A Step Towards High Frequency Physics-Informed Neural Networks

José Eduardo Alves Pereira Filho[1], Cédric Escudero[2], Sara Abdellaoui[3], Emil Dumitrescu[4] and Eric Zamaï[5]

Abstract— **This article explores an enhanced method for solving high-frequency forward problem Partial Differential Equations (PDEs) using Physics-Informed Neural Networks (PINNs). While PINNs have demonstrated success across multiple domains, they often face challenges in accurately capturing high-frequency components due to the inherent "spectral bias" of neural networks, where the network tends to focus on low-frequency features. To overcome these limitations, we introduce a novel approach that utilizes spline interpolation for signal augmentation and frequency decomposition, followed by transfer learning to progressively incorporate higher-frequency components into the learning process. The proposed method ensures smoother loss landscapes, making it easier for the network to learn high-frequency components. The proposed method is evaluated using a well-known forced mass-spring system, demonstrating the enhanced ability to capture high-frequency dynamics compared to conventional PINN methods.**

I. Introduction

Partial Differential Equations (PDEs) are fundamental mathematical tools used to describe dynamic phenomena across multiple domains. Therefore, solving those equations is crucial and Physics-Informed Neural Networks (PINNs) leverage neural networks to solve them. PINNs performance has been demonstrated across different fields such as fluid mechanics, power systems and even finance. Raissi et al. [1] have shown and popularized the use cases of PINNs and described their capabilities, dividing them into three main categories: (i) *Forward problem* to solve the PDE given its initial boundary conditions, (ii) *Inverse problem* to solve the PDE and find its parameters, (iii) *Dynamics discovery problem* to find hidden dynamics of a system given some possible relations.

Regarding the forward problem task of PINNs, most approaches target the architecture of the neural network itself [2], changing the loss function [3], transforming the data into different domains [4], preserving conservation laws [5], or even dividing the domain to parallelize computing power [5], [6]. Also other approaches use transfer learning [7] and interpolation techniques, the latter either applied to the data itself or as the network structure [8], [9]. Most of these approaches enable PINNs to address more complex non-linear problems incorporating not only the physical laws into the learning process by adding the physical PDE as part of the total loss function, but also introducing some other heuristic, thereby improving solution fidelity. However, they still struggle with inherit high frequency PDEs [10], [11].

In engineering, certain phenomena are governed by high-frequency PDEs that dictate the dynamics of the system, as demonstrated in studies such as [12]. Utilizing tools to solve these PDEs is crucial for a deeper understanding and accurate prediction of system responses. A fundamental example of this is the resistor-capacitor-inductor circuit, often employed as a passive resonator for signal transmission and reception [12]. While analytical solutions are feasible for simple linear cases, more intricate PDEs require numerical iterative methods for resolution. Physics-Informed Neural Networks is one such method, but they face challenges due to, what is often called in the literature, "spectral bias" or the "F-principle". A phenomenon where neural networks tend to focus on learning low-frequency features to minimize loss functions, potentially neglecting high-frequency or needing several more epochs or parameters to slowly learn high frequency components [11].

To tackle the challenges presented by high-frequency pathology cases in neural networks, several studies have focused on analyzing the spectral bias of neural networks. These studies aim to understand the relationship between higher frequencies and the learning rate or the number of epochs required for effective training. Some researchers employ Neural Tangent Kernels (NTK) and Fourier Features (FF) to explore the mathematical connections between learning ability and signal frequency [4], [10], [11], [13]. Others, such as De Ryck et al. [14], investigate this frequency relationship using the spectral Barron space, rigorously demonstrating the relationships between Feed Forward Neural Networks (FNN), stability, error generalization, and frequency.

Current methods fall short of fully integrating essential techniques like signal interpolation, transfer learning, and signal decomposition. Enhancing the spectral resolution by interpolating available data points to better capture the original frequency components can significantly improve the performance of PINNs, as demonstrated in [8]. Additionally, applying transfer learning from simpler problems within the same family of functions allows the network to learn higher frequency components of the original equation more efficiently, as shown in [7], [15].

[1]José Eduardo Alves Pereira Filho is with Univ Lyon, INSA Lyon, Université Claude Bernard Lyon 1, Ecole Centrale de Lyon, CNRS, Ampère, UMR5005, 69621 Villeurbanne, France (`jeduapf@gmail.com`)

[2]Cédric Escudero is with Univ Lyon, INSA Lyon, Université Claude Bernard Lyon 1, Ecole Centrale de Lyon, CNRS, Ampère, UMR5005, 69621 Villeurbanne, France (`cedric.escudero@insa-lyon.fr`)

[3]Sara Adbellaoui is with Univ Lyon, INSA Lyon, Université Claude Bernard Lyon 1, Ecole Centrale de Lyon, CNRS, Ampère, UMR5005, 69621 Villeurbanne, France (`sara.adbellaoui@insa-lyon.fr`)

[4]Emil Dumitrescu is with Univ Lyon, INSA Lyon, Université Claude Bernard Lyon 1, Ecole Centrale de Lyon, CNRS, Ampère, UMR5005, 69621 Villeurbanne, France (`emil.dumitrescu@insa-lyon.fr`)

[5]Eric Zamaï is with Univ Lyon, INSA Lyon, Université Claude Bernard Lyon 1, Ecole Centrale de Lyon, CNRS, Ampère, UMR5005, 69621 Villeurbanne, France (`eric.zamai@insa-lyon.fr`)

In this article, we take a step towards solving high-frequency forward problems. By implementing a signal spline interpolation and further decomposing the augmented signal into its most energetic frequency components, we apply gradual transfer learning sessions to the same network to capture a better loss landscape, intuitively giving a less flat loss landscape to the next training session and reducing the final loss compared to classical PINNs. This paper presents a proof of concept rather than an extensive analysis of different use cases. By successfully addressing a simple, well-known phenomenon, we gain valuable insights that can guide future work on more complex real-world problems, particularly in inverse problems using PINNs.

This article is structured as follows. Section II revisits the PINN framework in its most general form as well as synthesizes recent advancements in enhancing PINNs for addressing high-frequency PDEs. Section III presents the proposed method for solving intrinsic high-frequency PDEs, further implemented in a well-known forced mass-spring case study example at Section IV where results are discussed comparing with other high-frequency PINNs methods. Finally, Section V discusses the conclusions and suggests directions for future work.

For comprehensive documentation regarding the implementation details and source code, readers are directed to the project repository accessible at: GitHub.

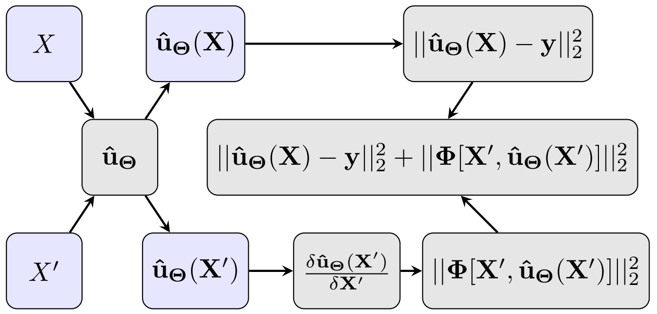
II. Related works on Physics-Informed Neural Networks

A. Classical Physics-Informed Neural Networks

Physics-Informed Neural Networks are both simple and versatile, as they leverage the powerful approximation capabilities of neural networks through physics regularization. As rigorously demonstrated by [16], neural networks can serve as universal function approximators. Consequently, a simple architecture such as a FNN can be employed to describe the intrinsic solution of a PDE. In this setup, the independent variables, act as the inputs of the network, while the dependent variables serves as the output. The PINN framework, popularized by Raissi et al. in [1], adds significant value by incorporating a physics-based regularization loss to the regression loss. As mentioned earlier, there are three main approaches to implement PINNs: forward problems, inverse problems, and discovery problems. In this article, the primary focus is on forward problems, where all the parameters of the PDE are known, and the objective is to find its solution.

The general regression problem dataset, can be described as a finite data points collection $(\mathbf{X}, \mathbf{y}) = \{(\mathbf{x_i}, \mathbf{y_i}) \mid i = \{1, \dots, P\}$, where $\mathbf{X} \in \mathbb{K}^{P \times N}$ are the P independent variables points with N features in a given set $\mathbb{K}$, and $\mathbf{y} = \mathbf{u}(\mathbf{X}) \in \mathbb{K}^{P \times M}$ are the dependent variables that describes the system $\mathbf{u}$. For the sake of generality, the set $\mathbb{K}$ is a generic set depending on the variables. In this article, we assume a random collection of internal and boundary points for $\mathbf{X}$. Each one of the dependent variables $\mathbf{y_i}$ are mapped as $\mathbf{y_i} = \mathbf{u}(\mathbf{x_i})$ where $\mathbf{y_i}$ is the i-th row of $\mathbf{y}$. For input points $\mathbf{X}$, it is possible to use either boundary points or internal points within the domain $\mathbf{\Omega}$ that spans all possible points of the independent variables in the context of PINNs. For instance, if we consider an electric motor dynamics described by a differential equation, $\mathbf{X}$ is the time t and $\mathbf{y}$ is the angular position.

By utilizing a Feed Forward Neural Network as a universal estimator for the unknown function $\mathbf{u}(\mathbf{X})$, with a neural network with parameters Θ, it is possible to minimize the quadratic error between the estimated values $\hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X})$ and the known values $\mathbf{y} = \mathbf{u}(\mathbf{X})$ of the data points using gradient descent algorithms. This constitutes a classical regression task, typically employing the Euclidean norm (or L_2 norm) to ensure smoother function approximations, although other norms or distance measures may also be used [17].



X
$\hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X})$
$||\hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X}) - \mathbf{y}||_2^2$
$\hat{\mathbf{u}}_{\mathbf{\Theta}}$
$||\hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X}) - \mathbf{y}||_2^2 + ||\mathbf{\Phi}[\mathbf{X}', \hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X}')]||_2^2$
X'
$\hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X}')$
$\frac{\delta \hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X}')}{\delta \mathbf{X}'}$
$||\mathbf{\Phi}[\mathbf{X}', \hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X}')]||_2^2$

Fig. 1: Classical PINN workflow

To incorporate physics into the loss manifold alongside the regression loss, it is necessary to include the PDE equation in the total loss function. Following the general form of a PDE as presented in [14], we introduce a non-linear operator $\mathbf{\Phi}[*]$ that encapsulates the PDE equation with all its non-zero terms. The physics-informed loss function $\mathbf{f}(\mathbf{X})$ can then be expressed as:

$$\mathbf{f}(\mathbf{X}) := \Phi[\mathbf{X}, \mathbf{u}(\mathbf{X})] = 0, \quad \mathbf{X} \in \mathbf{\Omega} \tag{1}$$

Through automatic differentiation of the estimation of $\mathbf{u}(\mathbf{X})$, the physics-informed loss function $\mathbf{f}(\mathbf{X})$ can be expressed in terms of the neural network parameters Θ, as it is a function of both $\mathbf{X}$ and $\mathbf{u}(\mathbf{X})$. Consequently, the estimated physics-informed loss function can be expressed as $\hat{\mathbf{f}}_{\mathbf{\Theta}}(\mathbf{X}) := \Phi[\mathbf{X}, \hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X})]$. By employing the L_2 error to $\hat{\mathbf{f}}_{\mathbf{\Theta}}(\mathbf{X})$ and using the same network with parameters Θ as the estimator for the PDE solution $\hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X})$, the total loss function $L(\Theta)$ is defined as:

$$L(\Theta) = ||\hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X}) - \mathbf{u}(\mathbf{X})||_2^2 + ||\hat{\mathbf{f}}_{\mathbf{\Theta}}(\mathbf{X}')||_2^2 \tag{2}$$

Here, $\mathbf{X}$ represents all the input points for which $\mathbf{u}(\mathbf{X})$ is available (i.e., measured points within the domain and their corresponding function outputs). Conversely, $\mathbf{X}'$ denotes a chosen distribution of physics points in the domain, typically sampled evenly. The amount of physics points $\mathbf{X}'$ may differ from the amount of data points $\mathbf{X}$, with $\mathbf{X}'$ often containing a greater number of evenly spaced points. Hence, $\mathbf{X}' \in \mathbb{K}^{Q \times N}$ and $\mathbf{u}(\mathbf{X}') \in \mathbb{K}^{Q \times M}$, where $Q \geq P$. Since this approach can act as a form of regularization, as in [11], one can introduce a constant that multiplies the physics-informed regularization term to balance the contributions of the regression and physics losses.

The function $\hat{\mathbf{f}}_{\mathbf{\Theta}}(\mathbf{X}')$ is computed using automatic differentiation, which enables the network-approximated solution $\hat{\mathbf{u}}_{\mathbf{\Theta}}(\mathbf{X})$ to be differentiated with respect to $\mathbf{X}'$ efficiently and accurately. The complete scheme of PINN implementation is illustrated in Figure 1. Thanks to recent advancements in automatic differentiation algorithms, as well as improvements in computational precision and speed, PINNs have gained significant attraction, as noted in [1].

Finally, the goal is to minimize the total loss function $L(\Theta)$ in (2) with respect to the network parameters Θ using a gradient-based optimization algorithm. Typically, this is done using either the ADAM optimizer, which leverages adaptive learning rates and momentum, or the L-BFGS algorithm, a quasi-Newton method that approximates the Hessian matrix to accelerate convergence. Both methods aim to iteratively update Θ to reduce the total loss, driving the model towards a solution that better fits the data $(\mathbf{X}, \mathbf{y})$ and satisfies the underlying physical constraints.

Although the method appears promising and simple to apply, there are some caveats, particularly for more complex systems or those with high-frequency oscillations. Such systems are of significant interest, especially in the design of electrical systems, such as motors or electronics, for instance [12]. Nevertheless, even for simple, well-known systems such as mass-spring systems, PINNs often fail to accurately model high-frequency data, despite providing a sufficient number of data points in accordance with Nyquist's law as discussed in [11] and shown in Section IV.

B. Physics Informed Neural Networks applied to high-frequency problems

In order to comprehend the relationship between high-frequency features and the learning capabilities of Physics-Informed Neural Networks, some studies have used Neural Tangent Kernels (NTK) to interpret the frequency components in the learning process [10], [13]. The NTK is a mathematical tool used to highlight the characteristics of a particular strategy in the training of neural networks. It is not part of the implementation of the algorithm, but rather a mathematical apparatus to understand the intrinsic association between the variables and inputs of a neural network. Therefore, the general model of a neural network, as well as the fundamentals of NTKs, are presented here to explain the use of Fourier Features (FFs) in the PINN framework as Fourier Features PINNs (FFPINNs).

Jacot et al. [18] demonstrated the use of NTKs to describe phenomena in neural network training by modeling a neural network in the infinite width limit. To investigate the implications of Fourier Features on a theoretical basis, [13] explains how the method can be understood by decomposing the matrix $\mathbf{K} \in \mathbb{K}^{P \times P}$ of the kernel regression of the underlying neural network function into its eigenvalues, which can be further linked to frequency components of the data. For instance, a univariate neural network with parameters Θ, denoted by $u_\Theta(x)$, can be approximated in terms of a kernel regression $k(x_i, \mathbf{X})$ as follows:

$$u_\Theta(x) \approx \sum_{i=1}^{P} \mathbf{Z}_i k(x_i, \mathbf{X}) \tag{3}$$

where i corresponds to the i-th element of the dataset $\mathbf{X}$, $k(x_i, \mathbf{X})$ is the kernel applied to point x_i and $\mathbf{Z} = \mathbf{K^{-1}y}$

In this sense, [18] shows that the kernel approximates the dynamics of the network, which can be applied to demonstrate that, given a Fourier Feature input to the network, its ability to learn high-frequency features does not decrease at the same rate as when using the original inputs, which tend to decrease exponentially. The FFs network is an input mapping function γ that transforms the input values $\mathbf{X} \in \mathbb{K}^{P \times N} \rightarrow \gamma(\mathbf{X}) \in \mathbb{K}^{P \times O}$, further fed into the network as $\mathbf{u}_{\mathbf{\Theta}}(\mathbf{X}) \rightarrow \mathbf{u}_{\mathbf{\Theta}}(\gamma(\mathbf{X}))$. The transformation is of the form shown in equation (4), where $\mathbf{b_k} \in \mathbb{K}^O$ are the basis frequencies, sometimes drawn independently from Gaussian distribution and $a_k \in \mathbb{K}$ are the scale factors, where $k = \{1, \ldots, O\}$:

$$\gamma(X) = \left[a_1 \cos(2\pi \mathbf{b_1^T X}), \ldots, a_N \cos(2\pi \mathbf{b_O^T X})\right]^T \tag{4}$$

For instance, Tang et al. [4] implement a simplified version of the FF technique by letting $a_k = 1$, $\forall k \in \{1, \ldots, O\}$ in (4). They also retain the original input values alongside with the Fourier Features as the network's input, which implies $O = P + 1$, to solve one-dimensional radiation transport problems, allowing the capture of high-frequency details with smoother loss behavior.

Another interesting approach to tackle high-frequency artifacts and improving efficiency is proposed by Moseley et al. [6]. To enhance not only the high-frequency components in the PDE solution produced by the neural network but also to reduce the training time, Moseley proposed a domain decomposition method. The method consists in parallelizing the training phase of the network, as well as keeping the sub-boundary intersections to maintain the continuity of the solution using a differentiable window function.

Using a sigmoid-based window function to subdivide the complete domain as many times as desired, it is possible to subdivide the standard PINN domain to a Finite Basis PINN (FBPINN). Moreover, for more complex functions with high-frequency details, it is observed that FBPINN performs better than standard PINN. It is also shown that for simpler problems, FBPINN and PINN produce similar solutions and convergence rates. Only for higher frequencies and more complex functions FBPINN significantly outperforms the classical version.

Finally, another method used to address high-frequency elements in PDEs is transfer learning, as demonstrated in [7]. Transfer learning techniques involve pre-training a neural network on a general case or a simpler, yet related, problem, as extensively discussed in [15]. Specifically, the transfer learning approach employed in [7] is a data-driven transfer learning method, focusing on a feature-based approach. The core idea is to uncover latent relationships between the transferred domains.

III. Spline Transfer Learning PINNs

This section outlines the main stages of the proposed PINN method to tackle the high-frequency forward problem PDE.

A. General idea

By integrating three techniques, our proposed method called Spline Transfer Learning Physics-Informed Neural Network (STLPINN) highlights the high-frequency components of the original signal $(\mathbf{X},\mathbf{y})$ and filters each component into a separated augmented signal. The network is first trained on the lowest frequency of the decomposed signal, gradually incorporating additional frequency components until the entire original signal $(\mathbf{X},\mathbf{y})$ is reconstructed. The primary goal is to break down the problem into smaller sub-problems, providing a smoother not flat loss landscape to the network with each new frequency component addition. This approach enables the network to capture higher frequencies more effectively, without relying on heuristics or using parallel computing strategies.

The methodology consists of three main stages schematized in Figure 2. The first stage (Signal Augmentation) pre-processes the original signal $(\mathbf{X},\mathbf{y})$ to increase the amount of the data points. This first stage creates an augmented signal $(\tilde{\mathbf{X}},\tilde{\mathbf{y}})$ with $\tilde{\mathbf{X}} \in \mathbb{K}^{P'\times N}$ and $\tilde{\mathbf{y}} \in \mathbb{K}^{P'\times M}$, $P' > P$, where P' is the amount of resampled points after the signal augmentation stage. Then, the second stage (Input Decomposition) decomposes the augmented signal $(\tilde{\mathbf{X}},\tilde{\mathbf{y}})$ into ζ new signals, one for each identified frequency, denoted as $(\tilde{\mathbf{X}},\tilde{\mathbf{y}}_{f_\eta})$ with $\eta \in \{1,\dots,\zeta\}$. Thereafter, the last stage called Transfer Learning is applied, which involves a succession of ζ training sessions. The objective of those training sessions is to train the network ζ times using as regression data a simpler problem, the cumulative decomposed signal $(\tilde{\mathbf{X}},\tilde{\mathbf{y}}_\eta)$ with $\tilde{\mathbf{y}}_\eta = \sum_{e=1}^{\eta} \tilde{\mathbf{y}}_{\mathbf{f_e}}$ (see Figure 2). During the last training session $\zeta+1$, the physics points $\mathbf{X}'$ are then added to train the previously trained network $\hat{\mathbf{u}}^{\zeta}_{\mathbf{\Theta}}$. This network can learn in a less flat loss manifold due to a better estimation of the system $\mathbf{u}$, accelerating the gradient descent without relying on any heuristics.

In order to have a good resolution, selecting a number of physics points that is 4 times the number of augmented data points proves effective, which means that for P' augmented data points $(\tilde{\mathbf{X}},\tilde{\mathbf{y}}_\zeta)$, there should be Q physics points $\mathbf{X}'$, where $Q \approx 4P'$. These P' data points have already been resampled at a higher rate through spline interpolation in the signal augmentation stage (see Section III-B).

This approach relies fundamentally on the availability of observed data points, as its primary objective is to enhance equation solving within a data-driven PINN framework. Without available data, the method defaults to a vanilla PINN that solely enforces initial and boundary conditions, rendering the STLPINN extension inapplicable. A practical implementation would initially deploy a standard PINN to generate a sparse domain data set, subsequently transitioning to the STLPINN formulation to refine both the solution and parameter estimates.

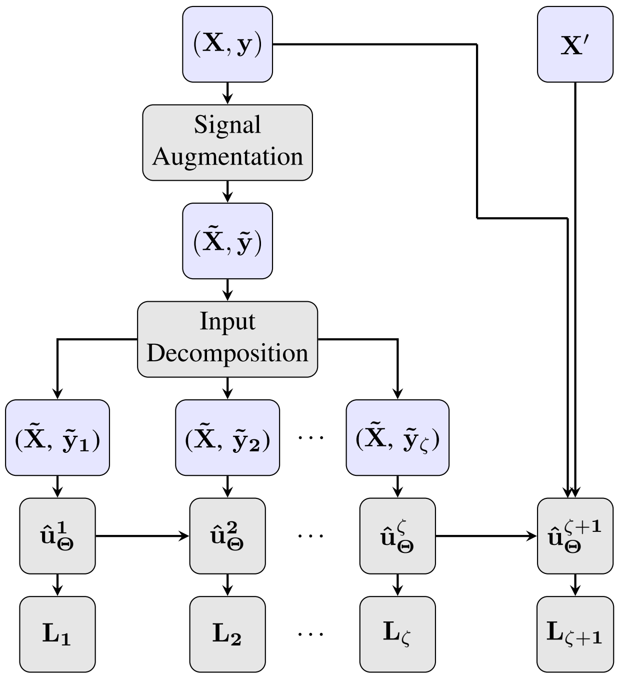

$(\mathbf{X},\mathbf{y})$
$\mathbf{X}'$
Signal Augmentation
$(\tilde{\mathbf{X}},\tilde{\mathbf{y}})$
Input Decomposition
$(\tilde{\mathbf{X}},\tilde{\mathbf{y}}_1)$
$(\tilde{\mathbf{X}},\tilde{\mathbf{y}}_2)$
$\cdots$
$(\tilde{\mathbf{X}},\tilde{\mathbf{y}}_\zeta)$
$\hat{\mathbf{u}}^{1}_{\mathbf{\Theta}}$
$\hat{\mathbf{u}}^{2}_{\mathbf{\Theta}}$
$\hat{\mathbf{u}}^{\zeta}_{\mathbf{\Theta}}$
$\hat{\mathbf{u}}^{\zeta+1}_{\mathbf{\Theta}}$
$\mathbf{L_1}$
$\mathbf{L_2}$
$\mathbf{L}_\zeta$
$\mathbf{L}_{\zeta+1}$

Fig. 2: Spline Transfer Learning PINN workflow. Signal augmentation and decomposition on data points $(\mathbf{X},\mathbf{y})$, that are separated into several new data points $(\tilde{\mathbf{X}},\tilde{\mathbf{y}}_{\mathbf{f}_\eta})$ added cumulatively to train the model $\hat{\mathbf{u}}^{\eta}_{\mathbf{\Theta}}$ at each session η resulting in a final loss L_η. In the last training session $\zeta+1$, the trained model at session ζ, denoted by $\hat{\mathbf{u}}^{\zeta}_{\mathbf{\Theta}}$, is used with the original data points $\mathbf{X}$ and the physics collocation points $\mathbf{X}'$.

B. Signal Augmentation

This first stage, the signal augmentation, begins by increasing the amount of the original data points $(\mathbf{X},\mathbf{y})$, from P to P' using spline interpolation [19]. The original data points are at least a multiple of the Nyquist minimum requirement, based on the maximum frequency of the signal, expressed as νf_{max} where $\nu \geq 2$, $\nu \in \mathbb{R}^+$. This ensures sufficient sampling points before the signal augmentation for accurate signal representation after the spline interpolation.

Spline interpolation is a piecewise polynomial method used to approximate a function over a given dataset [19]. The v-th order spline is defined as a set of v-degree polynomials, each describing the behavior of the function in an interval between two consecutive data points. These polynomials are constructed to ensure smoothness at the boundaries, satisfying continuity up to the $(v-1)$-th derivative. Given the set of data points of a PINN as $(\mathbf{X},\mathbf{y}) = \{(\mathbf{x}_i,\mathbf{y}_i) \mid i = \{1,\dots,P\}$, developed in Section II-A, the v-th order spline interpolation for $x \in [x_i, x_{i+1}]$ is:

$$S_i(x) = \sum_{j=0}^{v} a_{i,j}(x-x_i)^j \tag{5}$$

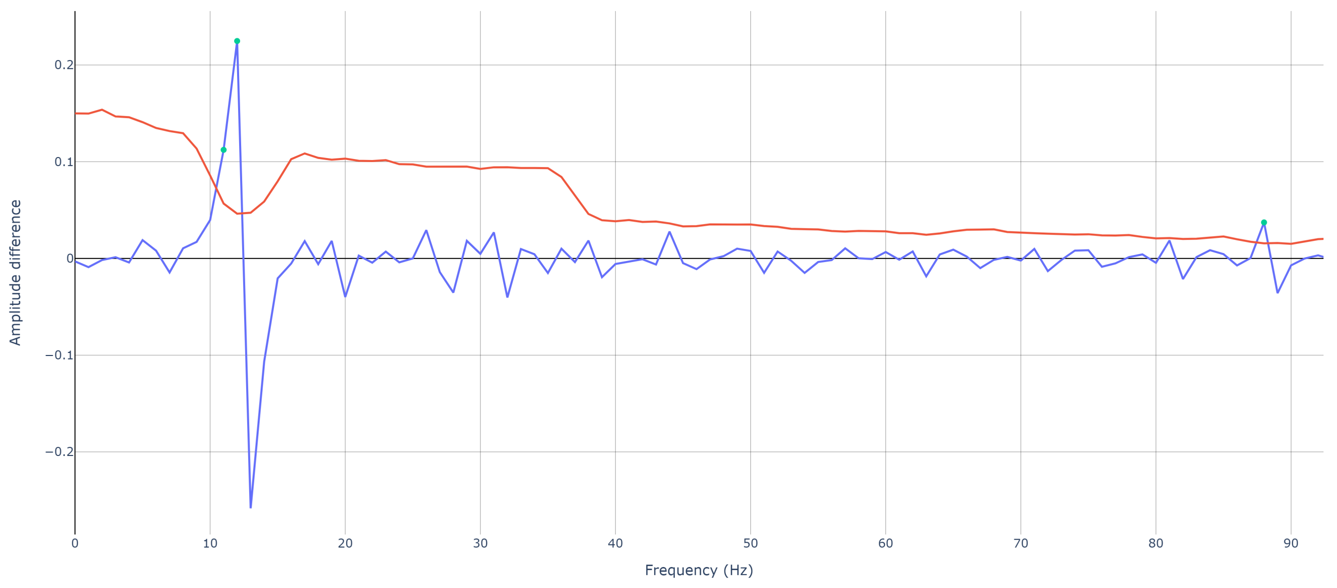


Fig. 3: CFAR implementation over a $f_0 = 12.7\ Hz$, $f_{max} = 89.1\ Hz$ system – Green dots are the CFAR identified peaks, the red curve is the CFAR threshold and the blue curve is the FFT difference $(|FFT(f_{k+1})| - |FFT(f_k)|)$.

where $S_i(x)$ is the function that describes any point $x \in [x_i, x_{i+1}]$. The coefficients $a_{i,j} \in \mathbb{R}$ are determined based on the continuity conditions of two different interpolation functions: $S_i^{(\tau)}(x_{i+1}) = S_{i+1}^{(\tau)}(x_{i+1})$ for $\tau = \{1, \ldots, n-1\}$ for a given $n \geq 2 \in \mathbb{N}$, where $S_i^{(\tau)}(x)$ represents the τ-th derivative of $S_i(x)$. Finally, the boundary constraints $S_i(x_i) = y_i$ and $S_{i+1}(x_{i+1}) = y_{i+1}$ are imposed.

The augmented signal, denoted by $(\tilde{\mathbf{X}}, \tilde{\mathbf{y}})$, now with P' data points chosen to be $P' \geq P$ enables clearer peak identification by Constant False Alarm Rate (CFAR) in the Fast Fourier Transform (FFT) difference $(|FFT(f_{k+1})| - |FFT(f_k)|)$ than the classical uniformly sampled signal with the original amount of points, simplifying the filtering process in the frequency domain, as demonstrated in [19]. Figure 3 shows the application of this technique in the study case in Section IV.

Although increasing the spline degree v generally yields a more accurate approximation—with smoother curves and fewer low frequency interpolation artifacts—the computational and practical costs often outweigh these benefits. Lower-degree splines are faster to evaluate and simpler to interpret, and they impose lighter demands during model training: by using fewer high-frequency basis functions, they avoid introducing unwanted oscillations into the fitted function. Therefore, in this paper a 3rd degree spline is used.

C. Input Decomposition

After identifying the frequency components in the augmented signal FFT using CFAR, the augmented signal is filtered to isolate each of the found frequency components in this second stage. These filters can be simple m th-order Butterworth filters with a margin around the desired frequency. If two or more peaks are within the defined margin, a single filter is constructed with the central frequency defined as the average of the peaks. This results into ζ filtered augmented signals denoted as $(\tilde{\mathbf{X}}, \tilde{\mathbf{y}}_{\mathbf{f}_\eta})$ with $\eta \in \{1, \ldots, \zeta\}$. The cumulative value $\tilde{\mathbf{y}}_\eta$ alongside with $\tilde{\mathbf{X}}$ will then feed each training session η during the transfer learning stage.

The CFAR peak identification parameters must be set depending on the resolution of the augumented signal. For simplicity, the implemented version uses 0.1% guard cells and 0.9% band cells with respect to the length of the augumented signal, while using a 20% margin in 6-th order Butterworth filter.

D. Transfer Learning

With the filtered augmented network's inputs and outputs $(\tilde{\mathbf{X}}, \tilde{\mathbf{y}}_\eta)$, the last stage of the method initiates with ζ transfer learning training sessions. At each training session η, the input of the neural network is the augmented points of the domain, $\tilde{\mathbf{X}}$, that will be regressed with respect to the cumulative decomposed outputs, $\tilde{\mathbf{y}}_\eta$, where $(\tilde{\mathbf{X}}$,$\tilde{\mathbf{y}}_\eta)$ are given in Section III-A. At each training session η, the network $\hat{\mathbf{u}}_{\mathbf{\Theta}}^{\eta}$ is trained from the pre-trained network at session $\eta - 1$ and then transferred to the next training session. At the ζ-th training session, the neural network $\hat{u}_{\mathbf{\Theta}}^{\zeta}$ is trained from the pre-trained network $\hat{\mathbf{u}}_{\mathbf{\Theta}}^{\zeta-\mathbf{1}}$ with the inputs $\tilde{\mathbf{X}}$ and the outputs $\tilde{\mathbf{y}}_\zeta \approx \tilde{\mathbf{y}}$. Thereafter, the last training session $\zeta + 1$ trains the neural network $\hat{\mathbf{u}}_{\mathbf{\Theta}}^{\zeta+\mathbf{1}}$ from the pre-trained network $\hat{\mathbf{u}}_{\mathbf{\Theta}}^{\zeta}$, the data points $(\mathbf{X}, \mathbf{y})$ and the physics points $\mathbf{X}'$.

The transfer learning is conceptually straightforward, but computationally intensive. Indeed, it requires retraining the entire network $\zeta + 1$-times $(\hat{\mathbf{u}}_{\mathbf{\Theta}}^{\mathbf{1}}, \ldots, \hat{\mathbf{u}}_{\mathbf{\Theta}}^{\zeta+\mathbf{1}})$, each training session η with the respective cumulative decomposed signal $(\tilde{\mathbf{X}}, \tilde{\mathbf{y}}_\eta)$. Once the network $\hat{\mathbf{u}}_{\mathbf{\Theta}}^{\eta}$ is trained with the entire reconstructed signal $(\tilde{\mathbf{X}}, \tilde{\mathbf{y}_\zeta})$ the final training session $\zeta + 1$ use the original data points $(\mathbf{X}, \mathbf{y})$ and physics points $\mathbf{X}'$. During this transfer learning stage, the network retains the weights acquired from previous training sessions, leveraging them to enhance learning efficiency.

Intuitively, postponing the enforcement of the physics constraint until the final training stage speeds up STLPINNs and prevents the oscillations that often arise from high-order derivative terms in the physics loss. By focusing first on fitting the data points, the model enjoys a smoother, more stable loss landscape extracting the maximum information from observations before the physics constraints are imposed.

IV. Study Case

A. System description

To validate our method, we consider a well-known damped mass-spring system which is a second-order Differential Equation in time with an analytically obtainable complete solution given in (6):

$$m\frac{\partial^2 u(t)}{\partial t^2} + b\frac{\partial u(t)}{\partial t} + ku(t) = F_0 \sin(2\pi f_{max} t) \quad (6)$$

with $u(t)$ the position of the mass, $\dot{u}(t)$ its velocity with $m = 1$, $b = 4$ and $k = (2\pi f_0)^2$. f_0 is the natural resonant frequency and f_{max} is the maximum frequency of the system i.e. $f_{max} = q \times f_0$ with a chosen $q = 7$. Notice that the value of k and the forced input (right-hand side term in (6)) will vary with respect to f_0 to study the high frequency features.

By solving the PDE in (6), we can generate a number of data points, $P = \lfloor 2 \times f_{max} \rfloor$, respecting the Nyquist requirement as described in section III-B. The number of physics points is then chosen to be $Q = 4 \times P$ as explained in section III-C. Finally the initial conditions for the system are set as $u_0 = 1$ and $\dot{u}_0 = 0$.

B. PINN physics-informed loss

From (6), the physics-informed loss from (2) is:

$$||m\frac{\partial^2 u_\Theta(t)}{\partial t^2} + b\frac{\partial u_\Theta(t)}{\partial t} + ku_\Theta(t) - F_0 \sin(2\pi f_{max} t)||_2^2 \quad (7)$$

Recall that the PINN optimizes the network parameters Θ such that the physics loss in (7) converges to zero.

C. PINN experiment and settings

After having described the system under study, we now present the results of the proposed method in comparison with three other methods presented in Section II: classical PINNs, FBPINNs and FFPINNs. Regarding the neural network architectures, PINN, FFPINN, and STLPINN consist of 6 layers with 80 neurons per layer, whereas the FBPINN consist of 2 layers with 64 neurons per layer, for each of its 9 subdomain divisions. Notice that, an increment of weights is added for FFPINN to account for the input domain expansion into sines and cosines, $N = 64$ in equation (4), to serve as its domain features explained in Section II-B. Although the network architectures are slightly different, the amount of learnable parameters is similar as shown in Table I.

This experiment is conducted for different frequencies w_0 ranging from $[20, 40, 80, 100, 200, 250, 300, 350]$ rad/s, where $w_0 = 2\pi f_0$. Table I summarizes the results, including the resonance frequency ω_0 (rad/s), the total training time (min), the total normalized final loss multiplied by a factor of 10^{-3} (calculated as the sum of the data loss and the regularized physics loss normalized to 1 at the first iteration) and finally the number of learnable parameters of the network (”Params” in Table I).

For STLPINNs, two frequency peaks should be identified in the augmented input signal during the input decomposition stage, i.e. f_0 and f_{max} (see Figure 3). Hence, two training sessions are performed ($\zeta = 2$ in Figure 2): at training session $\eta = 1$, the network $\hat{\mathbf{u}}^{\mathbf{1}}_{\mathbf{\Theta}}$ is trained with the first decomposition of the augmented input signal containing the frequency f_0, at session $\eta = 2$ the network is trained with the augmented input signal containing the frequency f_0 and the one containing the frequency f_{max}. In some experiments, however, additional frequency peaks were observed due to artifacts introduced during the spline augmentation phase. The network was trained in the same manner but with more training sessions, resulting in longer training times and a greater number of iterations. These cases were excluded in presented results.

All calculations were performed on a laptop equipped with an NVIDIA GeForce GTX 1070 with Max-Q Design and 16 GB of RAM, running a Python 3.7 environment with PyTorch version 1.13 and CUDA 11.6. The trainings were point by point, no batch implementation. Notice that STLPINNs train once per frequency, with iterations limited to match other methods.

D. Results analysis and discussion

The proposed method, STLPINN, performs better than other methods for higher-frequency problems. For instance, at $\omega_0 = [250, 300, 350]$, STLPINN achieves a lower loss compared to other methods. However, it takes longer to converge for all frequencies, notably lower-frequency problems due to its pre-processing and multiple training sessions, as seen for $\omega_0 = [20, 40]$ in Table I.

Surprisingly, FFPINNs yield stunning results up to 100 rad/s, being fast and precise. Meanwhile, FBPINNs exhibit the same convergence issue as PINNs for higher frequency problems. The high derivative in the physics loss results in unstable loss values, causing the training phase to oscillate significantly.

As explained in Section III, the central idea of STLPINNs is to transfer a better loss landscape to the next training session, helping the network to learn in a less flat loss manifold. This accelerates the gradient descent when the original signal contains higher frequency components, without relying on any heuristics. This is why the loss of STLPINNs remains more stable compared to others, even when higher frequency components are present in each problem for a constant number of parameters. Figure 4 and Figure 5 reinforce this idea by showing the total normalized loss of PINN and STLPINN for the 80 rad/s study case scenario.

In conclusion, the proposed method demonstrates a good performance regardless of the signal's frequency components. However, it requires longer training times compared

TABLE I: Experiment results for different frequency w_0 with PINN, FFPINN, FBPINN and the proposed method STLPINN

PINN				**FFPINN**				**FBPINN**				**STLPINN**			
ω_0	**Time**	**Final Loss**	**Params**	ω_0	**Time**	**Final Loss**	**Params**	ω_0	**Time**	**Final Loss**	**Params**	ω_0	**Time**	**Final Loss**	**Params**
20	6.0	1.0	32 641	20	0.4	1.0	42 865	20	0.1	1.0	39 177	20	6.4	1.0	32 641
40	16.4	261.9	32 641	40	3.3	1.0	42 865	40	0.1	1.0	39 177	40	19.9	7.52	32 641
80	20.7	409.5	32 641	80	10.4	1.0	42 865	80	4.2	17.0	39 177	80	25.1	9.43	32 641
100	26.6	337.2	32 641	100	12.2	1.0	42 865	100	5.4	27.7	39 177	100	48.3	9.36	32 641
200	49.9	706.4	32 641	200	81.4	146.3	42 865	200	9.6	539.7	39 177	200	124.1	9.18	32 641
250	74.6	717.5	32 641	250	95.8	184.9	42 865	250	11.4	709.7	39 177	250	181.5	179.0	32 641
300	104.6	820.9	32 641	300	122.8	377.5	42 865	300	13.8	751.7	39 177	300	390.9	257.9	32 641
350	152.6	766.6	32 641	350	174.9	549.2	42 865	350	16.8	865.3	39 177	350	254.9	391.7	32 641

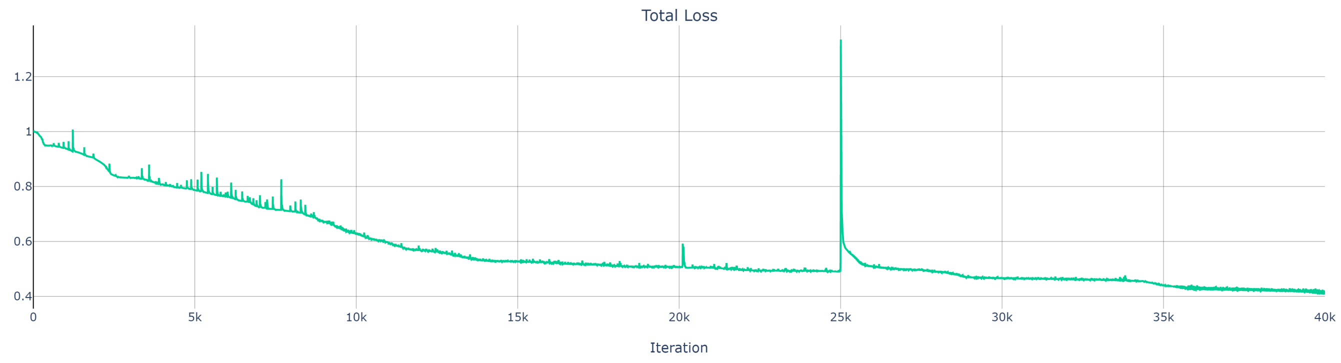


Fig. 4: PINN normalized losses for frequency 80 rad/s.

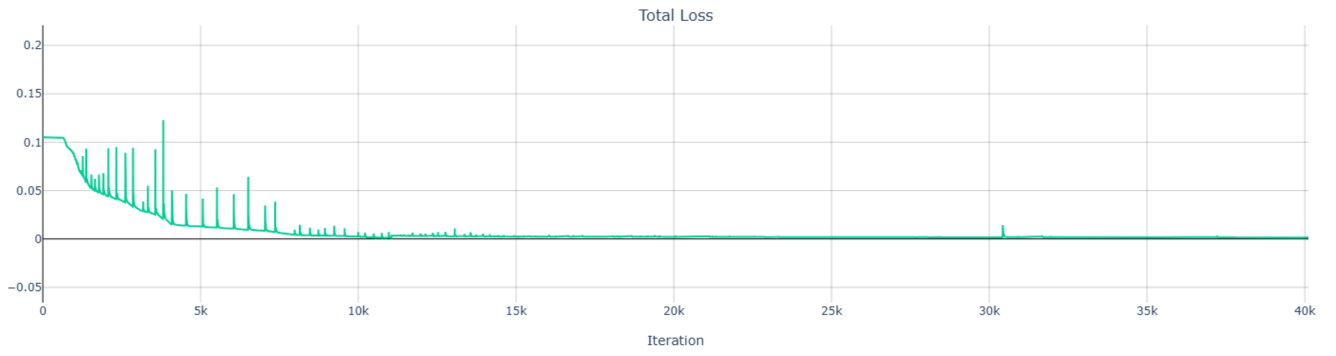


Fig. 5: PINN normalized losses for frequency 80 rad/s.

to FFPINNs and FBPINNs. While these alternative methods perform better for a specific range of frequencies, their computational cost increases significantly, following a power-law relationship with the maximum input frequency.

Although the results are promising, more detailed experimentation can be conducted by broadening the range of test systems and including additional frequency components. Further it is intended to explore inverse-problems approaches for determining the PDE parameters that govern a system's dynamics. In practice, both the forward and inverse formulations of STLPINNs can be deployed to infer a real system's internal equations, either to assess divergence from its nominal PDE initial model by minimizing the forward-problem loss divergency, or to monitor parameter drift over time via the inverse problem. This capability is particularly valuable in domains such as electrical circuits, transformers, and motors, where external measurements are readily available but direct internal inspection is costly or disruptive. In these settings, STLPINNs offer a noninvasive means to diagnose and track hidden system parameters without halting operation.

V. Conclusion

In this article, we proposed a novel Physics-Informed Neural Networks (PINNs) to effectively address the challenges posed by high-frequency components in solving forward problem Partial Differential Equations (PDEs). By augmenting input data through spline interpolation and progressively incorporating higher-frequency components via transfer learning, the proposed method improves both the accuracy and efficiency of PINNs. The successful application of this method to the forced mass-spring system demonstrates its potential to tackle complex, high-frequency dynamics in various engineering and physics problems. Future work will focus on refining the method (e.g. adding batch optimization, domain parallelization) and exploring its applicability to more diverse and complex PDE systems.

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," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[2] P. Ren, C. Rao, Y. Liu, J.-X. Wang, and H. Sun, "PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs," *Computer Methods in Applied Mechanics and Engineering*, vol. 389, p. 114399, 2022.

[3] L. Yang, X. Meng, and G. E. Karniadakis, "B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data," *Journal of Computational Physics*, vol. 425, p. 109913, 2021.

[4] Q. A. Huhn, M. E. Tano, and J. C. Ragusa, "Physics-informed neural network with fourier features for radiation transport in heterogeneous media," *Nuclear Science and Engineering*, vol. 197, no. 9, 2023.

[5] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, 2020.

[6] B. Moseley, A. Markham, and T. Nissen-Meyer, "Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations," *Advances in Computational Mathematics*, vol. 49, no. 4, p. 62, 2023.

[7] A. H. Mustajab, H. Lyu, Z. Rizvi, and F. Wuttke, "Physics-informed neural networks for high-frequency and multi-scale problems using transfer learning," *Applied Sciences*, vol. 14, no. 8, p. 3204, 2024.

[8] S. Tang, X. Feng, W. Wu, and H. Xu, "Physics-informed neural networks combined with polynomial interpolation to solve nonlinear partial differential equations," *Computers & Mathematics with Applications*, vol. 132, pp. 48–62, 2023.

[9] N. Wandel, M. Weinmann, M. Neidlin, and R. Klein, "Spline-PINN: Approaching PDEs without data using fast, physics-informed hermite-spline CNNs," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, pp. 8529–8538, 2022.

[10] S. Wang, X. Yu, and P. Perdikaris, "When and why pinns fail to train: A neural tangent kernel perspective," *Journal of Computational Physics*, vol. 449, p. 110768, 2022.

[11] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: Where we are and what's next," *J. Sci. Comput.*, vol. 92, Sept. 2022.

[12] J.-L. Kotny, X. Margueron, and N. Idir, "High-frequency model of the coupled inductors used in emi filters," *IEEE Transactions on Power Electronics*, vol. 27, no. 6, pp. 2805–2812, 2012.

[13] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[14] T. De Ryck and S. Mishra, "Numerical analysis of physics-informed neural networks and related models in physics-informed machine learning," *Acta Numerica*, vol. 33, 2024.

[15] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[16] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[17] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," *Annals of Data Science*, vol. 9, no. 2, pp. 187–212, 2022.

[18] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[19] L. Horowitz, "The effects of spline interpolation on power spectral density," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, pp. 22–27, Feb. 1974.