Physics informed neural networks for extreme mechanics problems

Jeongsu Lee

Department of mechanical, smart, and industrial engineering, Gachon University, Seongnam 13120, South Korea

Corresponding authors: leejeongsu@gachon.ac.kr

Abstract

Physics-informed neural networks (PINNs) constitute a continuous and differentiable mapping function that approximates a solution curve to given physical laws. Although recent extensive studies have exhibited great potential for PINNs as an alternative or complement to conventional numerical methods, the lack of accuracy and robustness remains a challenging problem. As a remedy, this study explores optimal strategies for constructing PINNs by resolving several extreme problems where the solution based on a regular PINN was not very effective. Examples include dynamic beam bending, fluid-structure interaction, and 1-D advection in high velocity. The novel strategies proposed are: 1) scaling analysis for the configuration, 2) incorporation of the second-order non-linear term of input variables, and 3) use of a neural network architecture that reflects a series solution of decomposed bases. The proposed approaches are shown to significantly improve PINN predictions, exhibiting an order-of-magnitude improvement in accuracy compared to regular PINNs. This study is expected to provide crucial baselines for constructing PINNs, which could be extended to high-order coupled differential equations.

1. Introduction

Physics-informed neural networks (PINN) have attracted extensive scientific interest in recent years owing to their versatility in problem configurations, including forward problem to find solution curves of integral and differential equations [1-4] and inverse problem for data-driven discovery under the regulation of given physical laws [5-7]. Since the first invention of PINN, which incorporates the residuals of partial differential equations (PDEs) into the loss function of neural networks [1], several pioneering studies have greatly elaborated PINN from various aspects, including neural network architectures [8-10], loss functions [11-13], training strategies [14-16], and practical implementation techniques [17-19].

Despite extensive studies aimed at elaborating on the PINN approach, it remains challenging to construct a PINN model for various physical and engineering problems. The solution of PINN significantly changes with hyperparameters [10, 14] and often converges to a trivial case that merely prompts zero-values over a certain region in the spacetime domain [10, 14, 19, 20]. It has been suggested that the failures of PINN originate from its inherent properties rather than the lack of expressivity in neural networks [10, 14]. In general, PINN constructs the loss function from the weighted summation of PDE residuals and mean-squared errors of initial and boundary conditions (ICs and BCs) [1]. This approach could involve an ill-conditioned problem, where the loss landscape changes dramatically by the selection of loss regularization coefficients, resulting in a non-convex optimization problem [10, 14].

Obtaining a reliable PINN solution can be difficult, especially when solving high-order differential equations. High-order differential equations are accompanied by high-order gradients that can easily vanish during neural network training. For instance, previous studies describing dynamic beam bending based on PINNs are surprisingly rare, even though it is a fundamental concept in engineering problems [21, 22]. It is worth noting that while the PINN solution describing static beam bending has been extensively studied [23-26], the dynamic counterpart has not. The relevant high-order differentials in a dynamic beam bending problem, fourth-order in space and second-order in time, could limit the acquisition of a reliable solution based on PINNs. This will be confirmed later in this study. Therefore, resolving high-order differential equations remains one of the challenges in PINNs.

This study explores optimal strategies for constructing PINN to solve several extreme problems, including high-order differential equations. The bottlenecks that hinder the construction of a reliable PINN solution are discussed, and possible remedies are proposed. Demonstrations of these remedies are provided through several examples. First, the free vibration of a cantilevered beam is considered, which is then extended to a much more challenging problem involving the coupled interaction of fluid and structure. Next, the study examines the 1D advection equation under extreme conditions, where a regular PINN is reported to fail in obtaining a reliable solution [14].

2. Proposed approach

The PINN aims to construct a continuous and differentiable mapping function Φ that approximates some physical quantity u while adhering to its governing equation. Suppose that the governing equation of u can be expressed as a differential operator $\mathcal{D}$ defined in space x and time t as follows:

$$\mathcal{D}(u(x,t)) = 0 \quad (1)$$

Here, space $x \in \Omega \subset \mathbb{R}^n$ and Ω is the spatial domain of dimension n and time $t \in [0,\infty]$. The mapping function constructed with neural networks is denoted as $\Phi(x,t \mid \theta)$ with trainable weights θ. In general, feed-forward neural networks with fully connected layers and tanh activation are employed to construct PINN [1, 2, 14].

The weights θ of the constructed PINN can be trained using gradient descent by minimizing the loss function $\mathcal{L}$. The loss function is constructed from the residuals of the governing equation $\mathcal{L}_{PDE}$, the initial conditions $\mathcal{L}_{IC}$, and the boundary conditions $\mathcal{L}_{BC}$. The relative importance of each loss component can be adjusted by introducing regularization coefficients α_{PDE}, α_{IC} and α_{BC}, resulting in the following expression for the total loss function $\mathcal{L}_{tot}$:

$$\mathcal{L}_{tot} = \alpha_{PDE}\mathcal{L}_{PDE} + \alpha_{IC}\mathcal{L}_{IC} + \alpha_{BC}\mathcal{L}_{BC} \quad (2)$$

When adopting PINN for inverse problems, an additional loss term accounting for the prediction error with respect to the measurements, $\mathcal{L}_{Data}$, can be included [1, 2]. In general, PINN exhibits better performance when resolving inverse problems. This study only considered forward problem without the guidance from the data, $\mathcal{L}_{Data}$, to examine the robustness of the proposed strategies for constructing PINN.

2.1 Scaling analysis

The convergence and accuracy of PINN are significantly impacted by various problem configurations, such as loss regularization coefficients, physical variables in PDEs, and reference values in PINN. However, configuring PINN is often challenging due to imbalances in physical dimensions and scales that can arise. The loss function is constructed from different physical dimensions in each loss category of PDEs, ICs, and BCs, and there is no guarantee that the scales of loss components are well-balanced even within the same loss category. For instance, differential equations with second-order time derivatives require two initial conditions having different physical dimensions, which can result in scale imbalances in $\mathcal{L}_{IC}$. To address these issues, scaling analysis can provide an advantageous initial estimate for the PINN configuration and help resolve any scale imbalances.

Let's assume we want to use PINN, $\Phi(x,t \mid \theta)$, to solve the differential operator $\mathcal{D}$, which can be decomposed into spatial differential $\mathcal{D}_x$ and temporal differential $\mathcal{D}_t$, in the domain of in $-\frac{L}{2} < x < \frac{L}{2}$ and $0 < t < \tau_h$. We can scale the PDE residuals, $\mathcal{D}_x(\Phi(x,t))$ and $\mathcal{D}_t(\Phi(x,t))$, with the physical variables of the problem, such as the characteristic length scale L and desired time horizon τ_h. For example, if the PDE is constructed from first-order spatial and temporal differentials, with $\mathcal{D}_x = \partial/\partial x$ and $\mathcal{D}_t = \partial/\partial t$, the scales of the differential operators correspond to $\partial/\partial x \sim 1/L$ and $\partial/\partial t \sim 1/\tau_h$, respectively. If the characteristic length scale is much larger than the time scale $L \gg \tau_h$, the differential operator can be approximated as $\mathcal{D} \approx \mathcal{D}_t$. Then, a trivial solution that remains constant in time would be found.

One way to prevent this problem is to adjust the time scale of the PINN to ensure that the scales of the differential operators are of the same order of magnitude, $O(\mathcal{D}_x) \sim O(\mathcal{D}_t)$.

This can be achieved by adopting a time-marching strategy [14, 15], which iteratively trains a series of PINN with a time step of τ until the desired time horizon τ_h is reached. The time step τ is then selected to balance the scales between differential operators. For the exemplified first-order differential operators, the scaling as $O(\tau) \sim O(L)$ is obtained from the requirement $O(\mathcal{D}_x) \sim O(\mathcal{D}_t)$.

In addition, the scale imbalance in the loss function must be further addressed. The conventional regularization coefficients α_{PDE}, α_{IC} and α_{BC} in equation (2) are insufficient for resolving the scale imbalance in second or higher-order differential equations. For example, when solving a second-order ODE (ordinary differential equation) for $u(t)$ in time t, the ICs can be given as $u(0)$ and $(du/dt)|_{t=0}$. If MSE (mean-squared-error) is used as the loss metric for $\mathcal{L}$, the resulting scales in each loss component, $\mathcal{L}_{IC,1}(\Phi(t)|_{t=0}, u(0))$ and $\mathcal{L}_{IC,2}(\mathrm{d}\Phi(t)/\mathrm{dt}|_{t=0}, (du/dt)|_{t=0})$, correspond to $O(E(u)^2)$ and $O(E(u)^2)/\tau^2$, respectively, with the operator E denoting the prediction error. Thus, a significant scale imbalance could arise due to the time step τ. To address this scale imbalance in $\mathcal{L}_{IC}$, additional regularization coefficients, such as $\alpha_{IC,1}$ and $\alpha_{IC,2}$, can be employed by enforcing the scales in each loss component to be of the same order of magnitude with an additional requirement, $\alpha_{IC,1}\mathcal{L}_{IC,1} \sim \alpha_{IC,2}\mathcal{L}_{IC,2}$. One choice to meet this requirement could be to set $\alpha_{IC,1} = 1$ and $\alpha_{IC,2} = O(\tau^2)$.

The above argument can be generalized using the notations of $\mathcal{L}_{v,i}$ and $\alpha_{v,i}$ for the loss components and regularization parameters, respectively. Here, $v \in \{\mathrm{PDE}, \mathrm{IC}, \mathrm{BC}\}$ corresponds to the category of the loss, and i is the loss index in each category. The total loss is then rewritten as a weighted summation of loss components with regularization coefficients: $\mathcal{L}_{tot} = \sum_v \sum_i \alpha_{v,i}\mathcal{L}_{v,i}$. The basic idea is to adjust the coefficients $\alpha_{v,i}$ by enforcing additional conditions that the scales of the weighted loss components be of the same order-of magnitude. For example, $\alpha_{v,i}\mathcal{L}_{v,i} \sim O(1)$.

Suppose that the loss component $\mathcal{L}_{v,i}$ considers the MSE of a physical variable $u_{v,i}$ with a desired relative tolerance $\varepsilon_{v,i}$ for the PINN solution. The prediction error can be scaled as $E_{v,i} \sim \varepsilon_{v,i} O(u_{v,i})$, leading to the loss component expressed as $\mathcal{L}_{v,i} \sim (E_{v,i})^2 \sim \varepsilon_{v,i}^2 O(u_{v,i}^2)$. Therefore, the regularization coefficients $\alpha_{v,i}$ have to be scaled as $1/\varepsilon_{v,i}^2 O(u_{v,i}^2)$ to adjust the scales of the weighted loss components to be of $O(1)$. Finally, the alternative description of the total loss is obtained as

$$\mathcal{L}_{tot} = \sum_v \sum_i \frac{\mathcal{L}_{v,i}}{\varepsilon_{v,i}^2 O(u_{v,i}^2)} \tag{3}$$

Here, the order-of magnitude of the target physical variable $O(u_{v,i})$ can be determined from the scaling analysis for the given physical problems, which is exemplified in the later sections.

As practical guidance, this study suggests using an initial guess of desired tolerances $\varepsilon_{v,i}$ as 10^{-2} for PDEs and 10^{-3} for BCs and ICs. The PDE loss tends to encourage convergence to a trivial solution of the PDE, such as a zero-valued function over the entire spacetime domain. On the other hand, the loss from ICs and BCs typically causes neural networks to reproduce ICs regardless of time change. Therefore, finding an optimal solution between two misguided

points is desired. One effective way found in this study is to guide neural networks to learn temporal variations obeying the PDE from the ICs/BCs. Then, the suggestion of the PDE tolerance being 10 times smaller than that of ICs and BCs usually works well to find a solution for a given problem.

One of the easiest ways to satisfy the requirements from scaling analysis is to start with governing equations in dimensionless form, which configures the most of dimensional variables and their derivatives to be scaled as $O(1)$. However, when the problem involves multiple characteristic scales in the same physical dimension, for example, the characteristic length scales in x- and y-dimensions are distinct, the proposed scaling analysis could be adopted to resolve the scaling imbalances in a more rigorous way. Furthermore, the scale of the dimensionless number in the governing equation could involve additional imbalance compared to ICs and BCs, which have to be properly resolved based on scaling analysis.

2.2 Variants in activation

The last layer of a PINN is typically constructed by combining hidden variables linearly, which can be expressed as:

$$\sum_i W_i \sigma_i(f(x,t)) + b \tag{4}$$

where f represents the non-linear operator of hidden layers except the last layer, and W_i and b correspond to the weights and biases in the last layer, respectively. The activation function σ plays a significant role in describing the mathematical expression of the PINN prediction. In this regard, the activation could express the expected mathematical representation of the solution curve. Although the tanh activation widely used in PINNs has a number of merits [27], there exist limitations that can deteriorate the expressivity of the neural networks in some problems [28, 29].

First, the tanh activation function exhibits poor performance in reproducing periodic behavior [28]. This could be a serious limitation of the PINN since periodic functions are fundamental base functions that construct solution curves of many physical problems. Second, the tanh activation hardly conveys a tendency described from the second-order expansion of the given input vector x_j. The first non-linear term in Taylor series expansion of tanh, $\tanh(x_j) = x_j - \frac{{x_j}^3}{3} + \frac{2{x_j}^5}{15} + O({x_j}^7)$, appears at the third-order, which implies that the second-order tendency could not be directly transferred to the next layer. This limitation can be resolved by adding a number of layers, as the second-order variants are reconstructed from the increased higher-order terms. However, this leads to the addition of unnecessary layers as well.

To address the above-described problems, this study employed an alternative activation function, $x_j + \sin^2(x_j)$, proposed by Ziyin et al. (2020). However, this activation introduces a correlation between the phase and scale in the output, as it directly bypasses the input x_j and incorporates a periodic function $\sin^2(x_j)$ with x_j as its phase. To overcome this limitation, a linear scaling layer $L_S(x_j) = wx_j$ can be added after the activation, resulting in the output $L_S(x_j + \sin^2(x_j))$. The scaling constant w is determined during training. Although the linear scaling layer may seem unnecessary, it allows the network to explore the

outcome of the activation without being affected by the correlation between phase and scale. This approach improves the convergence and robustness of the neural network in experiments. Similarly, adding a linear scaling layer before the activation can also improve performance, which is known as adaptive activation [30, 31].

2.3. Neural networks architecture

Revisiting the expression for the PINN prediction in equation (4), it can be interpreted as a truncated series solution with elements of the form $W_i\sigma_i$. Therefore, it can be argued that the PINN describes a solution curve similar to that obtained from series solution methods, such as power series, Fourier series, and Laplace decomposition [32, 33]. This argument inspires the following idea to extend the PINN architecture as a combination of parallels, which directly reflects the decomposed bases of a series solution.

Suppose that the PINN aims to describe a series solution of two decomposed bases. In this case, the neural network architecture can be modified as $\hat{u}(x,t) = \sum_k g_k(x,t)h_k(x,t)$, where g and h denote the decomposed bases produced from independent neural networks $G(x,t \mid \theta)$ and $H(x,t \mid \theta)$, respectively. The k index corresponds to the truncated number of series solution, and $\hat{u}$ is the final prediction of the PINN. Moreover, the input parameters of the decomposed bases can be designated to guide the PINN prediction. If we separate x and t inputs for each decomposed base, the PINN would represent the spatiotemporal decomposition (STD) and the prediction would be of the following form: $\hat{u}(x,t) = \sum_k g_k(x)h_k(t)$. The proposed neural network architectures will be further illustrated in the experimental sections that follow.

3. Experiments

3.1. Vibration of a cantilever

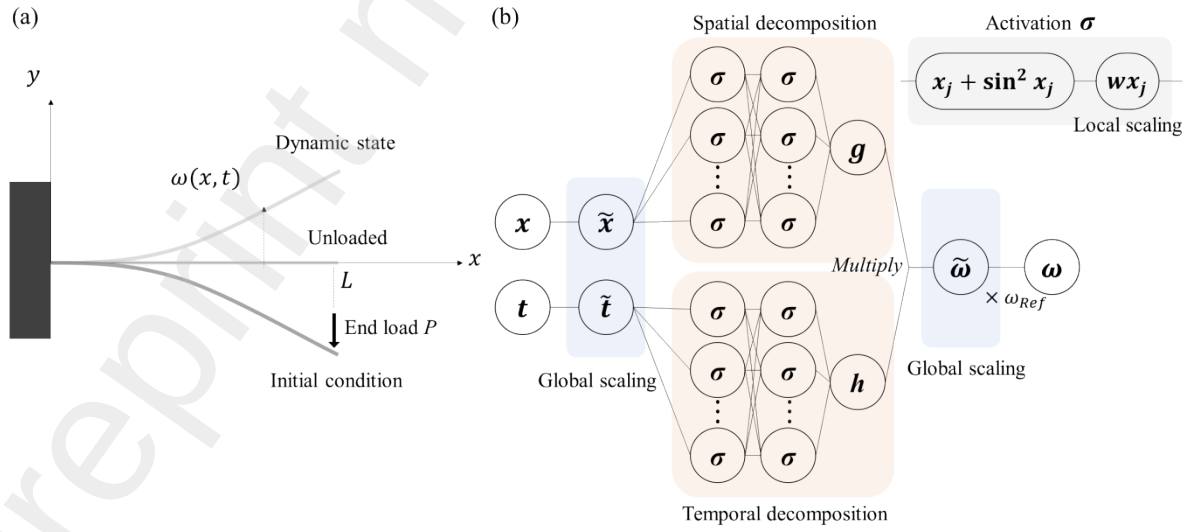

Figure 1. (a) Schematic illustration of the free vibration of a cantilevered beam. (b) Neural network architecture reflecting the STD approach employed to resolve the vibration of a cantilever.

3.1.1. Modeling

The vibrations of a cantilevered beam shown in Figure 1(a) can be described by the dynamic Euler-Bernoulli bending theory under the small amplitude assumption, which can be expressed as:

$$\mu\frac{\partial^2\omega(x,t)}{\partial t^2} + B\frac{\partial^4\omega(x,t)}{\partial x^4} = q(x,t) \tag{5}$$

Here, $\omega(x,t)$ represents the vertical deflection of the beam at the horizontal position x and time t. The horizontal position x varies from 0 to the beam length L. The parameters μ and B are the mass per unit length and flexural rigidity, respectively. For a rectangular cross-section, the mass per unit length μ can be expressed as $\rho_s hl$, where ρ_s is the density of the beam, h is the thickness, and l is the width. The flexural rigidity is the product of Young's modulus E and the area moment of inertia of the cross-section I, which can be expressed as $B = EI$. The area moment of inertia of the plate I can be calculated using the formula $I = \frac{h^3 l}{12}$.

The cantilevered beam is subject to fixed and free end boundary conditions at $x = 0$ and $x = L$, respectively. These conditions require that $\omega = 0$ and $\partial\omega/\partial x = 0$ at $x = 0$, and $\partial^2\omega/\partial x^2 = 0$ and $\partial^3\omega/\partial x^3 = 0$ at $x = L$, and are denoted as BC,1 to BC,4 in the following configuration. Additionally, to simulate the free vibration of the cantilevered beam, we consider an initial condition corresponding to a static cantilevered plate with end load *P*. This initial condition is characterized by the initial displacement of $\omega(x,0) = Px^2(3L - x)/6EI$ and an initial velocity $\partial\omega(x,0)/\partial t$ of zero. At the start of the simulation ($t = 0$), the load is released, and the cantilevered beam begins to vibrate freely, without any transverse load ($q(x,t) = 0$). The initial displacement and velocity conditions are denoted as IC,1 and IC,2, respectively.

3.1.2. Configuration

The characteristic time scale, τ, and characteristic length scale of deflection, ω_{Ref}, are considered at the beginning of the configuration. The time scale, τ, can be described by balancing the inertia and restoring forces in the governing equation (5), resulting in $\tau \sim (\frac{\mu}{B}L^4)^{1/2}$. The ω_{Ref} can be scaled by balancing the initial load and the restoring force, giving $\omega_{\text{Ref}} \sim PL^3/B$. These characteristic scales allow for the expression of the following conditions for each loss component.

The PDE error $E_{PDE,1}$ represents the difference between $\mu\frac{\partial^2\omega(x,t)}{\partial t^2}$ and $B\frac{\partial^4\omega(x,t)}{\partial x^4}$ in equation (5) in the absence of the external load $q(x,t)$. The PDE loss can be scaled using a desired relative tolerance $\varepsilon_{PDE,1}$ such that $E_{PDE,1} \sim \varepsilon_{PDE,1}\mu\frac{\partial^2\omega(x,t)}{\partial t^2} \sim \frac{\varepsilon_{PDE,1}\mu\omega_{\text{Ref}}}{\tau^2}$, leading to the scaling of the PDE loss as $\mathcal{L}_{PDE,1} \sim {E_{PDE,1}}^2 \sim (\frac{\varepsilon_{PDE,1}\mu\omega_{\text{Ref}}}{\tau^2})^2$. Similar arguments can be made for the BCs and ICs. Specifically, $\mathcal{L}_{BC,1} \sim (\varepsilon_{BC,1}\omega_{\text{Ref}})^2$, $\mathcal{L}_{BC,2} \sim (\varepsilon_{BC,2}\frac{\omega_{\text{Ref}}}{L})^2$, $\mathcal{L}_{BC,3}$

$\sim (\varepsilon_{BC,3}\frac{\omega_{\text{Ref}}}{L^2})^2$, and $\mathcal{L}_{BC,4} \sim (\varepsilon_{BC,4}\frac{\omega_{\text{Ref}}}{L^3})^2$ represent the loss due to boundary conditions, while $\mathcal{L}_{IC,1} \sim (\varepsilon_{IC,1}\omega_{\text{Ref}})^2$ and $\mathcal{L}_{IC,2} \sim (\varepsilon_{IC,2}\frac{\omega_{\text{Ref}}}{\tau})^2$ represent the loss due to initial conditions.

By integrating the above scaling relations into equation (3), the total loss $\mathcal{L}_{tot}$ can be rewritten as:

$$\mathcal{L}_{tot} = \tilde{\mathcal{L}}_{PDE} + \tilde{\mathcal{L}}_{IC} + \tilde{\mathcal{L}}_{BC} \tag{6}$$

, where the scaled losses are defined as follows:

$$\tilde{\mathcal{L}}_{PDE} = \frac{\tau^4}{(\varepsilon_{PDE,1}\mu\omega_{\text{Ref}})^2}\mathcal{L}_{PDE}$$

$$\tilde{\mathcal{L}}_{IC} = \frac{1}{(\varepsilon_{IC,1}\omega_{\text{Ref}})^2}\mathcal{L}_{IC,1} + \frac{\tau^2}{(\varepsilon_{IC,2}\omega_{\text{Ref}})^2}\mathcal{L}_{IC,2}$$

$$\tilde{\mathcal{L}}_{BC} = \frac{1}{(\varepsilon_{BC,1}\omega_{\text{Ref}})^2}\mathcal{L}_{BC,1} + \frac{L^2}{(\varepsilon_{BC,2}\omega_{\text{Ref}})^2}\mathcal{L}_{BC,2} + \frac{L^4}{(\varepsilon_{BC,3}\omega_{\text{Ref}})^2}\mathcal{L}_{BC,3} + \frac{L^6}{(\varepsilon_{BC,4}\omega_{\text{Ref}})^2}\mathcal{L}_{BC,4}$$

The relative tolerances are adopted as $\varepsilon_{IC} = \varepsilon_{BC} = 10^{-3}$ and $\varepsilon_{PDE} = 10^{-2}$ corresponding to the guidance for the initial guess in the former section except for the elaboration made for $\varepsilon_{BC,1} = 10^{-4}$. The loss construction based on the suggested scaling law dramatically improves the convergence of the PINN. It is worth noting that the hyperparameter search based on the regular configuration from equation (2) couldn't obtain a reliable solution in my experiments (see Supplementary Information). This is supported by the absolute lack of the PINN solution for the dynamic beam bending problem [23], despite its abundance in engineering applications.

3.1.3. Neural networks architecture

Figure 1(b) shows the PINN architecture used to solve the free vibration of a cantilever. The model is designed to take two input variables x and t, defined in the ranges $[0, L]$ and $[0, \tau]$, respectively, where τ is the characteristic time scale used in the configuration. A time-marching strategy is applied to extend the time horizon of the solution [14, 15]. This involves training a sequence of independent PINN models with a time step τ, using the predictions of the previous model as the initial conditions for the next one.

To separate the scales of trainable weights by physical dimensions, the PINN model has adopted two different global scaling layers at the top and bottom of the architecture. In the global scaling layers, the scaling constants are fixed by design, leading to zero trainable weights in those layers. The global scaling layer at the top adjusts the scales of the input variables (x and t) in the range of -1 to $+1$ with respect to L and τ. The scaled inputs are denoted as $\tilde{x}$ and $\tilde{t}$, respectively. Meanwhile, the global scaling layer at the bottom linearly increases the output with a proportionality constant equal to the characteristic deflection scale ω_{Ref}.

The hidden layers between the two global scaling layers are constructed based on STD. The hidden layers consist of two parallel structures that reflect STD, denoted as $G(\tilde{x}|\ \theta)$ and

$H(\tilde{t}|\ \theta)$. Each decomposed structure has two fully connected layers with 10 neurons each at the top. The $x_j + \sin^2 x_j$ activation is employed with a local scaling layer wx_j. The proportional constant in the local scaling layer w is determined by training. The outcomes of these fully connected layers are combined using an additional fully connected layer without activation, which produces single numeric valued outcomes $g(\tilde{x})$ and $h(\tilde{t})$ for structure G and H, respectively. The $g(\tilde{x})$ and $h(\tilde{t})$ are merged by multiplication, resulting in $\tilde{\omega}(\tilde{x}, \tilde{t}) = g(\tilde{x})h(\tilde{t})$. Therefore, the final prediction can be expressed as $\omega(x,t) = \omega_{\text{Ref}}\tilde{\omega}$ from the global scaling layer at the bottom. The constructed STD-PINN has a total of 286 trainable weights, which is orders of magnitude less than regular PINN architectures that have $O(10^3)$ to $O(10^4)$ trainable weights [1].

The collocation points were randomly sampled from the simulation domain, with 300, 300, and 5000 points selected for ICs, BCs, and PDE losses, respectively. The training process began with the Adam optimizer [34], using a learning rate of 10^{-2} for 200 epochs, which successfully initiated the learning procedure. Afterwards, the optimizer was switched to the L-BFGS-G algorithm using the Scipy implementation [35, 36]

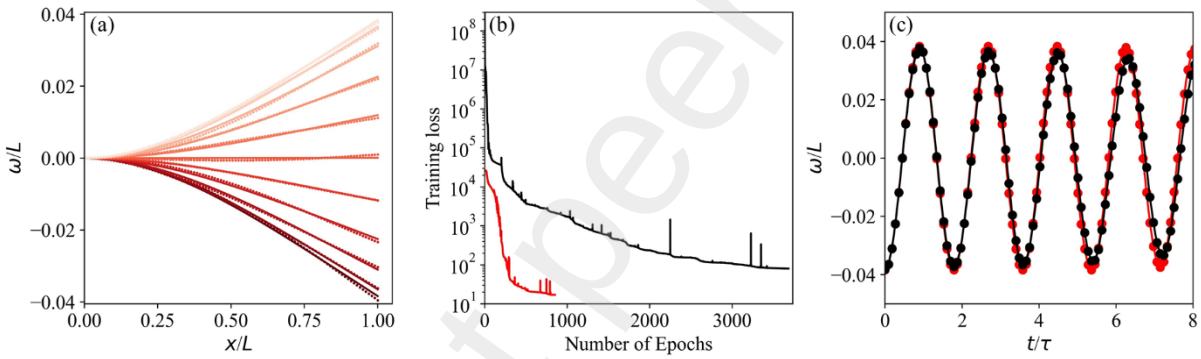

Figure 2. (a) Displacement profiles of the free vibration of a cantilevered beam over half-cycle of the first-bending mode from the truncated solution (dots) and STD-PINN (solid lines). (b) The trend in training loss over the number of epochs for model 1 (black) and model 2 (red). (c) The trajectory of the beam ends predicted by model 1 (black) and model 2 (red).

Table 1. Training errors of the regular PINN as a baseline, the model 1 configured from the scaling law, and the model 2 realized with the STD-PINN and the scaling law.

	$L1$ absolute error	$L2$ relative error
Regular PINN	7.56×10^{-3}	7.20×10^{0}
Model 1 (Scaling law)	1.48×10^{-4}	1.78×10^{-1}
Model 2 (STD with Scaling law)	3.05×10^{-5}	2.93×10^{-2}

3.1.4. Results

The performance of the proposed STD-PINN is examined by comparison to the regular PINN used as a baseline. The regular PINN is constructed with fully connected layers with

tanh activation, where three hidden layers of 11 neurons each are adopted, resulting in a similar size to the STD-PINN, with a total of 309 trainable weights. The global scaling layer is also adopted at the top, which adjusts the input variables to range from -1 to +1, identical to those of the present STD-PINN and the original PINN model [1]. However, the configuration based on scaling law and the resulting global scaling layer at the bottom are not adopted for the baseline model. The loss function is constructed based on equation (2).

Therefore, the stepwise improvements from the regular PINN can be suggested from the strategies proposed in this study as 1) the configuration based on the scaling law with the regular PINN architecture and 2) STD-PINN in addition to the scaling law, which termed as model 1 and model 2, respectively, in the following. The predictions of the PINN models (baseline, model 1, and model 2) are evaluated compared to the truncated semi-analytic solution from beam bending theory, which accounts for up to 5th bending modes. The semi-analytic solution is realized based on the Sympy library [37].

Table 1 shows the $L1$ absolute and $L2$ relative errors of the PINN models. The errors are calculated for 5 cycles of oscillation in terms of the period of the 1st bending mode. As mentioned earlier, the regular PINN was not able to find a reliable solution for the problem of free vibration of the beam. The $L1$ and $L2$ errors of the regular PINN are $7.56 \times 10^{-3}$and 7.20×10^{0}, respectively. As a remedy, the proposed strategies greatly improved the predictions of the PINN models, as shown in the errors of model 1 and model 2. The predictions of model 2 (the STD-PINN) exhibit approximately 10-fold smaller errors compared to model 1 (the regular PINN with the scaling law).

Figure 2(a) illustrates the beam displacement profiles obtained from the STD-PINN (solid lines) compared to those of the truncated solution (dots), where 10 consecutive snapshots are shown from the initial end load condition for half a cycle of the first bending mode. The predictions of the STD-PINN method exhibit good agreement with the semi-analytic solution. Note that the figure is drawn by amplifying the y-scale to clearly show the beam oscillation. The training procedure and predicted solutions of model 1 and model 2 are further compared in figures 2(b) and (c). The trend in training loss is plotted according to the training epochs in figure 2(b), where the black and red lines indicate model 1 and model 2, respectively. Model 1 (only with scaling law configuration) exhibits much slower convergence to an order-of-magnitude greater training loss compared to model 2 (STD-PINN extension). The trajectories of the beam end are illustrated in Fig. 2(c). The predictions of model 1 and 2 are almost identical from the start of the oscillation to the first cycle. However, the oscillation amplitude predicted from the regular PINN architecture attenuates gradually, indicating limited accuracy in describing the beam oscillation.

3.2. Flutter of a slender cantilever in axial flow

3.2.1. Modeling

The fluid flow-induced flutter of a cantilever is considered, where the transverse load $q(x, t)$ in equation (5) originates from the coupled interaction with fluid flow. Assuming that the cantilevered beam is slender ($L \gg l$), the aerodynamic force exerted on the cantilevered beam can be modeled using Lighthill's elongated body theory [38]. The transverse load is approximated as a reaction force to the rate of change of the fluid momentum, $-m_a(Dv/Dt)$,

where m_a is the added mass and v is the transverse fluid velocity. The added mass of the slender plate can be modeled as $\pi\rho_f l^2/4$, where ρ_f is the fluid density [39].

The transverse fluid velocity v can be described by the kinematic boundary condition on the cantilevered beam, which is expressed as $v = \partial\omega/\partial t + U\partial\omega/\partial x$, based on the small-perturbation assumption with respect to the incoming fluid velocity U. Integrating all the equations, the transverse load $q(x,t)$ is expressed as a second-order differential equation of the deflection of the beam ω:

$$q(x,t) = -\frac{\pi\rho_f l^2}{4}\left(\frac{\partial^2\omega}{\partial t^2} + 2U\frac{\partial^2\omega}{\partial x\,\partial t} + U^2\frac{\partial^2\omega}{\partial x^2}\right). \quad (7)$$

Thus, the entire system is described by the coupled equations of fluid and structure, which correspond to the fourth-order differential equation from the dynamic beam bending equation (5) and the second-order differential equation from the elongated-body theory (7).

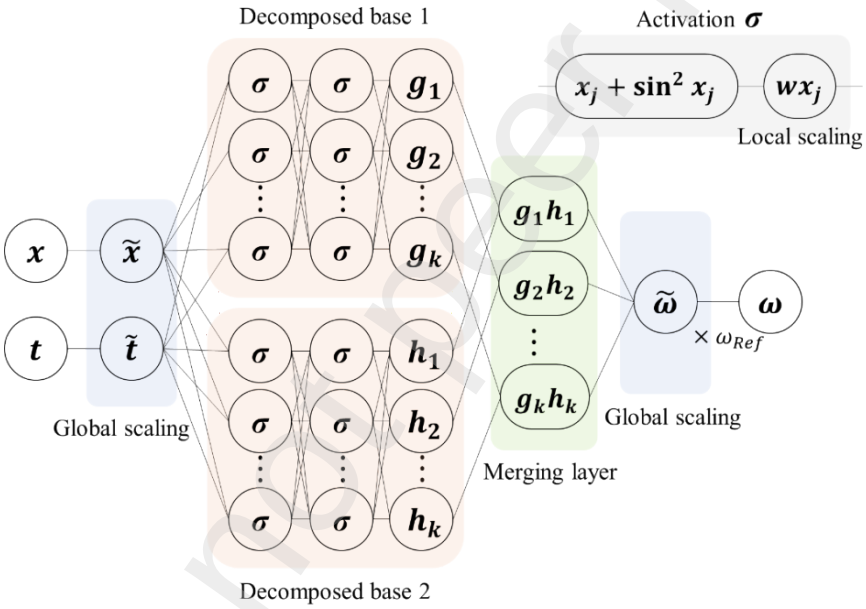

Figure 3. Neural networks architecture adopting 2 decomposed bases to resolve fluid induced flutter.

3.2.2. Configuration

Fluid-induced flutter of a cantilever arises above a certain critical velocity as the fluid dynamic pressure excites an oscillatory mode [40]. The characteristic timescale is then redefined by considering the balance of the fluid dynamic force and inertia as follows. The fluid force per unit length can be expressed as $\frac{1}{2}\rho_f U^2 l\frac{\omega_{\mathrm{Ref}}}{L}$, where ρ_f is the fluid density, U is the incoming fluid velocity, l is the width of the cantilever, and $\frac{\omega_{\mathrm{Ref}}}{L}$, is the scale of a tilted angle. Equating the inertia force $\mu\frac{\omega_{\mathrm{Ref}}}{\tau^2}$ with the fluid force $\frac{1}{2}\rho_f U^2 l\frac{\omega_{\mathrm{Ref}}}{L}$,, the characteristic timescale for fluid-induced flutter is obtained as $\tau \sim \left(\frac{2\mu L}{\rho_f U^2 l}\right)^{1/2}$. The losses in equation (6) are

then reconfigured with the modified timescale, which leads to scale variations in in $\tilde{\mathcal{L}}_{IC,2}$ and $\tilde{\mathcal{L}}_{PDE}$. Furthermore, to improve the convergence, we employ the elaboration of relative tolerances in $\varepsilon_{IC,1} = 10^{-4}$, in addition to $\varepsilon_{BC,1} = 10^{-4}$. The rest of the relative tolerances are determined as $\varepsilon_{IC} = \varepsilon_{BC} = 10^{-3}$ and $\varepsilon_{PDE} = 10^{-2}$.

3.2.3. Neural networks architecture

The PINN architecture is constructed as a sequence of the following layers, as schematically illustrated in Figure 3: the global scaling layer at the top, which adjusts the input to the range of -1 to 1; two parallel structures reflecting decomposed bases; a merging layer; and the final global linear scaling with ω_{Ref}. Whereas the global scaling layers are identical to those used for the free vibration problem, the parallel structures in the middle are designed to take both scaled space and time, in contrast to STD in the free vibration problem. Therefore, the parallel structures in the middle can be denoted as $G(\tilde{x}, \tilde{t}|\ \theta)$ and $H(\tilde{x}, \tilde{t}|\ \theta)$. Fully connected neural networks composed of two hidden layers and 20 neurons are employed for each of the parallel structures with $x + \sin^2 x$ activation. The outcomes of each structure are merged using elementwise multiplication as $g_i(\tilde{x}, \tilde{t})h_i(\tilde{x}, \tilde{t})$, where i corresponds to the number of hidden neurons, which is 20. The outputs are superimposed to produce a single value, $\tilde{\omega}$, using a fully connected layer without activation, leading to $\tilde{\omega} = \sum_i W_i g_i(\tilde{x}, \tilde{t})h_i(\tilde{x}, \tilde{t}) + b$, where W_i represents the weights and b represents the bias. The superimposed value is linearly scaled at the last layer to produce the final prediction as $\omega_{\text{Ref}}\tilde{\omega}$.

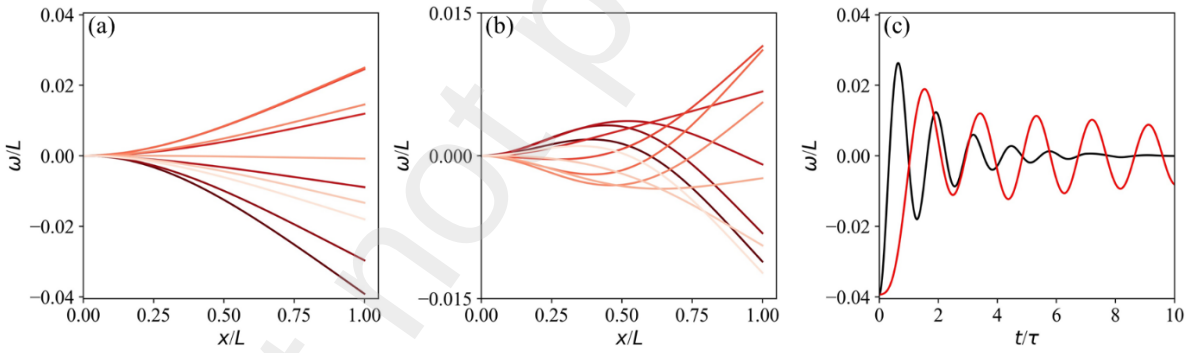

Figure 4. Displacement profiles of the vibration of a cantilevered beam in an axial flow of (a) 1m/s and (b) 6 m/s, which correspond to the $\tilde{\boldsymbol{U}}$ of 2.6 and 15.8, respectively, at the $\tilde{\boldsymbol{m}}$ of 0.14. (c) The trajectories of the beam end, where the black and red lines correspond the axial flow of 1 m/s and 6 m/s, respectively.

3.2.4. Results

The following experiment tested whether the PINN could accurately describe the sub-critical behavior of a cantilevered plate in axial flow [41, 42]. Two different conditions were studied, corresponding to the stable and fluttering modes at a dimensionless velocity of $\tilde{U} = \sqrt{\mu/BUL}$ of around 2.6 and 15.8, respectively, at the given dimensionless mass of $\tilde{m} = \rho_f Ll/\mu$ of around 0.14. It is important to note that the critical dimensionless velocity of flutter is expected to be around 15, based on Eloy et al. (2007). Below the critical velocity, the free vibration of a cantilever would be attenuated by stabilizing fluid dynamic pressure. On the

other hand, beyond the critical velocity, the incoming fluid flow would destabilize the cantilever, leading to fluid-induced flutter with an excitation of higher bending modes [43, 44].

Figure 4 exhibits the predictions of the proposed PINN model for the stable mode (Figure 4a) and the fluttering mode (Figure 4b). The critical behavior of the fluttering cantilevered beam is successfully captured by the proposed PINN model. In Figure 4a, snapshots over one cycle are shown starting from the release of the initial end load condition, with ten consecutive deflection profiles depicted in chronological order from dark to light red. The amplitude of displacement significantly decays as the fluid dynamic pressure acts like a damping force. The displacement curve of the cantilever end for prolonged cycles is shown as a black line in Figure 4c, clearly demonstrating that the fluid dynamic force stabilizes the vibration of the cantilever by gradually reducing the amplitude of the oscillation.

As the incoming fluid velocity increases, the emergence of fluttering behavior is predicted by the PINN model. Whereas the higher-order bending mode appears in the deflection profiles of the cantilever, as shown in Figure 4(b), the domination of the first bending mode is observed in both the free vibration (Figure 2b) and the stable mode (Figure 4a). The predicted profile corresponds to a single-neck flutter, widely observed in previous numerical simulations [40, 41] and experiments [42-44] in flag flutter. The displacement profile of the cantilever end, depicted as a red line in Figure 4c, also confirms the appearance of periodic flutter in the PINN prediction. It is remarkable that the PINN could capture the emergence of the critical behavior of the cantilever in axial flow by resolving coupled differential equations (5) and (7). Further considerations of inextensibility, large amplitude oscillation, and the integration of lift-driven and resistive forces could be the next challenge for the PINN in relevant areas.

3.3. 1-D advection in a high velocity

3.3.1. Modeling

The 1-D advection equation is one of the simplest equations in physics, typically used to describe wave propagation with constant velocity c, as shown below:

$$\frac{\partial u(x,t)}{\partial t} + c\frac{\partial u(x,t)}{\partial x} = 0 \tag{8}$$

Krishnapriyan et al. (2021) proposed that the regular PINN fails to obtain a reliable solution for the 1-D advection equation in high-velocity conditions, where $c > 10$ in a given domain of $x \in [0, 2\pi)$ and $t \in [0, 1]$. This observation has prompted the use of a new training strategy, termed curriculum training, in that paper, which repeatedly trains the PINN by gradually increasing the wave speed c. In this experiment, the strategies proposed in this study, including scaling analysis for configuration and alternative activation, are tested by resolving the extremely high-velocity condition of $c = 30$. The solution curve is found in the same domain with the initial condition corresponding to $u(x,0) = \sin(x)$ and periodic boundary conditions as $u(0,t) = u(2\pi,\mathrm{t})$.

3.3.2. Configuration

The characteristic time scale of the 1-D advection problem can be expressed as $\tau \sim \frac{L}{c}$

$= \frac{2\pi}{30}$, with L being the spatial domain size. This experiment has used a reference time scale of 0.2 with five time-marching steps. Then, the PINN constructs the solution curve within the time domain of interest, $t \in [0, 1]$. The characteristic scale of the wave amplitude $u(x,t)$ can be considered as $O(1)$ based on the initial condition, $u(x,0) = \sin(x)$. The PDE error is then scaled as $E_{PDE} \sim \varepsilon_{PDE} \frac{\partial u(x,t)}{\partial t} \sim \varepsilon_{PDE} \frac{u}{\tau} \sim \frac{\varepsilon_{PDE}}{\tau}$. The errors of IC and BCs correspond to the scales of desired tolerances, with $E_{IC} \sim \varepsilon_{IC} u \sim \varepsilon_{IC}$ and $E_{BC} \sim \varepsilon_{BC} u \sim \varepsilon_{BC}$.. Finally, the losses can be scaled with the square of the error, as follows: $L_{PDE} \sim (\frac{\varepsilon_{PDE}}{\tau})^2$, $L_{IC} \sim {\varepsilon_{IC}}^2$ and $L_{BC} \sim {\varepsilon_{BC}}^2$. The relative tolerances are selected as $\varepsilon_{IC} = \varepsilon_{BC} = 10^{-3}$ and $\varepsilon_{PDE} = 10^{-2}$.

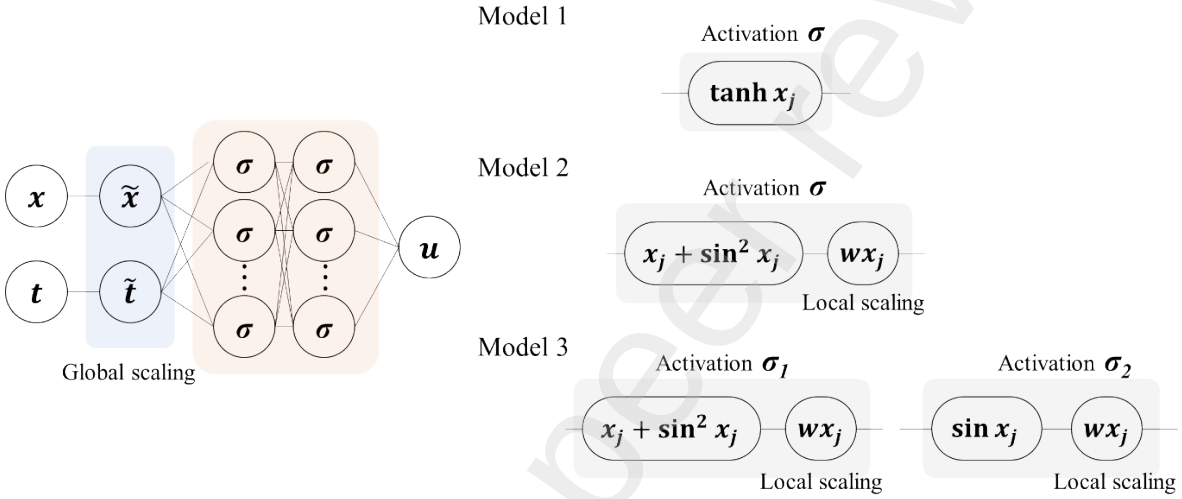

Figure 5. The neural networks architecture adopted to resolve 1D advection problem. The model 1, 2 and 3 employed different activation functions with the same neural networks architecture and scaling configuration.

Table 2. Training errors of the tested PINN architectures for 1D advection problem.

	$L1$ absolute error	$L2$ relative error	Ref
Regular PINN	5.42×10^{-1}	8.87×10^{-1}	Krishnapriyan et al. (2021)
Curriculum Training	1.10×10^{-2}	2.02×10^{-2}	Krishnapriyan et al. (2021)
Model 1 (Scaling law)	2.57×10^{-3}	4.48×10^{-3}	Present
Model 2 (Scaling law with $x + \sin^2 x$ activation)	1.29×10^{-3}	2.38×10^{-3}	Present
Model 3 (Scaling law with $x + \sin^2 x$ in hidden layer, $\sin x$ at the last layer)	1.00×10^{-3}	1.76×10^{-3}	Present

3.3.3. Neural Networks

This experiment examines three stepwise enhancements from the regular PINN, which are referred to as models 1 to 3. All models utilized the scaling law configuration described earlier, as well as an identical PINN architecture shown in Figure 5, except for the activation function. The architecture consisted of fully connected neural networks with two hidden layers, each containing 20 neurons. Notably, the global scaling layer was applied only at the top, and the final layer produced a scalar output through a dense layer without a global scaling layer.

First, model 1 used the tanh activation function. The only difference between the regular PINN and model 1 was the scaling law configuration. Next, model 2 replaced the tanh activation with the $x + \sin^2 x$ activation function and added a linear scaling layer after the activation layer. Finally, the prior knowledge of the solution curve of the 1-D advection problem inspired the use of an alternative activation function. It is expected that the 1-D advection equation has a traveling wave solution from the initial sinusoidal profile. Therefore, the model 3 was constructed with two hidden layers of different activations, the $x + \sin^2 x$ activation at the top and the $\sin x$ activation at the bottom, as shown in Figure 5. As a result, the PINN models had a 2×20×20×1 structure, leading to 501 trainable weights for model 1 and 503 trainable weights for models 2 and 3 by adding two more weights in the local scaling layer.

3.3.4. Results

Table 2 exhibits $L1$ absolute and $L2$ relative errors of the proposed PINN models, where the prediction errors are compared with the baseline results of Krishnapriyan et al. (2021). The experiments suggest that the proposed strategy could dramatically elaborate the PINN solution. The prediction errors are improved by an order-of-magnitude: from 2-digit to 3-digit relative errors. It is noted that the 2-digit error of the curriculum training strategy is already a tenfold improvement from the regular PINN [14]. Furthermore, the proposed modifications from model 1 to model 3 are shown to gradually improve PINN solutions. The best prediction error of model 3 suggest that the given knowledge of the mathematical expression of the solution curve can be used to guide the neural networks architecture.

4. Conclusion

This study systematically investigated the strategies to construct PINNs to resolve several extreme problems in mechanics, where the availability of the PINN solutions was previously limited. The study first resolved the free vibration of a cantilevered beam, which involves fourth-order differentials in space and second-order differentials in time. Second, the fluid-induced flutter of a cantilevered cylinder was considered based on the coupled dynamic equations from Euler-Bernoulli beam bending theory and Lighthill's elongated body theory. Lastly, the 1D advection problem in extremely high velocity conditions was examined.

The novel strategies proposed in this study aided in resolving extreme mechanics problems. Firstly, the configuration of the PINN was found using scaling analysis, which considers relative scales of the physical variables in terms of their characteristic scales. This approach helped balance the relative importance of constraints from PDEs, BCs, and ICs, resulting in significant improvements in both the PINN predictions and training procedures. Secondly, an alternative activation function to tanh was employed as $x_j + \sin^2(x_j)$activation, followed by linear scaling layer wx_j. This alternative activation function helps to convey second-order non-linearity of the input variable, which enables to capture the periodic nature

of the concerned phenomena. Lastly, the mathematical interpretation of the given problem guided the PINN architecture. For example, by constructing neural network architecture as a combination of parallels to reflect the decomposed bases of the desired solution, PINN could resolve a number of extreme problems physically interpreted from modal behaviors.

The strategies proposed in this study are expected to provide crucial guidelines to construct PINNs. Furthermore, the proposed strategies only consider the guided configuration and architecture designs. As such, they can be incorporated with recent approaches to improve PINN, including enhanced gradient flows [12], concerning causality [10, 13], parallelization [18], and collocation points sampling [20]. By incorporating the strategies proposed in this study with recent advancements, the potential of PINNs to resolve complex problems across various scientific domains can be further enhanced. Therefore, the proposed strategies are expected to greatly expand the scope and applicability of PINNs in tackling a wide range of challenging problems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code and trained results for examples are available at https://github.com/JeongsLee/

Supplementary Information

Regular PINN for the free vibration of a cantilever

This experiment examined the performance of a regular PINN in resolving the free vibration of a cantilever. The architecture of the PINN is the same as that used in the previous experimental section, consisting of three hidden layers of 11 neurons each and tanh activation. However, the scaling law was not adopted for the PINN configuration. The total loss, $\mathcal{L}_{tot}$, was constructed based on equation (2). The optimal hyperparameters of α_{PDE}, α_{IC} and α_{BC} were searched from the set $\{10^7, 10^8, 10^9\}$, with the central order of 10^8 obtained from $\frac{\tau^4}{(\varepsilon_{PDE,1}\mu\omega_{\mathrm{Ref}})^2} \sim O(10^8)$. Table S1 summarizes the $L1$ absolute and $L2$ relative errors for the total of 27 experiments. It should be noted that the errors were calculated for a reference time step of $\tau \sim (\frac{\mu}{B}L^4)^{1/2}$, in contrast to the previous experiments where errors were calculated for 5 cycles of the first-bending mode. The best-performing hyperparameter configuration corresponds $[\alpha_{BC}, \alpha_{PDE}, \alpha_{IC}] = [10^7, 10^7, 10^7]$, resulting in of $L1$ absolute and $L2$ relative errors of 7.79×10^{-4} and 6.68×10^{-1}, respectively. This is actually orders of magnitude larger than the results based on the proposed strategies, implying that the regular PINN was unable to accurately capture the physical variations from high-order differentiations.

Table S1. Training errors of the regular PINN over various hyperparameters

Hyperparameters $[\alpha_{BC}, \alpha_{PDE}, \alpha_{IC}]$	$L1$ absolute error	$L2$ relative error
$[10^7, 10^7, 10^7]$	7.79×10^{-4}	6.68×10^{-1}
$[10^7, 10^7, 10^8]$	1.66×10^{-3}	1.44×10^{0}
$[10^7, 10^7, 10^9]$	2.59×10^{-3}	2.28×10^{0}
$[10^7, 10^8, 10^7]$	1.31×10^{-3}	1.15×10^{0}
$[10^7, 10^8, 10^8]$	2.56×10^{-3}	2.16×10^{0}
$[10^7, 10^8, 10^9]$	1.11×10^{-3}	9.88×10^{-1}
$[10^7, 10^9, 10^7]$	1.24×10^{-3}	1.11×10^{0}
$[10^7, 10^9, 10^8]$	1.61×10^{-3}	1.34×10^{0}
$[10^7, 10^9, 10^9]$	1.63×10^{-3}	1.38×10^{0}
$[10^8, 10^7, 10^7]$	3.31×10^{-3}	2.62×10^{0}
$[10^8, 10^7, 10^8]$	2.64×10^{-3}	2.33×10^{0}
$[10^8, 10^7, 10^9]$	5.34×10^{-3}	5.41×10^{0}
$[10^8, 10^8, 10^7]$	1.02×10^{-3}	9.00×10^{-1}
$[10^8, 10^8, 10^8]$	1.16×10^{-3}	1.04×10^{0}
$[10^8, 10^8, 10^9]$	1.93×10^{-3}	1.61×10^{0}
$[10^8, 10^9, 10^7]$	1.35×10^{-3}	1.21×10^{0}
$[10^8, 10^9, 10^8]$	1.27×10^{-3}	1.11×10^{0}
$[10^8, 10^9, 10^9]$	2.03×10^{-3}	1.70×10^{0}
$[10^9, 10^7, 10^7]$	1.05×10^{-3}	9.05×10^{-1}
$[10^9, 10^7, 10^8]$	4.48×10^{-2}	3.12×10^{1}
$[10^9, 10^7, 10^9]$	2.20×10^{-2}	1.84×10^{1}
$[10^9, 10^8, 10^7]$	2.04×10^{-2}	1.67×10^{1}

$[10^9, 10^8, 10^8]$	9.09×10^{-4}	8.40×10^{-1}
$[10^9, 10^8, 10^9]$	1.53×10^{-3}	1.43×10^{0}
$[10^9, 10^9, 10^7]$	1.46×10^{-3}	1.33×10^{0}
$[10^9, 10^9, 10^8]$	1.51×10^{-3}	1.27×10^{0}
$[10^9, 10^9, 10^9]$	1.13×10^{-3}	1.04×10^{0}

References

[1] M. Raissi, P. Perdikaris, 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. 378 (2019) 686-707.

[2] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112789.

[3] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks for heat transfer problems, J. Heat Transfer 143(6) (2021) 060801.

[4] W. Peng, J. Zhang, W. Zhou, X. Zhao, W. Yao, X. Chen, IDRLnet: A physics-informed neural network library, arXiv preprint arXiv:2107.04320.

[5] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367(6481) (2020) 1026-1030.

[6] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. 43(6) (2021) B1105-B1132.

[7] E. Kharazmi, D. Fan, Z. Wang, M.S. Triantafyllou, Inferring vortex induced vibrations of flexible cylinders using physics-informed neural networks, J. Fluids Struct. 107 (2021) 103367.

[8] H. Gao, M.J. Zahr, J.X. Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Engrg. 390 (2022) 114502.

[9] Y. Kim, Y. Choi, D. Widemann, T. Zohdi, A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder, J. Comput. Phys. 451 (2022) 110841.

[10] R. Mojgani, M. Balajewicz, P. Hassanzadeh, Kolmogorov n-width and Lagrangian physics-informed neural networks: A causality-conforming manifold for convection-dominated PDEs, Comput. Methods Appl. Mech. Engrg. 404 (2023) 115810.

[11] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, J. Comput. Phys. 449 (2022) 110768.

[12] J. Yu, L. Lu, X. Meng, G.E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Methods Appl. Mech. Eng. 393 (2022) 114823.

[13] S. Wang, S. Sankaran, P. Perdikaris, Respecting causality is all you need for training physics-informed neural networks, arXiv preprint arXiv:2203.07404.

[14] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M.W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 26548-26560.

[15] A. Bihlo, R.O. Popovych, Physics-informed neural networks for the shallow-water equations on the sphere, J. Comput. Phys. 456 (2022) 111024.

[16] K. Haitsiukevich, A. Ilin, Improved Training of Physics-Informed Neural Networks with Model Ensembles, arXiv preprint arXiv:2204.05108.

[17] B. Moseley, A. Markham, T. Nissen-Meyer, Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, arXiv preprint arXiv:2107.07871.

[18] K. Shukla, A.D. Jagtap, G.E. Karniadakis, Parallel physics-informed neural networks via domain decomposition, J. Comput. Phys. 447 (2021) 110683.

[19] R. Leiteritz, D. Pflüger, How to Avoid Trivial Solutions in Physics-Informed Neural Networks, arXiv preprint arXiv:2112.05620.

[20] A. Daw, J. Bu, S. Wang, P. Perdikaris, A. Karpatne, Rethinking the importance of sampling in physics-informed neural networks, arXiv preprint arXiv:2207.02338.

[21] F.G. Yuan, S.A. Zargar, Q. Chen, S. Wang, Machine learning for structural health monitoring: challenges and opportunities, Sensors Smart Struct. Technol. Civil, Mech. Aerospace Syst. 2020, 11379 (2020) 1137903.

[22] Q. Chen, Physics Informed Learning for Dynamic Modeling of Beam Structures, [Master's thesis, North Carolina State University] (2020).

[23] Z. Zhang, G.X. Gu, Physics-informed deep learning for digital materials, Theor. Appl. Mech. Lett. 11 (2021) 100220.

[24] N. Sukumar, A. Srivastava, Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks, Comput. Methods Appl. Mech. Eng. 389 (2022) 114333.

[25] D. Katsikis, A. D. Muradova, G. E. Stavroulakis, A Gentle Introduction to Physics-Informed Neural Networks, with Applications in Static Rod and Beam Problems, J. Adv. Appl. Comput. Math. 9 (2022) 103-128.

[26] J. Bai, T. Rabczuk, A. Gupta, L. Alzubaidi, Y. Gu, A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics,

Comput. Mech. (2022) 1-20.

[27] T. Szandała, Review and comparison of commonly used activation functions for deep neural networks, in: Bio-inspired neurocomputing, Springer, Singapore, 2021, pp. 203-224.

[28] Z. Li, T. Hartwig, M. Ueda, Neural networks fail to learn periodic functions and how to fix it, in: Advances in Neural Information Processing Systems, 2020, pp. 1583-1594.

[29] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, in: Advances in Neural Information Processing Systems, 2020, pp. 7462-7473.

[30] S. Qian, H. Liu, C. Liu, S. Wu, H. San Wong, Adaptive activation functions in convolutional neural networks, Neurocomputing 272 (2018) 204-212.

[31] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.

[32] M. Khan, M. Hussain, Application of Laplace decomposition method on semi-infinite domain, Numer. Algorithms 56 (2) (2011) 211-218.

[33] G.P. Engworo, P.O. Ogunniyi, S.O. Edeki, Laplace decomposition method for series solutions of systems of linear partial differential models, J. Phys.: Conf. Ser. 2199 (1) (2022) 012022.

[34] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[35] C. Zhu, R.H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization, ACM Trans. Math. Softw. 23 (4) (1997) 550-560.

[36] P. Virtanen, R. Gommers, T.E. Oliphant et al., SciPy 1.0: fundamental algorithms for scientific computing in Python, Nat. Methods 17(3) (2020) 261-272.

[37] A. Meurer, C.P. Smith, M. Paprocki et al., SymPy: symbolic computing in Python, PeerJ Comput. Sci. 3 (2017) e103.

[38] M. J. Lighthill, Aquatic animal propulsion of high hydromechanical efficiency, J. Fluid Mech. 44(2) (1970) 265-301.

[39] M. J. Lighthill, Note on the swimming of slender fish, J. Fluid Mech. 9(2) (1960) 305-317.

[40] L. Tang, M. P. Paidoussis, On the instability and the post-critical behaviour of two-dimensional cantilevered flexible plates in axial flow, J. Sound Vib. 305(1-2) (2007) 97-115.

[41] C. Eloy, C. Souilliez, L. Schouveiler, Flutter of a rectangular plate, J. Fluids Struct. 23(6) (2007) 904-919.

[42] S. Taneda, Waving motions of flags, J. Phys. Soc. Jpn. 24(2) (1968) 392-401.

[43] C. Eloy, R. Lagrange, C. Souilliez, L. Schouveiler, Aeroelastic instability of cantilevered

flexible plates in uniform flow, J. Fluid Mech. 611 (2008) 97-106.

[44] H. Kato, M. Watanabe, Three-dimensional nonlinear analysis and wind-tunnel experiment of flutter generated on a rectangular sheet in uniform flow, J. Fluids Struct. 101 (2021) 103226.