PHYSICS-INFORMED DEEP LEARNING FOR NONLINEAR FRICTION MODEL OF BOW-STRING INTERACTION

Xinmeng Luan and Gary Scavone

CAML[†] , CIRMMT [‡] ,
xinmeng.luan@mail.mcgill.ca, gary.scavone@mcgill.ca

ABSTRACT

This study investigates the use of an unsupervised, physics-informed deep learning framework to model a one-degree-of-freedom mass-spring system subjected to a nonlinear friction bow force and governed by a set of ordinary differential equations. Specifically, it examines the application of Physics-Informed Neural Networks (PINNs) and Physics-Informed Deep Operator Networks (PI-DeepONets). Our findings demonstrate that PINNs successfully address the problem across different bow force scenarios, while PI-DeepONets perform well under low bow forces but encounter difficulties at higher forces. Additionally, we analyze the Hessian eigenvalue density and visualize the loss landscape. Overall, the presence of large Hessian eigenvalues and sharp minima indicates highly ill-conditioned optimization.

These results underscore the promise of physics-informed deep learning for nonlinear modelling in musical acoustics, while also revealing the limitations of relying solely on physics-based approaches to capture complex nonlinearities. We demonstrate that PI-DeepONets, with their ability to generalize across varying parameters, are well-suited for sound synthesis. Furthermore, we demonstrate that the limitations of PI-DeepONets under higher forces can be mitigated by integrating observation data within a hybrid supervised-unsupervised framework. This suggests that a hybrid supervised-unsupervised DeepONets framework could be a promising direction for future practical applications.

1. INTRODUCTION

In recent years, Physics-Informed Neural Networks (PINNs) [1], a prominent framework in scientific machine learning (SciML), has gained significant traction in computational physics. The core idea of PINNs is to approximate the solution of ordinary differential equations (ODEs) or partial differential equations (PDEs) using a neural network, where the inputs are typically the space and time coordinates. By utilizing automatic differentiation in neural networks, we can efficiently compute the required gradients in the PDE residuals, along with the losses associated with initial and boundary conditions (ICs, BCs). This results in a composite loss function with multiple competing objectives, effectively framing the problem of solving ODEs/PDEs as a multi-task deep learning challenge. Another framework, Physics-Informed Deep Operator Networks (PI-DeepONets) was proposed to incorporate ICs as part of the input [2, 3, 4]. This approach is effective in simulating long time-integration systems, providing an advantage over PINNs, which typically require the time-marching scheme [5] or the causal training strategy [6].

Physical modelling and physics-based sound synthesis of musical instruments require solving the system's underlying governing equations [7, 8]. SciML has been utilized in musical instrument measurement methodologies, such as near-field acoustic holography, to reconstruct violin plate vibration patterns [9, 10, 11]. A PINNs approach for acoustic tube modeling was proposed in [12], which can be applied to model resonators of wind instruments [13] or the vocal tract [14], and reconstructing the acoustic field within a tube using sparsely measured pressure [15], identifying physical coefficients in tubes [16], and determining geometric parameters of a trumpet [13].

Beyond physics-informed deep learning, some studies have explored data-driven, supervised deep learning approaches for solving the governing equations in musical instrument modeling. For example, the Fourier Neural Operator has been applied to model stiff membrane vibrations [17]. Recurrent neural networks, state space models and Koopman-based deep learning techniques have been used to simulate dispersive linear lossy and nonlinear tension modulated strings [18, 19].

However, these approaches treat the problem purely as a supervised learning task, disregarding the underlying physical knowledge encoded in the governing equations. Physics-informed approaches are inherently more challenging than purely data-driven methods due to their unsupervised nature. As shown in [20], incorporating training data into the physics-informed framework to create a hybrid physics-informed data-driven approach can simplify the loss landscape and mitigate ill-conditioned optimization.

In this paper, we address the bowing simulation via PINNs and PI-DeepONets. The distinctive sound of bowed string instruments arises from the bowing mechanism, which involves continuous excitation resulting from bow / string interactions and nonlinear friction forces. When simulating bowed strings, the primary computational challenge lies in handling this nonlinearity. This challenge applies not only to traditional numerical methods, such as finite difference and finite element methods, but also to physics-informed deep learning approaches like PINNs and PI-DeepONets, as we will demonstrate in this paper. Therefore, we focus exclusively on the bowing mechanism by solving a bowed one-degree-of-freedom mass-spring system to evaluate the effectiveness of PINNs and PI-DeepONets in such simulations. To the best of our knowledge, no existing research has explored modelling the bow-string friction interaction using deep learning, whether through data-driven or physics-informed methods. Scenarios with different bow forces have been investigated using both PINNs and PI-DeepONets. The results show PINNs successfully address the problem across different bow force scenarios, while

[†] Computational Acoustic Modeling Laboratory, McGill University
[‡] Center for Interdisciplinary Research in Music Media and Technology

PI-DeepONets perform well under low bow forces but encounter difficulties at higher forces. Additionally, we analyze the Hessian eigenvalue density and visualize the loss landscape. The presence of large Hessian eigenvalues and sharp minima indicates highly ill-conditioned optimization. We also demonstrate that the limitations of PI-DeepONets under higher forces can be mitigated by integrating observation data within a hybrid supervised-unsupervised framework.

The remainder of this paper is organized as follows. Section 2 describes the bowed mass-spring model. Section 3 describes PINNs and PI-DeepONets frameworks. Section 4 presents the results and discussion. Finally, Section 5 summarizes the study and outlines future directions.

2. BOWED MASS-SPRING MODEL

Instead of modeling the full string dynamics, we consider a bowed simple harmonic oscillator, represented by a mass-spring system with nonlinear frictional forcing. This archetypal test model is widely used in research to study numerical simulation challenges [7, 21]. The schematic illustration is shown in Fig. 1. The motion of the mass m is described by

$$\begin{cases} u_{tt} + \omega^2 u = -F_B \phi(\eta), \\ \eta = u_t - v_B, \end{cases} \tag{1}$$

where u is the mass displacement (m), ω is the angular frequency ($\mathrm{rad\,s^{-1}}$), F_B is the bow force normalized by the object's mass ($\mathrm{m\,s^{-2}}$), v_B is the bow velocity ($\mathrm{m\,s^{-1}}$), η is the relative velocity between the mass and the bow ($\mathrm{m\,s^{-1}}$), and the function $\phi(\eta)$ represents the bow friction characteristic. In this case, we

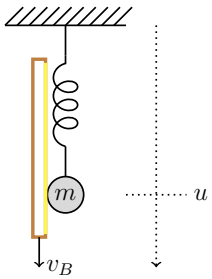

Figure 1: Illustration of a bowed mass-spring system.

consider the soft characteristic static friction model [7]

$$\phi(\eta) = \sqrt{2a}\eta e^{-a\eta^2+1/2}, \tag{2}$$

where a is a free parameter. See Fig. 2 for a plot of $\phi(\eta)$ with $a = 100$. The derivative $\dfrac{d\phi}{d\eta}$ is also plotted, and the highly nonlinear region is defined as the interval between its two local minima. This region corresponds to the stick phase, characterized by low values of η, while the regions outside this interval are associated with the slip phase. Note that this friction model is not derived from physical principles, but it provides a reasonable approximation of the discontinuity and is relatively easier to handle numerically [7]. Seeking a more accurate and physically realistic bow friction model remains an active research topic in the musical acoustics community; see [22] for a comparison of different models. However, since our primary goal is to evaluate novel computational approaches, we adopt this classical static bowed mass-spring

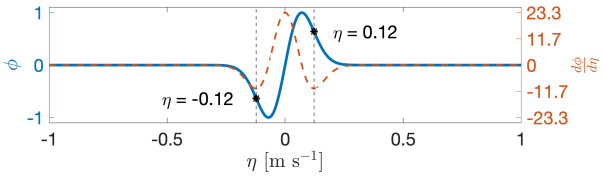

Figure 2: Soft characteristic static friction model ϕ and its derivative $\dfrac{d\phi}{d\eta}$, given by (2), with $a = 100$.

model, which is commonly employed as a benchmark in physics-informed sound synthesis for testing numerical methods [7, 21]. Indeed, it would be worthwhile to further explore the application of the elasto-plastic friction model [22], which provides a more physically accurate representation.

This one-degree-of-freedom mass-spring system is governed by two coupled ODEs (1), which can be reformulated into first-order form [21] by introducing the generalized coordinate q and generalized momentum p, as

$$q = \omega u, \quad p = u_t. \tag{3}$$

Therefore, (1) becomes

$$\begin{cases} q_t - \omega p = 0, \\ p_t + \omega q + F_B \phi(\eta) = 0. \end{cases} \tag{4}$$

In the following, we use this first-order form (4) for the simulation. To solve (4), we require two initial conditions (ICs) for both $p|_{t=0}, q|_{t=0}$.

3. NEURAL NETWORKS

We present the solution of the bowed mass-spring system by formulating it as an optimization problem, which is then solved using the PINNs and PI-DeepONets frameworks.

3.1. Modified FCNN

A modified fully-connected neural network (FCNN) [23], inspired by attention mechanisms, is used in this study. It outperforms standard FCNNs in PINNs by capturing multiplicative interactions between input dimensions and including residual connections [23]. Given the network inputs X and outputs O, the forward pass is defined by propagating X through the network layers to compute O, through [23]

$$\begin{aligned} &U = \sigma(XW^U + b^U), \quad V = \sigma(XW^V + b^V), \\ &H^{(1)} = \sigma(XW^{Z,1} + b^{Z,1}), \\ &Z^{(k)} = \sigma(H^{(k)}W^{Z,k} + b^{Z,k}), \quad k = 1, ..., L, \\ &H^{(k+1)} = (1 - Z^{(k)}) \odot U + Z^{(k)} \odot V, \quad k = 1, ..., L, \\ &O = H^{(L+1)}W^O + b^O, \end{aligned} \tag{5}$$

where σ is the activation function and $\odot$ denotes element-wise multiplication. The network parameters are

$$\theta = \{W^U, b^U, W^V, b^V, (W^{Z,k}, b^{Z,k})_{k=1}^L, W^O, b^O\}, \tag{6}$$

where $W^{(\cdot)}$ and $b^{(\cdot)}$ denote the weight matrices and bias vectors of the corresponding layer. The channel or layer sizes of $U, V, Z^{(k)}$ and O are denoted as $c_U, c_V, c_{Z^{(k)}}$ and c_O, respectively. Z has L

layers in total. A graphical representation of this modified FCNN is shown in Fig. 3 (c).

3.2. PINNs

PINNs are formulated as

$$\begin{cases} p = \mathcal{F}_{1,\theta_1}(t), \\ q = \mathcal{F}_{2,\theta_2}(t), \end{cases} \tag{7}$$

where $\mathcal{F}_{1,\theta_1}$ and $\mathcal{F}_{2,\theta_2}$ represent two distinct networks with identical architecture. Since $\mathcal{F}_{1,\theta_1}$ and $\mathcal{F}_{2,\theta_2}$ follow the same flow, we illustrate the process using one as an example. The input to $\mathcal{F}_{1,\theta_1}$ is the time coordinate $t \in [0, t_N] \subset \mathbb{R}^{1\times N}$, a single channel vector sampled at N points, which is first scaled by scaling factor s^t for normalization. Then the Random Fourier Feature (RFF) embedding [24] is employed with a scale parameter σ', which controls the range of frequencies in the embedding, and is applied with an encoding size of c_{RFF}. RFF maps input data into a higher-dimensional space using sinusoidal transformations, helping to mitigate *spectral bias* phenomenon—the tendency of neural networks to learn low-frequency components faster while struggling with high-frequency ones [25]. RFF has proven effective in overcoming spectral bias in PINNs [26]. The embedded features next serve as the input to the modified FCNN. The output of the FCNN is $p \subset \mathbb{R}^{1\times N}$, which are scaled by scaling factor s^p (s^q for the other case). Automatic differentiation is then used to compute the loss functions, which encompass the PDE and IC losses, all formulated as mean squared error (MSE) terms. Predicted values are denoted with a hat $\hat{\cdot}$. The PDE losses are expressed as

$$\begin{cases} \mathcal{L}_{ODE_1} = \dfrac{1}{N_{ODE}} \left\| \hat{q}_t - \omega \hat{p} \right\|, \\ \mathcal{L}_{ODE_2} = \dfrac{1}{N_{ODE}} \left\| \hat{p}_t + \omega \hat{q} + F_B \sqrt{2a} \eta e^{-a\eta^2 + 1/2} \right\|, \\ \qquad\qquad\qquad\qquad t \in [0, t_N]. \end{cases} \tag{8}$$

The IC losses are

$$\begin{cases} \mathcal{L}_{IC_1} = \dfrac{1}{N_{IC}} \left\| \hat{p}|_{t=0} - p|_{t=0} \right\|, \\ \mathcal{L}_{IC_2} = \dfrac{1}{N_{IC}} \left\| \hat{q}|_{t=0} - q|_{t=0} \right\|. \end{cases} \tag{9}$$

N_{ODE} and N_{IC} represent the respective numbers of collocation points used for the ODE and IC loss computations. Then the total loss function is

$$\begin{aligned} \mathcal{L} = \lambda_{ODE_1} \mathcal{L}_{ODE_1} + \lambda_{ODE_2} \mathcal{L}_{ODE_2} + \\ \lambda_{IC_1} \mathcal{L}_{IC_1} + \lambda_{IC_2} \mathcal{L}_{IC_2}, \end{aligned} \tag{10}$$

with λ_{ODE_1}, λ_{ODE_2}, λ_{IC_1} and λ_{IC_2} as the loss function weights. We employ learning rate annealing for loss balancing to determine these weights, as described in [23]. The total loss function is then fed into the optimizer, where a gradient descent routine is applied via back propagation to update θ_1 and θ_2. A diagram for the architecture of PINNs is shown in Fig. 3 (a).

However, PINNs often encounter failure mode challenges when the optimization problem is ill-conditioned [5]. Various approaches have been proposed to address these issues, and a comprehensive review of PINNs training strategies can be found in [27]. In long-time integration problems (always the case for sound synthesis or musical acoustic simulation), it is extremely difficult to solve the entire time domain simultaneously. Additionally, studies have shown that continuous-time PINNs models can violate causality, making them prone to converging toward incorrect solutions [6]. To overcome the time related issues, we utilize the *time-marching* scheme [5] and *causal training* strategy [6]. For the time-marching scheme, the entire time domain is segmented into M_{tm} subdomains (or windows), as

$$\begin{aligned} t^{(i)} \in [t_i, t_{i+1}] \subset \mathbb{R}^{1\times N_i}, \text{for } i = 0, ..., M_{tm} - 1, \\ \text{where } \sum_{i=0}^{M_{tm}-1} N_i = N, t_0 = 0, t_{M_{tm}} = t_N. \end{aligned} \tag{11}$$

In other words, a separate network is utilized and trained after the optimization of the previous subdomain is solved, following a time-marching routine and ultimately resulting in a total of M_{tm} sub-networks. The schematic of time-marching for PINNs is presented in Fig. 4. For a more challenging task, we adopt the causal training strategy, for which each subdomain $t^{(i)}$ is divided into M_{cas} chunks

$$\begin{aligned} t^{(j)} \in [t_j, t_{j+1}] \subset \mathbb{R}^{1\times N_j}, \text{for } j = 0, ..., M_{cau} - 1, \\ \text{where } \sum_{j=0}^{M_{cau}-1} N_j = N_i, t_0 = t_i, t_{M_{cau}} = t_{i+1}. \end{aligned} \tag{12}$$

The training of the i-th sub-network starts with the time samples from $t^{(j=1)}$. Once the loss condition $\mathcal{L}_{ODE_1} < \eta_{cau}$ is satisfied, the time samples are augmented by adding samples from $t^{(j=2)}$, continuing this process iteratively until the last chunk is included. Note that the data augmentation procedure occurs within a single network training process, making it fundamentally different from time-marching. A schematic view of time-marching and causal training is provided in Fig. 4.

3.3. PI-DeepONets

PI-DeepONets [3] are formulated as

$$(p, q) = \mathcal{F}_{3,\theta_3}(t, (p, q)|_{t=0}). \tag{13}$$

The inputs to $\mathcal{F}_{3,\theta_3}$ are $t \in [0, t_M] \subset \mathbb{R}^{1\times M}$, similar to PINNs, scaled by s_i^t and $(p, q)|_{t=0} \in [p_{min}, p_{max}] \times [q_{min}, q_{max}] \subset \mathbb{R}^{2\times M}$, scaled by $s^{p,q}$. Then, the RFF embedding is applied to both t and $(p, q)|_{t=0}$, which are then fed into the branch and trunk net, respectively. Both networks are modified FCNNs. The latent features from both the branch and trunk network outputs are split into two equal parts, with each half merged via a dot product to separately produce the outputs $p \subset \mathbb{R}^{1\times M}$ and $q \subset \mathbb{R}^{1\times M}$, which are scaled by $s^{p,q}$. Then, the backpropagation process follows a similar routine as PINNs. A diagram for the architecture of PI-DeepONets is shown in Fig. 3 (b).

3.4. PINNs vs. PI-DeepONets

It is natural to ask about the differences between PINNs and PI-DeepONets. From the input perspective, PINNs use only the coordinate t as input. In contrast, PI-DeepONets take t as input to the branch network and also incorporate the initial condition $(p, q)|_{t=0}$ as input to the trunk network. From the solution per-

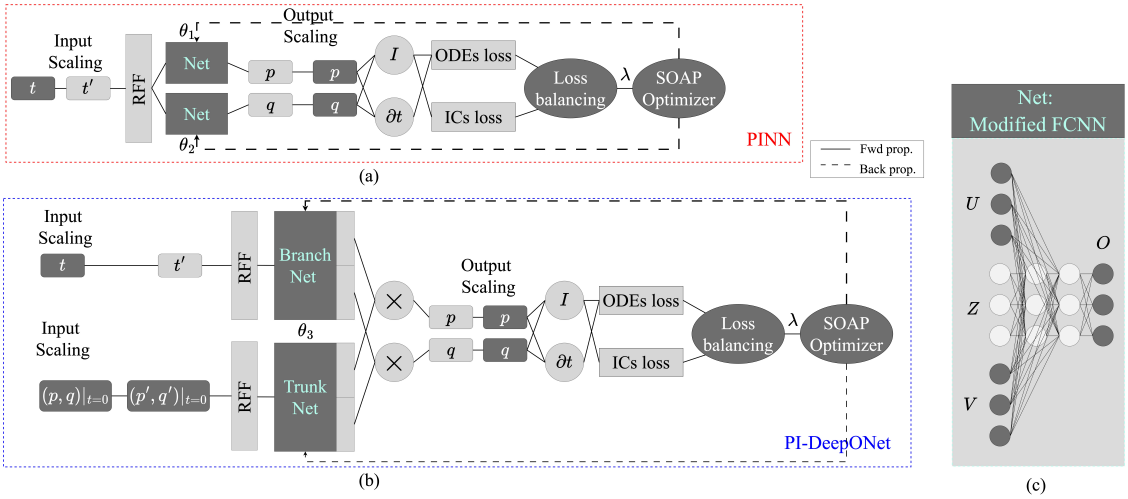

Figure 3: Network Architectures: (a) PINNs, (b) PI-DeepONets, and (c) Modified FCNN [23]. Note that (c) serves as a component of the PINNs' Nets, as well as the Branch and Trunk Nets in PI-DeepONets. The legend indicating forward propagation (fwd prop.) and backpropagation (back prop.) applies to both (a) and (b).

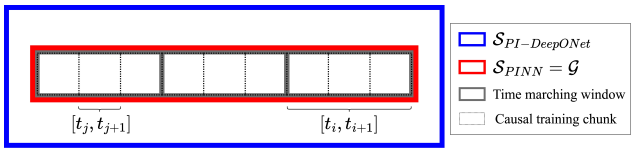

Figure 4: The solution spaces that PINNs and PI-DeepONets aim to solve.

spective, the ODEs themselves (4) define an infinite domain $\mathcal{S}$ (with $t \in [0, \infty]$), but the solution space of interest—referred to as the goal space $\mathcal{G}$—is a subspace of this infinite domain, with $\mathcal{G} \subset \mathcal{S}$. In the case of PINNs, solving a long-time integration problem requires employing multiple (M_{tm}) neural networks over different time windows to cover $\mathcal{G}$. Therefore, under the ideal assumption that PINNs can perfectly solve the ODEs, we have

$$\mathcal{S}_{PINN} = \bigcup_{i=0}^{M_{tm}-1} \mathcal{S}^{(i)}_{PINN}, \quad \mathcal{G} = \mathcal{S}_{PINN}. \tag{14}$$

However, PI-DeepONets, by randomly sampling ICs $p(0)$ and $q(0)$, inherently capture $\mathcal{G}$ within their solution space $\mathcal{S}_{PI-DeepONet}$. Under the ideal assumption that PI-DeepONets can perfectly solve the ODEs, we obtain

$$\mathcal{G} \subset \mathcal{S}_{PI-DeepONet}. \tag{15}$$

Essentially, the solutions corresponding to these randomly sampled ICs do not necessarily lie within $\mathcal{G}$. In other words, PI-DeepONets operate over a significantly larger solution space compared to PINNs. Ideally, if both methods achieve equally good results, PI-DeepONets might be the preferred choice. However, given the broader solution space, the complexity of PI-DeepONets is inherently higher than that of PINNs. A visualization of the solution space is provided in Fig. 4.

4. SIMULATION RESULTS

4.1. Implementation

We validate the PINNs and PI-DeepONets for $\omega = 2\pi f, f = 100\,\text{Hz}$, $a = 100$, and $v_B = 0.2\,\text{m}\,\text{s}^{-1}$, considering three different values of F_B set as 10, 100, and 1000. The choice of varying F_B is motivated by its influence on the bowing mechanism, resulting in different waveforms. For both PINNs and PI-DeepONets, the hyperbolic tangent (tanh) activation functions are utilized, the RFF encoding size, c_{RFF}, is set to 50 and the resulting layer U and V have $c_U = c_V = 100$ channels. For PINNs, each modified FCNN consists of $L = 4$ Z layers, with each layer containing $c_{Z^{(k)}} = 100$ channels. The output layer O has $c_O = 1$ channel. For PI-DeepONets, each modified FCNN in both the branch and trunk net consists of $L = 6$ Z layers, with each layer containing $c_{Z^{(k)}} = 100$ channels. The output layer O has $c_O = 200$ channels.

For PINNs, $N_{ODE_1} = N_{ODE_2} = 1000$, $N_{IC_1} = N_{IC_2} = 1$. The ODEs input t are uniformly sampled within the interval $[0, s^t]$. The training of PINNs follows a full-batch paradigm. For PI-DeepONets, $N_{ODE_1} = N_{ODE_2} = 10000 \times 1000$, $N_{IC_1} = N_{IC_2} = 10000$. To construct the dataset, we first generate a single group of data. Each group starts with an IC input (p, q) randomly sampled from the range $[-s^{p,q}, s^{p,q}]$ at $t = 0$ (we set $s^p = s^q = s^{p,q}$ for all cases). The corresponding ODEs inputs (p, q) are then repeated 1000 times while 1000 values of t are randomly sampled from $[0, s^t]$. This process is repeated 10000 times to create the final input dataset. PI-DeepOnets employ a mini-batch training strategy with a batch size of 50000. We employ the state-of-the-art second order optimizer ShampoO with Adam in the Preconditioner's eigenbasis (SOAP) [28] for both PINNs and PI-DeepONets. SOAP has been demonstrated to efficiently approximate the Hessian preconditioner, leading to significant performance improvements in PINNs [29]. Note that either full-batch training or a large batch size is used, as it is suggested in [28] that a large batch size enhances the performance of the SOAP optimizer. The other hyperparameters for SOAP remain at their default values. The initial learning rate is set to 0.003. An exponential learning rate scheduler is applied with a decay rate of 0.9. The decay step is set to 10000 for PINNs and 3000 for PI-DeepONets. A loss weight balance strategy, learning rate annealing (ann.) [23], may be employed for some cases. If the adaptive loss weight annealing strategy is not applied, weights are manually set as $\lambda_{ODE_1} = \lambda_{ODE_2} = 10, \lambda_{IC_1} = \lambda_{IC_2} = 1 \times 10^6$. The training strategies and other hyperparameters are summarized in Table 1. It is worth mentioning that different hyperparameters may be used for different values of F_B, depending on the

Table 1: Hyperparameters and training strategies for PINNs and PI-DeepONets.

Net	F_B	s^t	$s^{p,q}$	cas.	M_{cau}	η_{cau}	tm.	M_{tm}	RFF	σ'	ann.	runtime [h]
PINNs	10	0.1	0.2	×	-	-	✓	3	✓	1	×	38.76
PINNs	100	0.03	0.2	×	-	-	✓	3	✓	1	×	38.14
PINNs	1000	0.01	1	✓	50	0.1	✓	5	✓	3	✓	14.59
PI-DeepONets	10	0.01	0.35	×	-	-	×	-	✓	1	✓	36.62
PI-DeepONets	100	0.01	0.35	×	-	-	×	-	✓	1	✓	21.41
PI-DeepONets	1000	0.01	2	×	-	-	×	-	✓	3	✓	14.22

complexity of the optimization task. As discussed in Section 3.2, incorporating the ODEs into the loss function can lead to an ill-conditioned optimization problem, which may cause the network to encounter failure modes [5]. Intuitively, more difficult tasks demand a more careful design, potentially incorporating additional strategies, more time-marching windows, and more chunks for causal training, when such techniques are used. In other words, these hyperparameters are sensitive to F_B. The training processes are terminated once the monitored loss converges. Notice that to successfully train PINNs for $F_B = 1000$, the causal training process may terminate early at an intermediate chunk rather than progressing to the final chunk if incorporating data from the next chunk introduces a bias that compromises the accuracy of the entire window. The implementation is carried out using PyTorch and an NVIDIA GeForce RTX 4080 GPU with 16 GB VRAM. The code and accompanying sound samples are available on github[1].

4.2. Results

4.2.1. Zero ICs: PINNs vs. PI-DeepONets

The first and second rows of Fig. 5 present the simulation results for zero ICs ($p|_{t=0} = q|_{t=0} = 0$), where we compare PINNs and PI-DeepONets against the Finite Difference Method (FDM) benchmark. The FDM scheme for the bowed mass-spring model follows the implementation in [7] and employs an extremely high sampling rate of 4410 kHz to ensure accuracy. Overall, both PINNs and PI-DeepONets perform well for $F_B = 10$ and $F_B = 100$. However, for $F_B = 1000$, the optimization of PI-DeepONets fails, whereas PINNs successfully converge to the desired solution. When $F_B = 1000$, training the model becomes more challenging, often leading to failure modes in the bowed mass-spring model. To mitigate this issue, we employ causal training and a time-marching strategy with a short time duration. Notably, this is the only model that utilizes causal training.

To further assess the accuracy of the results, we present the ODE_1 and ODE_2 losses in the third and fourth rows of Fig. 5. In particular, we compare the losses of PINNs and PI-DeepONets against FDM. Here, FDM refers to the previously mentioned high sampling rate, while FDM-low represents a lower sampling rate (audio rate) of 44.1 kHz. PINNs demonstrate better accuracy than PI-DeepONets and can achieve competitive accuracy with FDM at the high sampling rate. Meanwhile, PI-DeepONets can achieve competitive accuracy with FDM at an audio sampling rate. When examining PINNs for $F_B = 1000$, it is worth noting that even FDM at an audio sampling rate can exhibit relatively large local numerical errors, although the results may still be acceptable for sound synthesis purposes. This suggests that the challenge is not exclusive to PINNs and PI-DeepONets but also affects traditional numerical methods.

The results above intuitively suggest that for $F_B = 1000$, the optimization process is ill-conditioned. To further validate this hypothesis, we analyze the deep learning dynamics using both the Hessian eigenvalue density histogram and loss landscape visualization (for PINNs, we only show the result from the first window network for time-marching). The Hessian matrix, denoted as $H = \nabla^2 \mathcal{L}(\theta)_{i,j} = \dfrac{\partial^2}{\partial_{\theta_i} \partial_{\theta_j}} \mathcal{L}(\theta), H \in \mathbb{R}^{n_\theta \times n_\theta}$, consists of the second-order partial derivatives of the loss function with respect to the neural network parameters, where n_θ is the total number of parameters. It captures the information about the curvature of the loss landscape and can be computed using automatic differentiation. The Hessian matrix can be decomposed as $H = Q \Lambda Q^T$, where Q contains the eigenvectors and $\Lambda = diag(\lambda_1, ..., \lambda_{n_\theta})$ is a diagonal matrix of the corresponding eigenvalues.

The Hessian eigenvalue density histograms are shown in the fifth row of Fig. 5. Pyhessian [30] is utilized for the calculation of Hessian matrix. When analyzing PINNs and PI-DeepONets separately, we observe that as F_B increases, the maximum eigenvalue also increases. Furthermore, when comparing PINNs to PI-DeepONets, PI-DeepONets generally exhibit higher maximum eigenvalues. Overall, the observed maximum eigenvalues are quite large, indicating a high condition number (ratio of maximum to minimum eigenvalues), which suggests that the optimization problem is ill-conditioned. Moreover, as F_B increases, the level of ill-conditioning also intensifies.

We visualize the loss landscape of PINNs and DeepONets in Fig. 5, shown in the sixth and seventh rows, respectively. The network parameters are perturbed along two directions $\varepsilon_1, \varepsilon_2$: the eigenvectors corresponding to the top two Hessian eigenvalues, resulting in the network parameters as $\theta^* = \theta + \alpha\varepsilon_1 + \beta\varepsilon_2$, with $\alpha \in [-0.5, 0.5], \beta \in [-0.5, 0.5]$. Moreover, layer-wise normalization is applied, following [31]. Sharp minima are consistently observed in all cases. These correspond to regions in the loss landscape where the loss function varies rapidly. Mathematically, this also aligns with the previously shown large Hessian eigenvalues, indicating high curvature. The comparison of the loss landscapes of PINNs and PI-DeepONets reveals that PINNs tend to have sharper minima. This is consistent with the intuition that sharper minima often correspond to poorer generalization performance [32], although this may not hold strictly in all cases. Moreover, this observation is consistent with expectations, given that the goal space $\mathcal{G}$ of PI-DeepONets is significantly larger than that of PINNs. In essence, PINNs are not expected to generalize well, as their test data replicates the scenarios used during training. Conversely, the test set for PI-DeepONets contains randomly sampled ICs, which will be detailed later. This indicates that optimizing PI-DeepONets is inherently more complex than optimizing PINNs.

4.2.2. Random ICs: PI-DeepONets

We evaluate the generalization of PI-DeepONets for different F_B by testing on 100 cases with uniformly sampled ICs within $[-s^{p,q}, s^{p,q}]$. Additionally, we ensure that the resulting values of p and q within the time domain stay within the same range. The maximum test time t_{max} is provided in Table 2. The solution trajectories of the test p and q from the FDM simulation are shown

[1] https://github.com/Xinmeng-Luan/bowmass-dafx

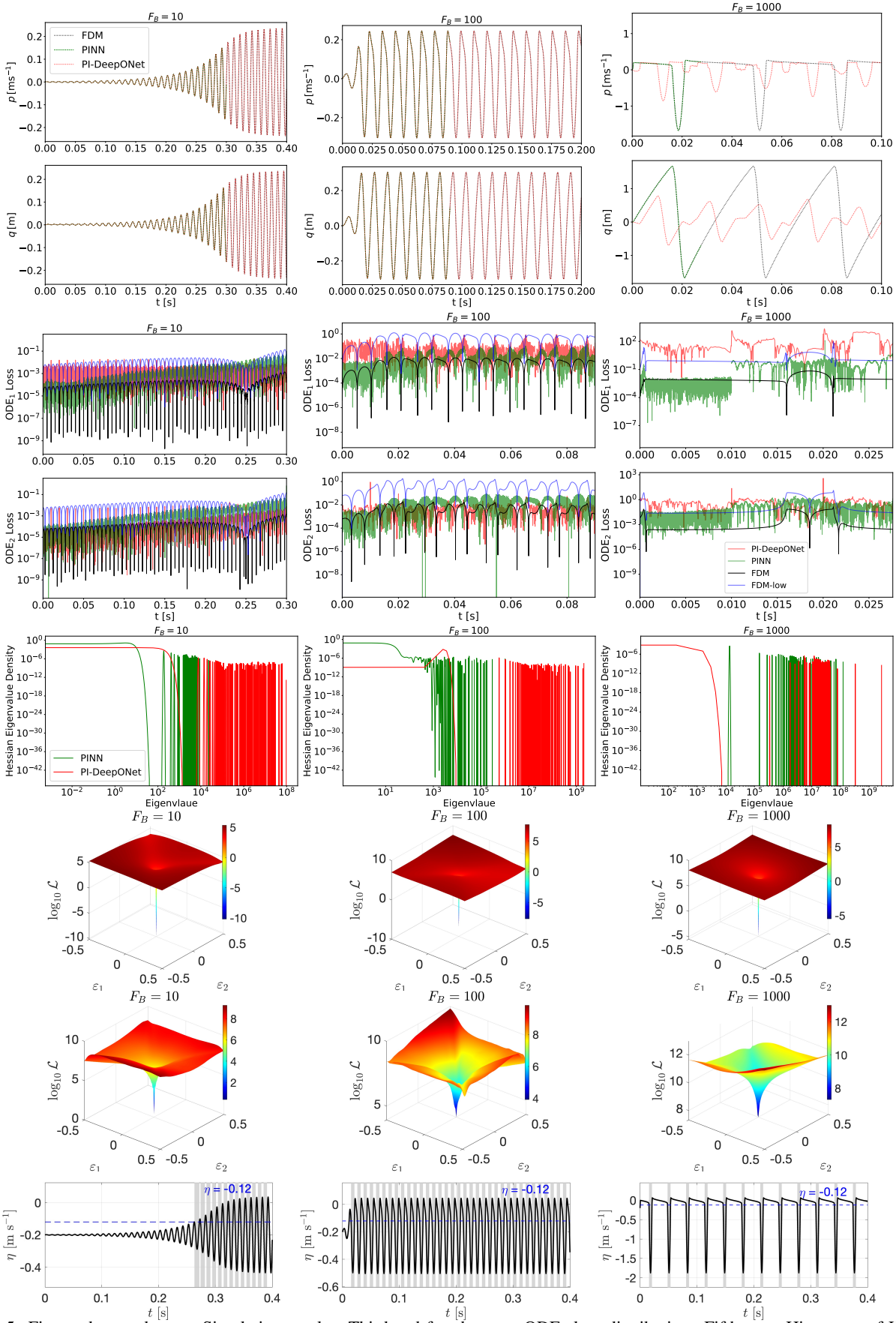

Figure 5: First and second rows: Simulation results. Third and fourth rows: ODEs loss distribution. Fifth row: Histogram of Hessian eigenvalue density. Sixth row: Loss landscape for PINNs. Seventh row: Loss landscape for PI-DeepoNets. Eighth row: Relative velocity η obtained from FDM, with the slip phases highlighted in the shaded regions.

in Fig. 6. For each case, the solid black region corresponds to

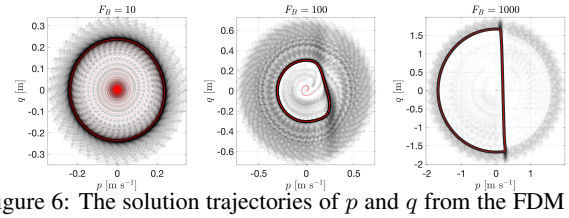

Figure 6: The solution trajectories of p and q from the FDM simulation with random ICs. The zero ICs cases are marked in red.

the steady-state solution, while the transient behavior varies with different ICs, resulting in the lighter black spots observed in the figure. We also mark the zero ICs cases in red. The mean normalized mean square error (NMSE) and normalized cross correlation (NCC) across 100 cases are reported in Table 2. We observe that the predictions for $F_B = 10$ and $F_B = 100$ are almost perfect, while the results for $F_B = 1000$ are notably poor.

Table 2: Prediction results of PI-DeepONets for randomly sampled ICs.

F_B	t_{max} [s]	NMSE(p)	NMSE(q)	NCC(p)	NCC(q)
10	0.4	1.70×10^{-8}	1.78×10^{-8}	100.00%	100.00%
100	0.2	2.57×10^{-3}	2.19×10^{-3}	99.87%	99.89%
1000	0.1	2.81	1.69	5.80%	3.37 %

4.2.3. Hybrid PI-DeepONets for the failure mode

The results demonstrate that both PINNs and PI-DeepONets can effectively solve the bowed mass-spring model. However, for $F_B = 1000$, PI-DeepONets fails to converge, while PINNs require time-marching and causal training. From a physics perspective, the nonlinear bow-string friction interaction is known to induce stick-slip behavior [8], which defines the bowing characteristics or texture of the sound. In the eighth row of Fig. 5, we highlight the slip phases corresponding to various bow force F_B scenarios, using the relative velocity η derived from the FDM simulation over the same time duration. The identification of stick-slip phases follows the criteria described in Section 2 and Fig. 2, where the slip phase is defined for $\eta \in [-0.12, 0.12]$. It is observed that in the bowed mass-spring model, a larger bow force F_B leads to a shorter slip phase, indicating that the system spends more time operating within the highly nonlinear region. This, in turn, leads to failure modes. The heightened complexity further challenges PI-DeepONets, making it harder for them to generalize across the broader goal space compared to PINNs.

A common approach to mitigating this issue is extending the framework by incorporating data-driven methods. For instance, integrating observation solutions from FDM as an additional term in the loss function term can help guide the network toward the expected solution, transforming the approach into a hybrid unsupervised–supervised training scheme. This idea has been well explored in PINNs research [20], where supervised training has been shown to significantly ease optimization. Therefore, we add

$$\mathcal{L}_{ob_1} = \frac{1}{N_{ob}} \|\hat{p} - p\|, \quad \mathcal{L}_{ob_2} = \frac{1}{N_{ob}} \|\hat{q} - q\|, \qquad (16)$$

where p and q are obtained from FDM with a high sampling rate, given zero initial conditions and $t \in [0, 0.1]$. The results are shown in Fig. 7. The evaluation metrics comparing the hybrid DeepONet with the high sampling rate FDM under zero initial conditions are: $\text{NMSE}(p) = 6.88 \times 10^{-3}$, $\text{NMSE}(q) = 1.72 \times 10^{-3}$, $\text{NCC}(p) = 99.66\%$ and $\text{NCC}(q) = 99.91\%$. Clearly, the hybrid DeepONets demonstrate accurate predictions.

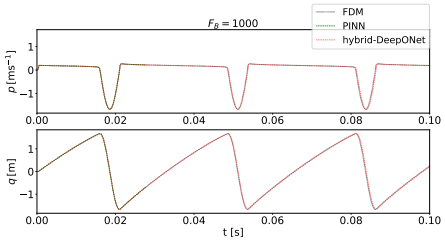

Figure 7: Comparison of simulation results for $F_B = 1000$: FDM vs. PINNs vs. hybrid DeepONets.

5. DISCUSSION AND CONCLUSION

In this study, PINNs and PI-DeepONets are utilized to solve the nonlinear bowed one-degree-of-freedom mass-spring system. The equation-solving task is formulated as an optimization problem and carried out within the framework of physics-informed deep learning. Scenarios with different bow forces have been investigated using both PINNs and PI-DeepONets.

The results show that while PINNs successfully solve the problem across all cases, PI-DeepONets perform well for low bow forces but struggle at higher bow forces. We further analyze the deep learning dynamics by examining the Hessian eigenvalue density and visualizing the loss landscape perturbed along the directions of the dominant Hessian eigenvalues. The presence of large Hessian eigenvalues and sharp minima in the loss landscape suggests ill-conditioned optimization. This highlights the challenge for purely physics-informed deep learning techniques for modeling the nonlinear bow-string friction interaction.

Although PINNs can handle scenarios with high bow force, they require additional strategies, such as causal training and the use of short time windows in the time-marching scheme to optimize effectively. While PINNs are limited by fixed ICs and face increasing computational costs as the number of time windows grows, PI-DeepONets excel in generalizing across varying ICs but may fail in highly ill-conditioned scenarios. Our findings suggest that a hybrid approach, combining the strengths of both physics-informed and data-driven methods, can alleviate the limitations of PI-DeepONets for higher bow forces.

As this is an initial exploratory study, our primary objective is to investigate the potential of physics-informed deep learning approaches for capturing nonlinearities in string instruments. Therefore, we do not yet demonstrate the use of the trained network for sound synthesis in real-world applications, though we definitely plan to pursue this in future work. For the deep learning approaches employed in this paper, we train a surrogate neural network to model the physical system, embedding the governing physical laws directly into the training process. While the training phase can be complex, requiring thoughtful strategy design and careful hyperparameter tuning, once trained, the model can be used for inference without explicitly solving the physics. This allows sound synthesis to be performed efficiently by simply running the network, without the need to solve equations in real-time, sample by sample, as is required in traditional numerical methods like FDM. Since the physical dynamics are already captured during training, the inference stage will be significantly faster, making deep learning approaches highly promising for achieving real-time operation, an essential yet challenging goal in physics-informed sound synthesis. We do not include inference in this work, as it falls outside the scope of this paper. However, we are actively

working on it and plan to explore it in future studies.

Furthermore, by emphasizing the differences between PINNs and PI-DeepONets in the solution space, we argue that PI-DeepONets are more suitable for sound synthesis applications due to their ability to cover a broader solution space. Considering the nature of musical signals, which typically include transient, steady-state, and decay stages while remaining predominantly periodic, DeepONets are particularly well-equipped to capture these characteristics. Moreover, DeepONets demonstrate a strong ability to train a single model capable of solving multiple systems governed by the same equation with varying parameters, making them highly adaptable for sound synthesis. On one hand, the results underscore the potential of physics-informed deep learning for modeling nonlinear physical processes in musical acoustics. On the other hand, the practicality of a purely physics-informed approach at the current technological level may vary depending on the specific task, as challenges remain in capturing complex nonlinearities. To translate these insights into practical sound synthesis applications, a hybrid approach that combines physics-informed and data-driven methods presents a more viable and effective path forward. Looking ahead, the next step will be to train a hybrid model that incorporates a variety of physical parameters as input, further enhancing its flexibility and applicability.

6. REFERENCES

[1] 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.

[2] S. Wang, H. Wang, and P. Perdikaris, "Learning the solution operator of parametric partial differential equations with physics-informed deeponets," *Sci. Adv.*, vol. 7, no. 40, p. eabi8605, 2021.

[3] S. Wang and P. Perdikaris, "Long-time integration of parametric evolution equations with physics-informed deeponets," *J. Comput. Phys.*, vol. 475, p. 111855, 2023.

[4] L. Lu, P. Jin, and G. E. Karniadakis, "Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators," *arXiv preprint arXiv:1910.03193*, 2019.

[5] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 34, pp. 26 548–26 560, 2021.

[6] S. Wang, S. Sankaran, and P. Perdikaris, "Respecting causality for training physics-informed neural networks," *Comput. Methods Appl. Mech. Eng.*, vol. 421, p. 116813, 2024.

[7] S. Bilbao, *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*. John Wiley & Sons, 2009.

[8] A. Chaigne and J. Kergomard, *Acoustics of musical instruments*. Springer, 2016.

[9] M. Olivieri, M. Pezzoli, F. Antonacci, and A. Sarti, "A physics-informed neural network approach for nearfield acoustic holography," *Sensors*, vol. 21, no. 23, p. 7834, 2021.

[10] X. Luan, M. Olivieri, M. Pezzoli, F. Antonacci, and A. Sarti, "Complex-valued physics-informed neural network for near-field acoustic holography," in *2024 32nd Eur. Signal Process. Conf (EUSIPCO)*. IEEE, 2024, pp. 126–130.

[11] X. Luan, M. Pezzoli, F. Antonacci, and A. Sarti, "Physics-informed neural network-driven sparse field discretization method for near-field acoustic holography," *arXiv preprint arXiv:2505.00897*, 2025.

[12] K. Yokota, T. Kurahashi, and M. Abe, "Physics-informed neural network for acoustic resonance analysis in a one-dimensional acoustic tube," *J. Acoust. Soc. Am.*, vol. 156, no. 1, pp. 30–43, 2024.

[13] K. Yokota, M. Ogura, T. Kurahashi, and M. Abe, "Physics-informed cnn for the design of acoustic equipment," in *2024 Int. Jt. Conf. Neural Netw. (IJCNN)*. IEEE, 2024, pp. 1–8.

[14] K. Yokota, M. Ogura, and M. Abe, "Synthesis of voiced sounds using physics-informed neural networks," *Acoust. Sci. Technol.*, vol. 45, no. 6, pp. 333–336, 2024.

[15] X. Luan, K. Yokota, and G. Scavone, "Acoustic field reconstruction in tubes via physics-informed neural networks," *arXiv preprint arXiv:2505.12557*, 2025.

[16] K. Yokota, M. Ogura, and M. Abe, "Identification of physical properties in acoustic tubes using physics-informed neural networks," *Mech. Eng. J.*, vol. 11, no. 5, pp. 24–00 228, 2024.

[17] C. D. L. V. Martin and M. B. Sandler, "Physical modelling of stiff membrane vibration using neural networks with spectral convolution layers," *Proc. 10th Conv. Eur. Acoust. Assoc. Forum Acusticum*, 2023.

[18] S. Schlecht, J. Parker, M. Schäfer, and R. Rabenstein, "Physical modeling using recurrent neural networks with fast convolutional layers," in *Int. Conf. Digit. Audio Eff. (DAFx)*, 2022, pp. 138–145.

[19] R. Diaz, C. D. L. V. Martin, and M. Sandler, "Towards efficient modelling of string dynamics: A comparison of state space and koopman based deep learning methods," in *Proc. of the 27th Int. Conf. Digit. Audio Eff. (DAFx)*, 2024, pp. 200–207.

[20] V. Gopakumar, S. Pamela, and D. Samaddar, "Loss landscape engineering via data regulation on pinns," *Mach. Learn. Appl.*, vol. 12, p. 100464, 2023.

[21] R. Russo, M. Ducceschi, and S. Bilbao, "Efficient simulation of the bowed string in modal form," in *Int. Conf. Digit. Audio Eff. (DAFx)*, 2022, pp. 122–129.

[22] E. Matusiak and V. Chatziioannou, "A comparison of friction models for bow-string interaction based on experimental measurements," in *Proc. Stockh. Music Acoust. Conf.*, 2023.

[23] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM J. Sci. Comput.*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[24] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, pp. 7537–7547, 2020.

[25] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *Int. Conf. Mach. Learn. (ICML)*. PMLR, 2019, pp. 5301–5310.

[26] S. Wang, H. Wang, and P. Perdikaris, "On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks," *Comput. Methods Appl. Mech. Eng.*, vol. 384, p. 113938, 2021.

[27] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, "An expert's guide to training physics-informed neural networks," *arXiv preprint arXiv:2308.08468*, 2023.

[28] N. Vyas, D. Morwani, R. Zhao, I. Shapira, D. Brandfonbrener, L. Janson, and S. Kakade, "Soap: Improving and stabilizing shampoo using adam," *arXiv preprint arXiv:2409.11321*, 2024.

[29] S. Wang, A. K. Bhartari, B. Li, and P. Perdikaris, "Gradient alignment in physics-informed neural networks: A second-order optimization perspective," *arXiv preprint arXiv:2502.00604*, 2025.

[30] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, "Pyhessian: Neural networks through the lens of the hessian," in *2020 IEEE Int. Conf. Big Data (Big data)*. IEEE, 2020, pp. 581–590.

[31] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 31, 2018.

[32] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.