) f_\mathrm{e}^{n+1}\big)\big] \end{cases} \tag{13}$$

Since matrix $\mathbf{I} + k\mathbf{S}$ is diagonal, its inverse can be easily computed. Using (6), we obtain an audio output as $w^n = \boldsymbol{\Phi}^\mathrm{T}(x_\mathrm{o})\boldsymbol{q}^n$.

5. LEARNING ALGORITHM

5.1. Neural Ordinary Differential Equations (NODEs)

NODEs can be defined through the following first-order system:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{g}_\theta(\boldsymbol{y}, t), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0. \tag{14}$$

Here $\boldsymbol{y} = \boldsymbol{y}(t)$: $\mathbb{R}^+ \to \mathbb{R}^K$ is an unknown function of time t, $\boldsymbol{y}_0 \in \mathbb{R}^K$ is an initial condition and $\boldsymbol{g}_\theta(\boldsymbol{y}, t)$: $\mathbb{R}^K \times \mathbb{R}^+ \to \mathbb{R}^K$ is a neural network where θ denotes the set of all learnable parameters and K denotes the state dimension. Generally, a simple neural architecture such as an MLP or convolutional network is chosen for $\boldsymbol{g}_\theta(\boldsymbol{y}, t)$. Chen et al. [5] have showed that the system (14) in combination with a numerical solver, labelled as ODE-Net, can be trained from data to reproduce dynamics of a target system.

Assume a target trajectory $\tau = \{\boldsymbol{y}_0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^{N-1}\}$ sampled on a uniform time grid with a step k. Given a predicted trajectory $\widetilde{\tau} = \{\boldsymbol{y}_0, \widetilde{\boldsymbol{y}}^1, \ldots, \widetilde{\boldsymbol{y}}^{N-1}\}$ by a numerical solution to the initial value problem (14), i.e., a forward pass of the ODE-Net, we can construct an objective function $\mathcal{J}(\theta)$ such as a mean squared error (MSE):

$$\mathcal{J}(\theta) = \frac{1}{KN}\sum_{n=0}^{N-1} \left\|\widetilde{\boldsymbol{y}}^n - \boldsymbol{y}^n\right\|_2^2, \tag{15}$$

where $\|\cdot\|_2$ is the Euclidean norm. We search for a local minimum of $\mathcal{J}(\theta)$ using gradient-based optimisation techniques where the gradient $\nabla_\theta \mathcal{J}$ can be computed using the backpropagation algorithm [18] through internal operations of a numerical solver or the adjoint sensitivity method [19, 5]. In most cases the objective function $\mathcal{J}(\theta)$ will be averaged for a finite set of target trajectories before each optimisation step.

5.2. Extension for Physical Modelling Synthesis

In the case of modal synthesis, there is a known ODE structure (12) which can serve an inductive bias for a NODEs framework [20]. In particular, we parametrise only a dimensionless memoryless nonlinear function $\boldsymbol{f}_\theta(\boldsymbol{q})$: $\mathbb{R}^M \to \mathbb{R}^M$ with a neural network, yielding a system of physically-informed NODEs:

$$\begin{bmatrix}\dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}}\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{I} \\ -\boldsymbol{\Omega}^2 & -2\mathbf{S}\end{bmatrix}\begin{bmatrix}\boldsymbol{q} \\ \boldsymbol{p}\end{bmatrix}}_{\text{Linear vibration}} + \gamma^2 \underbrace{\begin{bmatrix}\mathbf{0} \\ \boldsymbol{f}_\theta(\boldsymbol{q})\end{bmatrix}}_{\text{Neural network}} + \underbrace{\begin{bmatrix}\mathbf{0} \\ \boldsymbol{\Phi}(x_\mathrm{e})\end{bmatrix} f_\mathrm{e}(t)}_{\text{Excitation}} \tag{16}$$

As mentioned earlier, initial conditions for a state vector $\boldsymbol{y}^{\mathrm{T}} = [\boldsymbol{q}^{\mathrm{T}}, \boldsymbol{p}^{\mathrm{T}}]$ are assumed to be zero. To compute a forward pass of the physically-informed ODE-Net, we:

- set $\mathbf{S}, \boldsymbol{\Omega}, \gamma, \boldsymbol{\Phi}(x_{\mathrm{e}})$ in (16) using physical parameters of a target solution;
- precompute $f_{\mathrm{e}}^n = f_{\mathrm{e}}(nk), \;\; n = 0, \dots, N-1$ using (4) and excitation parameters $f_{\mathrm{amp}}, T_{\mathrm{e}}$ of a target solution;
- use the Störmer-Verlet method for (16) as in the case of a regular system (13) to produce a predicted trajectory $\widetilde{\tau}$.

The formulation (16) has strong implications. First, the exact expression for linear vibration exploits the periodic, harmonic and lossy nature of a musical system. Thus, we aid optimisation of the network by constraining the space of possible solutions and improve interpretability of the model.

Second, the neural network $\boldsymbol{f}_\theta(\boldsymbol{q})$ is memoryless and dimensionless, thus does not depend on physical parameters of a system and external excitation. Theoretically, these parameters can be changed after the training as long as range of modal displacements $\boldsymbol{q}$ stays the same as in a training dataset to simulate other configurations of a system.

Finally, we are able to use numerical methods developed directly for second-order systems (e.g., the Störmer-Verlet method) that are commonly used in musical acoustics. In the linear case, some such methods allow for exact discretisation of the harmonic oscillator equation [1].

6. EVALUATION

The physically-informed ODE-Net described in Section 5 has been implemented in the PyTorch framework [21] and evaluated for two case studies: nonlinear oscillator (Section 6.1) and nonlinear transverse string (Section 6.2). The training was conducted on cloud servers equipped with NVIDIA GeForce RTX 2080 Ti GPUs. The source code used for dataset generation and training is available in the accompanying GitHub repository[2].

For the parametrisation of $\boldsymbol{f}_\theta(\boldsymbol{q})$ we use an MLP with a linear output layer and a varying number of hidden layers of 100 units. Leaky rectified linear units (Leaky ReLUs) are used as activation functions in hidden layers and Kaiming initialisation [22] is used for the initial weights of the network. It was noted that saturating activation functions such as the hyperbolic tangent function might cause a vanishing gradient problem when modal displacements $\boldsymbol{q}$ span a wide range. Moreover, we are interested in reducing the computational cost of $\boldsymbol{f}_\theta(\boldsymbol{q})$ compared to the target nonlinear function $\boldsymbol{f}(\boldsymbol{q})$ (10), making rectified linear units a compelling choice. Compared to regular ReLUs, Leaky ReLUs provided faster convergence in optimisation in our experiments.

The training loss is the MSE taken over the whole state vector (15), i.e., including both modal displacements $\boldsymbol{q}$ and modal velocities $\boldsymbol{p}$. Backpropagation is performed using internal operations of the numerical solver (13), i.e., the "discretise-then-optimise" method, which is generally a preferred approach due to its gradient accuracy, speed and straightforward implementation [20]. The Adam optimiser [23] is used with default parameters. The dataset is divided into two subsets: 80% is used for training and 20% is used for validation. The training is performed for 5000 epochs. The resulting model is chosen based upon the lowest loss obtained on the validation subset.

For training we use a variation of the teacher forcing technique [24] by splitting up a target trajectory into 1 ms segments and providing true initial conditions for each segment to the ODE-Net. Since the numerical method (13) is given as a one step update we have access to both displacement and velocity of the target numerical solution at each time step, and thus initial conditions for each segment. In addition, the excitation function (4) is shifted in time to reflect a new starting point of integration. The main reason for using this technique is to speed up training as the number of integration steps in the numerical solver is significantly reduced. These integration steps can not be parallelised in time. Moreover, the likelihood of vanishing and exploding gradient problems during optimisation is also reduced by this technique as we avoid backpropagation on long time series [20].

6.1. Nonlinear Oscillator

Up to this point, discussion has been centred around the nonlinear transverse string model. To illustrate the generality of the proposed approach, consider a simple nonlinear oscillator of the following form:

$$\ddot{q} + \omega_0^2 q = \gamma^2 f(q) + f_{\mathrm{e}}(t), \tag{17}$$

where $\omega_0 = 400$ and $\gamma = 110$.

We generate two datasets consisting of 60 one-second trajectories at 44.1 kHz using exactly the same set of randomly-generated external excitations. The first dataset is created using the cubic nonlinearity $f(q) = -q^3$ and the other using the hyperbolic sine nonlinearity $f(q) = -\sinh(q)$. We use an MLP with two hidden layers of 100 units to parametrise $f_\theta(q)$ to ensure that the network has enough capacity to learn the underlying nonlinearities.

This lumped case is especially useful for analysis as we are able to easily visualise the learned nonlinear functions. As seen in fig. 1, the network is capable of reproducing both the cubic and the hyperbolic sine nonlinearities and can distinguish between them from data. Ranges of displacement for both cases correspond to minimum and maximum values in the datasets.

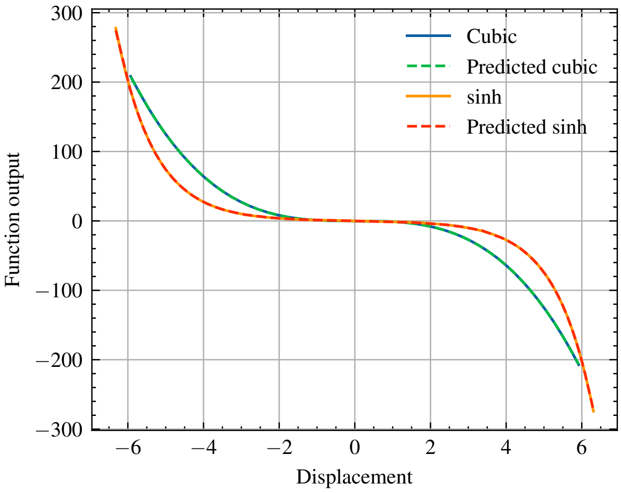

Figure 1: *Target and predicted nonlinear functions for the oscillator* (17).

[2] https://github.com/victorzheleznov/dafx25

6.2. Nonlinear Transverse String

For the case of nonlinear transverse string, which is described by the multi-dimensional system (7), we need to choose the number of modes we will use in simulations. In this work we have settled on 100 modes. Taking into account the effect of stiffness, this covers most of the audible range for the chosen string parameters — up to 17 kHz for the lowest considered fundamental frequency at around 60 Hz. Simulation of this wide range of frequencies requires oversampling by two for standard sampling rates such as 44.1 kHz and 48 kHz to avoid instability of the Störmer-Verlet method and aliasing due to the nonlinear effects for high amplitude excitations.

6.2.1. Datasets

Simulation parameters used for the generation of two datasets — one for training and validation and the other for testing — are outlined in table 1, where T_{sim} corresponds to the duration of the simulation. Each dataset consists of 60 string trajectories, which include both displacement and velocity information for each mode. For training and validation, only one set of physical parameters is chosen corresponding to a 61.72 Hz string. This string is excited by randomly-generated excitation functions (4) at randomised excitation positions x_{e} using a uniform distribution for the specified parameter ranges. For testing, string parameters γ and κ are also randomly generated to produce a range of fundamental frequencies from around 65 Hz to 123 Hz. The simulation duration is increased to account for a longer decay time due to a smaller damping parameter σ_0. In addition, the sampling rate is increased from 88.2 kHz to 96 kHz. Even though the ranges of the excitation parameters are the same for the two datasets, excitation functions are generated independently. For both datasets, audio output is drawn from randomised positions x_{o} along a string for each trajectory.

These parameters are motivated by two considerations. First, we want to test generalisation of the model to physical parameters, sampling rates and time scales not seen during training. In view of other machine learning approaches, this flexibility and controllability of the physically-informed ODE-Net can be considered as its main advantage. Second, strings with low fundamental frequencies are chosen so that the nonlinear effect is more prominent in simulations [1]. Since the network architecture is designed to learn the residual between the linear and nonlinear solutions, the training and validation dataset needs to reflect a significant difference between them.

6.2.2. Results

For training we use an MLP with five hidden layers of 100 units to parametrise $\boldsymbol{f}_\theta(\boldsymbol{q})$. Although no extensive research was conducted on the optimal size of the network with regards to simulation accuracy and computational cost, this structure provided the best empirical results when training MLPs with varying number of hidden layers. Further optimisation beyond 5000 epochs did not provide significant reduction in the validation loss, suggesting that the model has converged to a local minimum.

For evaluation, we are using the relative MSE for displacement trajectory $\boldsymbol{q}^n$ and audio output w^n:

$$\frac{\sum_n \|\widetilde{\boldsymbol{q}}^n - \boldsymbol{q}^n\|_2^2}{\sum_n \|\boldsymbol{q}^n\|_2^2}, \quad \frac{\sum_n |\widetilde{w}^n - w^n|^2}{\sum_n |w^n|^2},$$

Table 1: *Simulation parameters used for dataset generation.*

Parameter	Training and Validation	Test
f_{s}	88.2 kHz	96 kHz
T_{sim}	2 sec	3 sec
T_{e}	$[0.5, 1.5]$ ms	$[0.5, 1.5]$ ms
γ	123.4	$[130, 246]$
κ	1.01	$[1.01, 1.1]$
σ_0	3	2
σ_1	2×10^{-4}	2×10^{-4}
x_{e}	$[0.1, 0.9]$	$[0.1, 0.9]$
x_{o}	$[0.1, 0.9]$	$[0.1, 0.9]$
f_{amp}	$[2, 3] \times 10^4$	$[2, 3] \times 10^4$

where $\widetilde{\boldsymbol{q}}^n$ and $\widetilde{w}^n$ are predictions given by the physically-informed ODE-Net. These metrics are chosen as the audio output of simulation is directly dependent on the displacement trajectory of a string.

Metrics for the training and validation dataset and the test dataset are provided in table 2. Metrics are evaluated for the initial 100 ms and for the full duration of simulation, i.e., 2 or 3 sec, respectively. As can be seen, metrics for the test dataset remain on the same order of magnitude compared to the training and validation dataset, especially for the initial 100 ms. This suggests that the performance of the trained physically-informed ODE-Net does not significantly degrade for unseen simulation parameters.

Table 2: *Metrics for the nonlinear transverse string datasets.*

Metric	Training and Validation	Test
Evaluated for initial 100 ms		
Rel. MSE for displacement	3.91×10^{-3}	5.07×10^{-3}
Rel. MSE for output	3.66×10^{-3}	5.16×10^{-3}
Evaluated for full duration		
Rel. MSE for displacement	3.64×10^{-2}	6.68×10^{-2}
Rel. MSE for output	3.54×10^{-2}	7.00×10^{-2}

To illustrate a specific example, we select a trajectory from the test dataset with the largest relative MSE for audio output considering the full simulation duration. This corresponds to a 77.72 Hz string. Fig. 2 shows the displacement trajectory of the selected test string for the initial 100 ms. As can be seen, the predicted trajectory maintains the structure of target solution but starts to lag behind with time. This is to be expected, as any difference between the approximated and the underlying nonlinearity will be integrated over time by a numerical solver, thus accumulating the error. This is also confirmed by the metrics in table 2 which rise when evaluated for the full duration of simulation compared to the initial 100 ms. Examining the output waveform in fig. 3, we see that initially the predicted waveform closely follows the target solution, including high-frequency partials of higher modes. Moving forward in time, the predicted waveform still resembles the target solution much closer compared to the linear solution. Looking at displacements for individual modes in fig. 4, we also see that initially the predicted displacements follow the target solution. It should be noted that as vibration amplitude decreases over time due to loss in the system the nonlinear effects become less prominent [1], thus the initial response of the model to an external excitation is significantly more important for capturing the nonlinear behaviour.

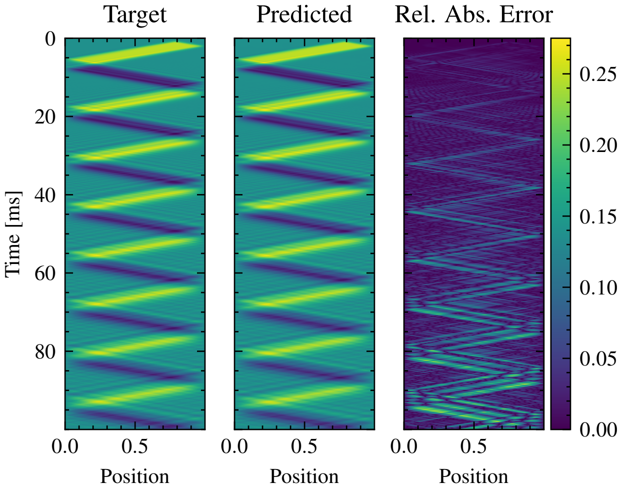

Figure 2: *Displacement trajectory for the selected test string. On the right, the relative absolute error between the target and predicted trajectories is shown, normalised by the maximum absolute value of target trajectory.*

We can examine the model further by comparing it to the linear solution. Fig. 5 shows MSE for the initial 100 ms of the linear and predicted trajectories compared to the target solution individually for each mode, evaluated over the whole test dataset. It can be clearly seen that the predicted trajectories capture displacements and velocities up to around 40th mode much more accurately compared to the linear solution. Since absolute values of MSE — the objective function of optimisation — are much lower for higher modes, the network seems to struggle to reproduce them. In a case by case examination of modes (e.g., the 40th mode in fig. 4), it was noted that the network is capable of accurately capturing a first few ms of the target solution. Afterwards, the main cause for the error of predicted trajectories becomes incorrectly estimated amplitude rather than instantaneous frequency. As can be seen on the output spectrogram for the selected test string (fig. 6), for which the 40th mode corresponds to around 4 kHz, the pitch glide effect is reproduced for higher modes in line with the target solution.

Considering the computational cost in this particular example, computation of $\boldsymbol{f}_\theta(\boldsymbol{q})$ requires far fewer floating point operations compared to the underlying nonlinearity $\boldsymbol{f}(\boldsymbol{q})$ (10). For 100 modes, the tensor $A^m_{i_1,i_2,i_3}$ (11) consists of $N_A = 2597200$ non-zero elements when accounted for the symmetries. Thus, it will require $\mathcal{O}(N_A)$ floating point operations to compute $\boldsymbol{f}(\boldsymbol{q})$. On the other hand, $\boldsymbol{f}_\theta(\boldsymbol{q})$ consists of five hidden layers of 100 units with Leaky ReLUs and a linear output layer. Computation of $\boldsymbol{f}_\theta(\boldsymbol{q})$ takes 121000 summations and multiplications combined when using a naive matrix multiplication algorithm. This suggests that the presented hybrid approach with a relatively small neural network might be able to sufficiently capture the behaviour of a complex physical model, while reducing the computational cost compared to the regular modal synthesis method. As of now, there were no formal listening tests conducted to assess a trade-off in perceptual accuracy, but readers are encouraged to listen to audio examples presented on the accompanying page[3].

7. CONCLUSIONS

A method for modelling distributed musical systems in a modal form using neural ordinary differential equations has been considered here. The proposed approach separates the problem into the linear and nonlinear parts, trying to combine complementary strengths of modal synthesis and machine learning. The analytical solution for linear vibration of modes allows the isolation of physical system parameters and external excitation from the neural network, which learns a dimensionless and memoryless nonlinear coupling between the modes. It has been shown that such structure can be used for simulation of a system with physical parameters unseen during training without significant degradation in accuracy of the resulting solution. In addition, the use of a neural network allows for an efficient representation of a complex nonlinear function for the non-trivial case study of nonlinear transverse string vibration.

This paper leaves many avenues and unanswered questions for further work, several of which will be explored in the near future. Currently, the model uses both displacement and velocity information of each mode for learning. However, as modal displacements and velocities are inherently connected in a system of ordinary differential equations, it should be possible for the model to learn using just the displacement information or one-dimensional audio output taken at randomised positions along the string. This would simplify a comparison of the proposed approach to other machine learning methods [8, 10, 11]. For such a comparison, external forcing terms should be omitted.

The Störmer-Verlet method [17] used for simulation here is an efficient numerical method that allows for an explicit update equation. Nevertheless, even for strings with low fundamental frequencies, it requires oversampling to run in a stable manner. An extension to the scalar auxiliary variable technique [25, 26] seems natural as it is the only numerical method known to the authors which is not only explicit but both conditionally stable and general, i.e., can be applied to different nonlinearities using the same discretisation process.

Further research is required into potential computational savings to be had with a neural network compared to the underlying nonlinearity, as real-time capability of the presented model remains unexplored. There should be a crossing point in terms of number of modes and complexity of the network where the network becomes more efficient than the analytical expression. With these computational savings, a trade-off in perceptual accuracy of the resulting sound should be studied in formal listening tests.

Finally, leaving the realm of computer simulation, the presented model has a potential for identification of nonlinear acoustic systems by learning from real-world data. Bridging the gap between the theory and what is observed in practice using such hybrid approaches has already been suggested in other fields [20].

[3] `https://victorzheleznov.github.io/dafx25`

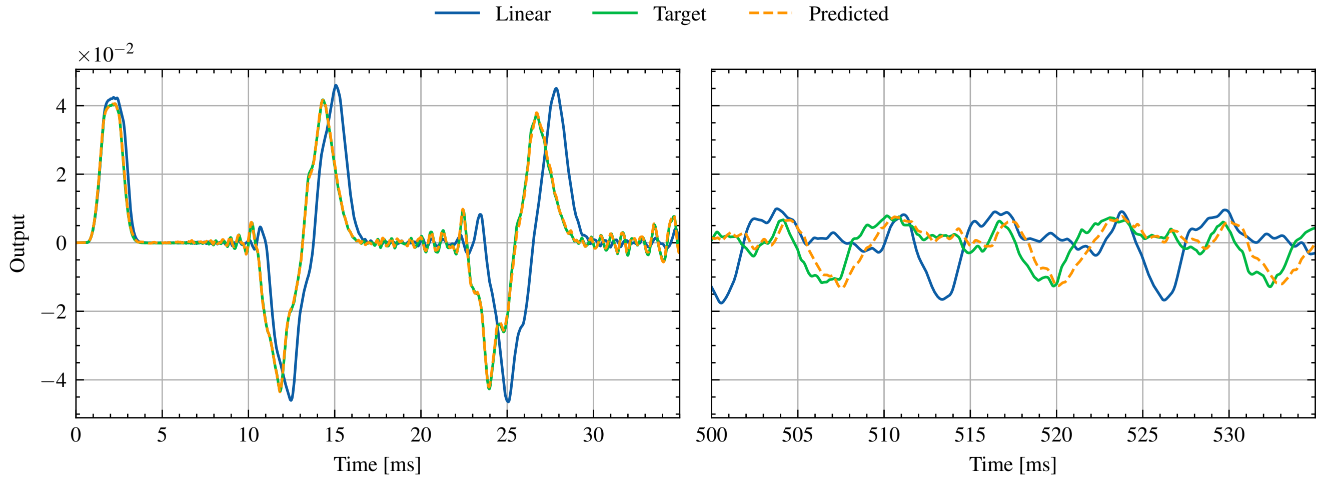

Figure 3: *Output for the selected test string at normalised position x_o equal to 0.87.*

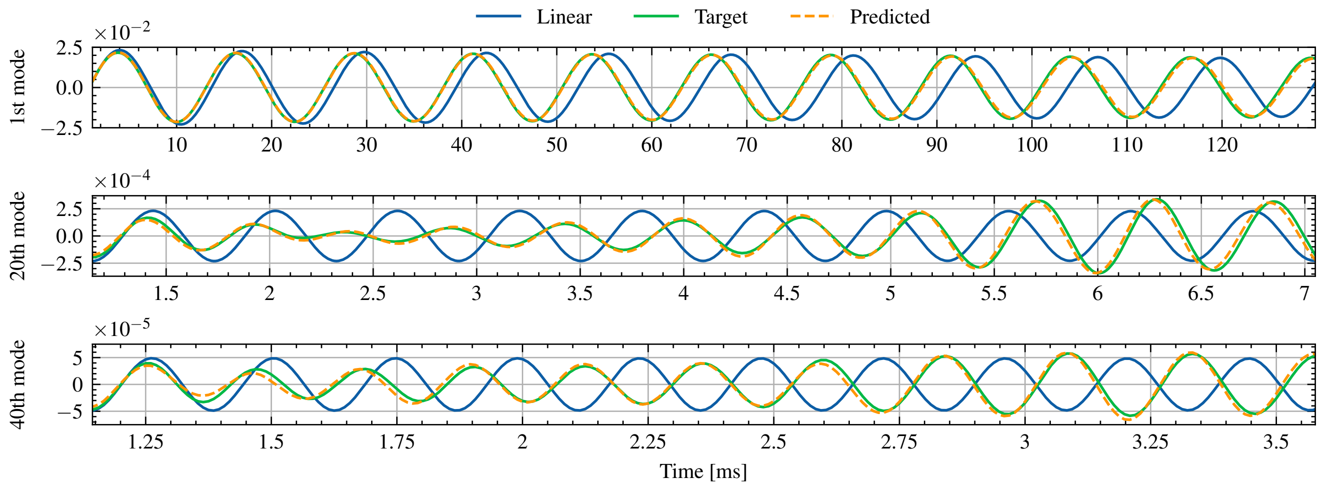

Figure 4: *Displacements of the 1st, 20th, 40th mode for the selected test string at normalised position x_o equal to 0.87 (initial 10 periods).*

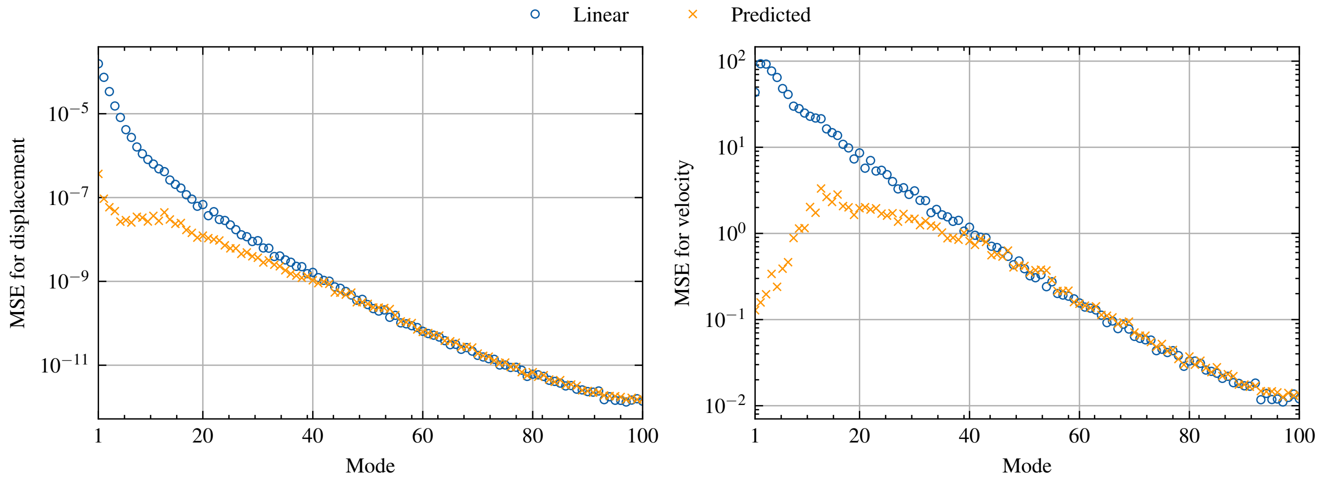

Figure 5: *MSE per mode for the initial 100 ms of the linear and predicted string trajectories for the test dataset.*

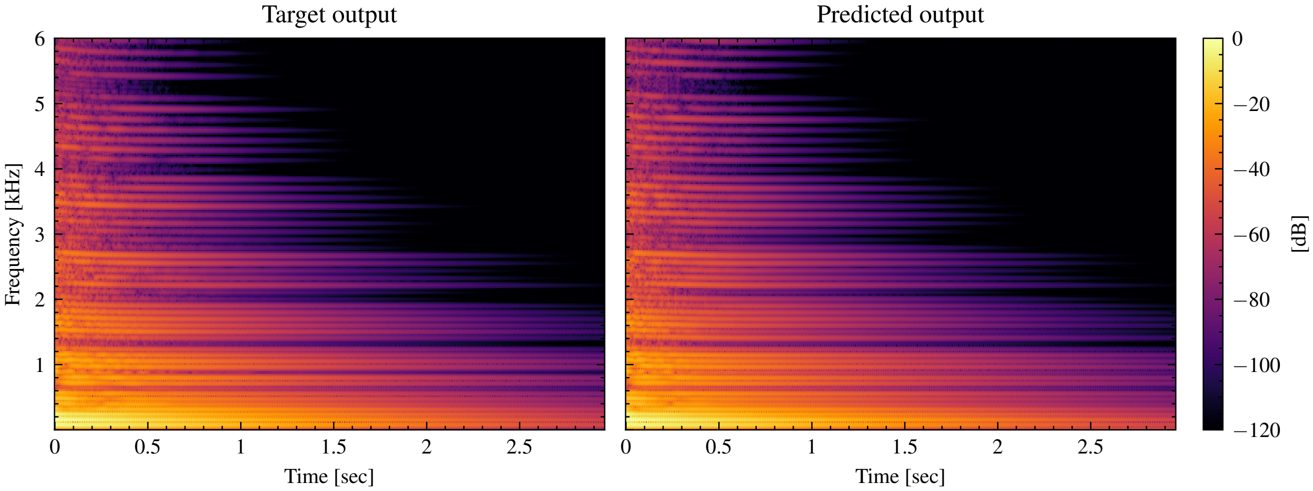

Figure 6: *Output spectrogram for the selected test string at normalised position x_o equal to 0.87.*

Considering musical acoustics, there are cases such as the bowed string where the underlying friction characteristic between the string and the bow is yet to be fully understood [27].

8. REFERENCES

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

[2] J. D. Morrison and J.-M. Adrien, "MOSAIC: A Framework for Modal Synthesis," *Comput. Music J.*, vol. 17, no. 1, pp. 45–56, 1993.

[3] A. Falaize and T. Hélie, "Passive simulation of the nonlinear port-Hamiltonian modeling of a Rhodes Piano," *J. Sound Vib.*, vol. 390, pp. 289–309, 2017.

[4] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. 22nd Int. Conf. Digital Audio Effects*, 2019.

[5] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," in *Adv. Neural Inf. Process. Syst.*, 2018, vol. 31.

[6] 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.

[7] J. Wilczek, A. Wright, V. Välimäki, and E. Habets, "Virtual Analog Modeling of Distortion Circuits Using Neural Ordinary Differential Equations," in *Proc. 25th Int. Conf. Digital Audio Effects*, 2022.

[8] J. D. Parker, S. J. Schlecht, R. Rabenstein, and M. Schäfer, "Physical Modeling using Recurrent Neural Networks with Fast Convolutional Layers," in *Proc. 25th Int. Conf. Digital Audio Effects*, 2022.

[9] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier Neural Operator for Parametric Partial Differential Equations," in *Proc. Int. Conf. Learn. Represent.*, 2021.

[10] 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 *Proc. 27th Int. Conf. Digital Audio Effects*, 2024.

[11] 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 *Adv. Neural Inf. Process. Syst.*, 2024, vol. 37, pp. 1058–1081.

[12] P. M. Morse and K. U. Ingard, *Theoretical acoustics*, McGraw-Hill, 1968.

[13] S. Bilbao, "Conservative numerical methods for nonlinear strings," *J. Acoust. Soc. Am.*, vol. 118, no. 5, pp. 3316–3327, 2005.

[14] G. Kirchhoff, *Vorlesungen über Mechanik*, Tauber, 1883.

[15] G. F. Carrier, "On the nonlinear vibration problem of the elastic string," *Q. Appl. Math.*, vol. 3, pp. 157–165, 1945.

[16] S. Bilbao, M. Ducceschi, and C. Webb, "Large-scale real-time modular physical modeling sound synthesis," in *Proc. of the 22nd Int. Conf. Digital Audio Effects*, 2019.

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

[18] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[19] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes*, Interscience Publishers John Wiley & Sons, Inc., 1962.

[20] P. Kidger, *On Neural Differential Equations*, Ph.D. thesis, University of Oxford, 2022.

[21] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Adv. Neural Inf. Process. Syst.*, 2019, vol. 32.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[23] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017, arXiv:1412.6980.

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

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

[26] R. Russo, S. Bilbao, and M. Ducceschi, "Scalar auxiliary variable techniques for nonlinear transverse string vibration," *IFAC-PapersOnLine*, vol. 58, no. 6, pp. 160–165, 2024.

[27] P. Galluzzo, J. Woodhouse, and H. Mansour, "Assessing Friction Laws for Simulating Bowed-String Motion," *Acta Acust. united Ac.*, vol. 103, pp. 1080–1099, 2017.