Hybrid acceleration techniques for the physics-informed neural networks: a comparative analysis

Fedor Buzaev, Jiexing Gao*, Ivan Chuprov, Evgeniy Kazakov

Moscow Research Center, 2012 Labs, Huawei Technologies Co., Ltd.,
Smolenskaya square 7-9, Moscow, 119121, Russia.

*Corresponding author(s). E-mail(s): gaojiexing@huawei.com;

Abstract

Physics-informed neural networks (PINN) has emerged as a promising approach for solving partial differential equations (PDEs). However, the training process for PINN can be computationally expensive, limiting its practical applications. To address this issue, we investigate several acceleration techniques for PINN that combine Fourier neural operators, separable PINN, and first-order PINN. We also propose novel acceleration techniques based on second-order PINN and Koopman neural operators. We evaluate the efficiency of these techniques on various PDEs, and our results show that the hybrid models can provide much more accurate results than classical PINN under time constraints for the training, making PINN a more viable option for practical applications. The proposed methodology in the manuscript is generic and can be extended on a larger set of problems including inverse problems

Keywords: Physics-informed neural networks, sinusoidal learning space, Fourier neural operators, Koopman neural operators

1 Introduction

Physics-informed neural networks (PINNs) belong to universal function approximators that are trained by taking into account the underlying physical laws during the learning process, and in this way, provide us a robust framework to make predictions bounded by the physical laws [1]. The main application of PINN is to solve partial

differential equations (PDEs) [2]. In this regard PINNs are often considered as a mesh-free alternative to traditional numerical PDE solvers [3]. Essentially, PINN is able to solve a PDE in the weak formulation, i.e. by minimizing the loss function which indicates how well the neural network satisfies the PDE. Usually the loss function is taken as the residual of the PDE and its boundary conditions. The minimization is performed by tuning the weights of the neural network. In order to compute the PDE residual, the partial derivatives of the neural network have to be computed, which can be done by using the automatic differentiation technique. The PINN methodology was rediscovered in [4] and after that was applied in a wide range of mathematical physics problems including computational fluid dynamics [5], electrical engineering [6], radiative transfer [7], nano-optics [8], heat transfer [9] etc. An extensive list of references describing PINN applications for solving PDEs can be found in [10]. Due to the growing interest to this topic, several frameworks dedicated to PINN have been introduced (there are over 400 projects related to physics-informed machine learning on Github), including that from NVIDIA [11] and Microsoft [12]. However, there are some concerns about the mathematical justification of this approach. In particular, minimization of the loss function in PINN is a highly non-convex optimization problem, wherein convergence to the global minimum cannot be guaranteed [13]. The structure of the neural network has to be adjusted to a PDE (and possibly, to parameters of a PDE). Sometimes, the efficiency of PINN does not depend smoothly on the number of layers, neurons and other characteristics of the neural network structure. That complicates the design of the optimal PINN structure for a given problem.

The main practical question is whether PINNs can be more effective than traditional solvers, such as the finite element method (FEM). Potentially, PINN can be faster and more efficient than FEM in some cases, because PINN can learn to approximate the solution to a PDE directly, without the need for discretization or mesh generation, which can be time-consuming and computationally intensive steps in the FEM process. In the recent work of Grossmann et al [14] it was shown that in terms of solution time and accuracy, PINNs were not able to outperform FEM. A similar conclusion was done in [3]. It was shown that low-frequency components of the solution converged quickly, while in contrast, an accurate solution of the high frequencies required an exceedingly long time. Such feature of PINN can be regarded as an implication of the F-principle in deep learning [15], that says that neural networks tend to fit the data by a low-frequency function. The main argument in defense of PINN is that, unlike traditional solvers, PINNs can invoke free parameters of the PDEs as extra inputs and in this way to be trained for a class of PDEs rather than for a single PDE. For instance, such an approach was applied to the nonlinear Schrödinger equation [16]. The launch power of the signal is embedded in PINN as a parametric feature, which makes PINN learn the signal transmissions under different powers simultaneously. There are dedicated frameworks based on PINN which are specifically designed to incorporate a large amount of parametric features, e.g. PI-DeepONet [17]. Since the training of PINN is performed offline and the resulting network is computationally very fast, the long training time is not an issue. The second argument in favor of PINN is based on the so-called transfer learning approach [18–20], wherein

a formerly trained PINN is reused as a starting point in a training procedure for a novel problem thereby accelerating the convergence.

In practice, the PINN training is a time-consuming procedure. In this regard, new methods for accelerating the PINN training are required. Recently several concepts involving novel architectures and approaches were proposed to improve the performance and efficiency of PINNs. Moreover, in some papers authors showed the cases where the classical PINN did not converge at all, while using proposed modifications it was possible to get an accurate solution [21].

Usually the acceleration methods for PINN are studied in isolation of each other and compared only against PINN based on a fully-connected neural network. The goal of this paper is to analyze the efficiency of the combined use of several acceleration techniques.

2 Overview of PINNs

PINNs approximate PDE solutions using a deep neural network. The basic features of PINN can be summarized as follows. Consider a PDE in the following form:

$$\mathfrak{F}(U(\boldsymbol{x})) = 0 \quad \text{for } \boldsymbol{x} \in D, \tag{1}$$

where $\mathfrak{F}$ is the differential operator, U is the solution to the PDE, while $\boldsymbol{x} = \{x_1, ..., x_n\} \in D$ is n-dimensional vector of coordinates belonging to the domain D. Operator $\mathfrak{F}$ may involve high-order partial derivatives of U over x_i. The domain D is bounded by Γ. Function U is a subject to boundary conditions (here we do not distinguish boundary and initial conditions),

$$\mathfrak{B}(U(\boldsymbol{x})) = 0 \quad \text{for } \boldsymbol{x} \in \Gamma, \tag{2}$$

where $\mathfrak{B}$ is the boundary operator. Our goal is to find the vector $\boldsymbol{b}$ incorporating the parameters of the neural network, so that PINN satisfies Eqs. (1) and (2). The problem is formulated in a weak form:

$$\boldsymbol{b} = \arg\min_{\boldsymbol{b}} \left[L_{\text{pde}} + L_{\text{bc}} \right], \tag{3}$$

where

$$L_{\text{pde}} = \sum_{i=1}^{N} \left[\mathfrak{F}(\text{PINN}(\boldsymbol{x}_i, \boldsymbol{b})) \right]^2 \quad \text{for } \boldsymbol{x}_i \in D, \tag{4}$$

$$L_{\text{bc}} = \sum_{i=1}^{M} \left[\mathfrak{B}(\text{PINN}(\boldsymbol{x}_i, \boldsymbol{b})) \right]^2 \quad \text{for } \boldsymbol{x}_i \in \Gamma, \tag{5}$$

while N and M are the number of sampling points within D and at Γ, respectively. The partial derivatives of PINN with respect to the elements of $\boldsymbol{x}$ in Eq. (4) can be computed on the basis of automatic differentiation tools implemented in machine learning libraries.

The schematic of PINN is shown in Figure 1.

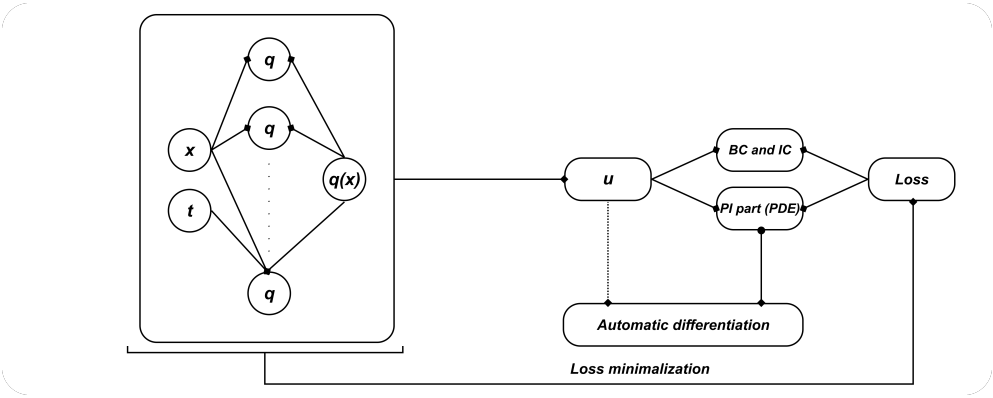

Fig. 1 Schematic of PINN, where the loss function of PINN comprises a mismatch in the boundary and initial conditions, as well as the residual for the PDE on a set of points in the chosen domain

The performance of PINN is strongly affected by the topology/architecture of the neural network, including the number of layers, the number of neurons in each layer, and the connections between neurons. In general, a more complex topology with more layers and more neurons can potentially improve the performance of PINN, but it also requires more training time and so, provide less accurate results under training time constraints.

3 Acceleration methods

Due to the large number of parameters that can be changed in the PINN model, such as the number of layers, number of neurons, optimizer, etc., the accurate analysis of acceleration methods for PINN is a challenging task. We have therefore chosen methods that we believe to be fundamentally different, but which nevertheless show high efficiency when applied individually. For the purpose of our analysis, we identify two groups of acceleration methods. The first group are techniques that are a modification of the way the data is taken from the deep learning block. From this group we consider separable PINN (SPINN), FO-PINN and SO-PINN. In addition, we implement first-order SPINN (FO-SPINN) and second-order SPINN (SO-SPINN) as hybrid models alongside them. The second group of methods can be attributed to the functional spaces. Essentially, it deals with what inside the deep learning block. From this group we consider PINN with sinusoidal activation function, PINN with Fourier neural operators (FNOs), PINN with U-Net architecture and, as a hybrid model, PINN with the sinusoidal activation function and FNO blocks. Subsequently, several hybrid PINN models are considered by implementing pairwise methods from two groups. Below is a brief discussion of each of these methods. Note that the distinction between two groups of methods is not strict, and some methods can be attributed to both categories. In our classification, we have mostly relied on the context of papers where these methods were originally described.

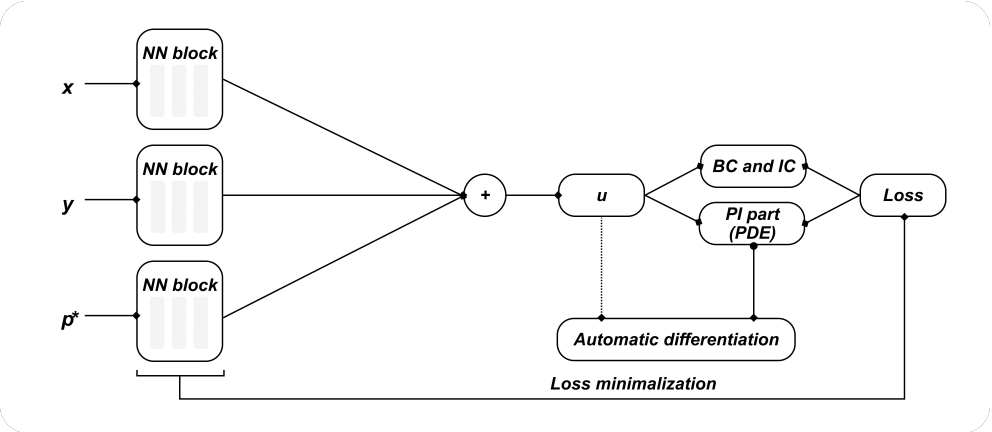

Fig. 2 Schematic of Separable PINN (SPINN), wherein the solution is decomposed into a product of functions that depend only on a single variable

3.1 Separable PINN

SPINNs [22] are a type of PINN that use a separable representation of the solution, where the solution is decomposed into a product of functions that depend only on a single variable. This approach can significantly reduce the computational cost of solving PDEs and can also improve the accuracy of the solution. SPINNs have been shown to be particularly effective for solving PDEs with separable solutions, such as certain types of elliptic and parabolic PDEs. In this regard, SPINN can be partially thought of as a representative of the second group methods. The schematic of the SPINN architecture is shown in Figure 2.

In particular, for an output function depending on n variables, SPINN consists of n sub-networks, each of them depends on a single coordinate. The final output is taken as a dot product of the corresponding outputs of the sub-networks. This approach reduces the number of propagation across the network during the learning process, while the complexity of the problem scales linearly of the number of dimensions unlike classical architectures with exponential scaling.

3.2 First-order PINN

It was observed that computation of partial derivative terms in the PINN loss functions via automatic differentiation during training are computationally expensive. In this regard, several methods were proposed in order to reduce the use of automatic differentiation. In particular, in the so-called FO-PINN [23] the neural network has additional outputs which approximate the partial derivatives, while in [24] derivatives are estimated by using meshless radial basis function-finite differences.

While in the classical PINN, the output of the neural network approximates the PDE solution and the loss function is estimated w.r.t. the function to be found, in FO-PINN or in FO-SPINN there are additional outputs ($\mathrm{PINN}_{\mathrm{d}}(\boldsymbol{x}_i)$) to approximate the first-order partial derivatives. Consequently, the loss function incorporates an extra

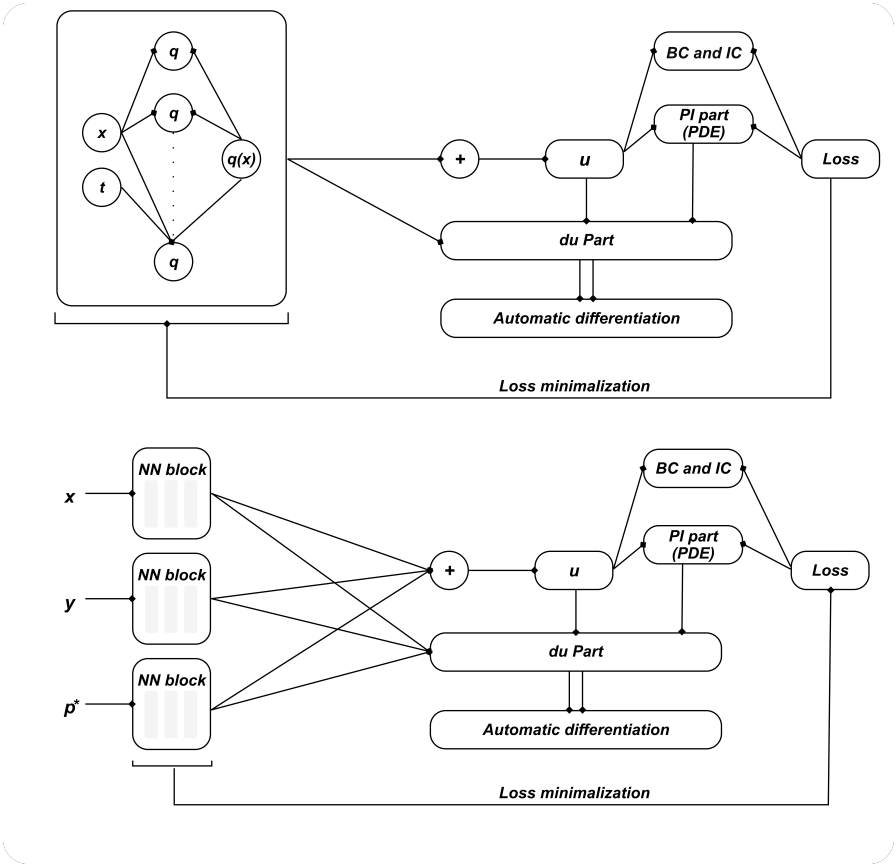

Fig. 3 Schematic of First-order PINN (FO-PINN) and first-order-separable PINN (FOS-PINN)

term L_{d} showing how accurate the approximation of derivatives as compared to those computed by using the automatic differentiation:

Note that the first-order PINN can be implemented for SPINN architectures. Such hybrid schemes are also analysed in this paper.

$$L_{\mathrm{d}} = \sum_{i=1}^{N} \left[\mathrm{PINN}_{\mathrm{d}}(\boldsymbol{x}_i) - \mathrm{autograd}(\mathrm{PINN}(\boldsymbol{x}_i), \boldsymbol{x}_i)\right]^2 . \tag{6}$$

The ideas of FO-PINN and FOS-PINN are illustrated in Figure 3. Note that the first-order PINN can be implemented for SPINN architectures. Such hybrid schemes are also analysed in this paper.

3.3 Second-order PINN

SO-PINN is a logical extension of FO-PINN. In our approach, we integrate an additional neural network that generates both first and second order derivatives. This means we have two neural networks operating concurrently. They are both trained

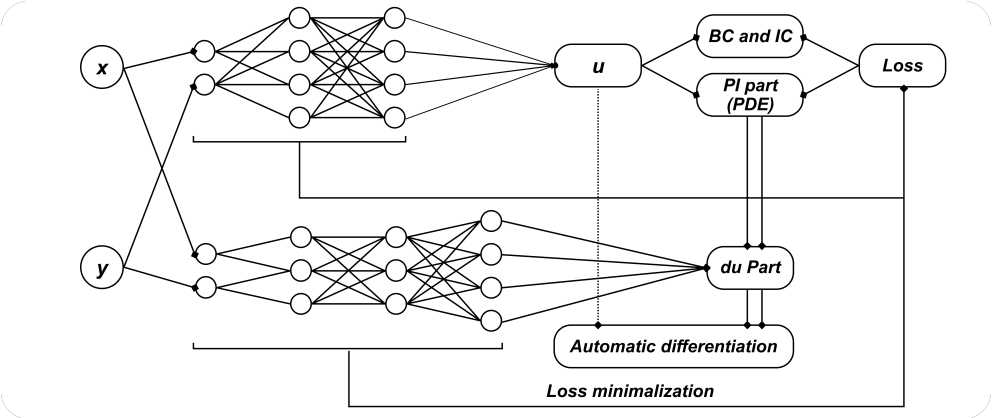

Fig. 4 Schematic of second-order PINN (SO-PINN)

with the same data batches, while the loss function to be minimized incorporates the loss terms of both networks. Importantly, we merge the two neural networks into a single graph, allowing for backpropagation through the entire network.

The architecture is illustrated in Figure 4. The first-order derivative loss term is the residual between the output for the first derivative of the second neural network and the output of the first neural network followed by automatic differentiation. In SO-PINN, the second-order derivative loss term is calculated as a residual between the corresponding outputs of the second neural network and the automatically differentiated output of the second neural network corresponding to the first-order derivative. In this paper, we adopt the following configuration of SO-PINN: 6 linear layers with layer shape is equal $[n_{\text{inputs}}, 200, 500, 500, 200, n_{\text{outputs}}]$, where n_{inputs} is a number of input coordinates shape,n_{outputs} is a number of output derivatives shape. We used GELU non-linearity as function of the activation.

3.4 Fourier Neural Operator and Koopman Neural Operator

FNO were introduced in [25] and designed to perform a mapping between infinite-dimensional function spaces [26]. Initially, FNO was designed to parametrize a set of PDE solutions using the training dataset. Mathematical properties of such neural operators are considered in [27]. Essentially, a neural operator can be represented as a neural network consisting of a stack of L layers:

$$G = Q \circ (W_L + K_L) \circ ... \circ \sigma(W_1 + K_1) \circ P, \tag{7}$$

where G is a neural operator, σ is the activation function, W_i is the point-wise linear operator, K_i is the integral kernel operator, while P and Q are the pointwise neural networks that encode the lower dimension function into higher dimensional space and decode the higher dimensional space into the lower dimensional space, respectively. In FNO, K_i is taken as a convolution operator, and the fast Fourier transform F is used to compute K_i. Thus,

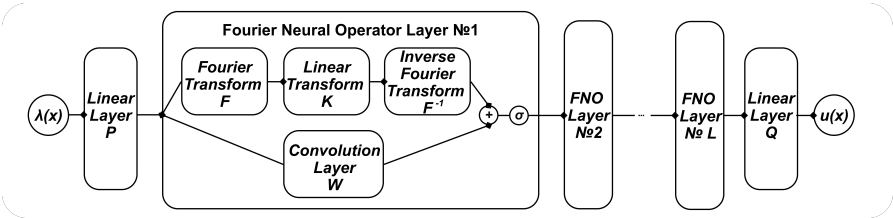

Fig. 5 Schematic of FNO

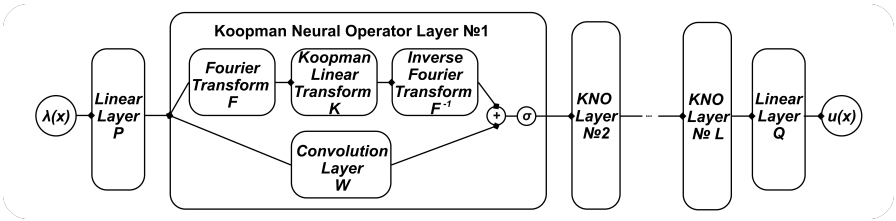

Fig. 6 Schematic of the Koopman neural operator (KNO)

$$(Kv)(x) = F^{-1}(r(Fv))(x), \tag{8}$$

where r represents the parameter to be learned. Its schematic representation is depicted in Figure 5. The use of FNO in PINN is described in [28] and [27]. The corresponding model is referred to as physics-informed neural operators (PINO). In [27] PINO was up to 2 orders of magnitude faster as compared to the classical solvers. However, the PINO error level could not be reduced below 1 % (or only perhaps at the cost of drastic increase of the training time). In the meantime, the classical solvers have no problem to achieve higher level of accuracy.

Koopman Neural Operator (KNO) was introduced in [29]. As the FNO, KNO was initially designed to parametrize a set of PDE solutions using the training dataset. Specifically, KNO takes as input a set of measurements or observations of the system at different times, and produces as output a set of predictions of the system's behaviour at future times. As described in [29], KNO is expressed as a composition of simpler operators that can be more easily approximated by a neural network, e.g. as a composition of nonlinear observables of the system, followed by a linear operator that maps the observables to their future values. KNO includes the following elements: an encoder of the input data, Fourier transform (similar to FNO), a linear layer (corresponding to a Hankel matrix), and inverse Fourier transform. On top of that, a convolutional layer is applied to the output of the encoder in order to extract high-frequency components of the solution. Then the outputs of the inverse Fourier transform and high frequency components are summed up, and the decoder is applied to the sum. A scheme of KNO is given in Figure 6.

Similar to FNO, KNO can also be integrated into the PINN framework. However, incorporating FNO and KNO into PINN is not a straight-forward process. Instead of treating a group of points as a single sample for training, in PINN each point can be treated as a separate sample. This fact raises the crucial question of which quantity

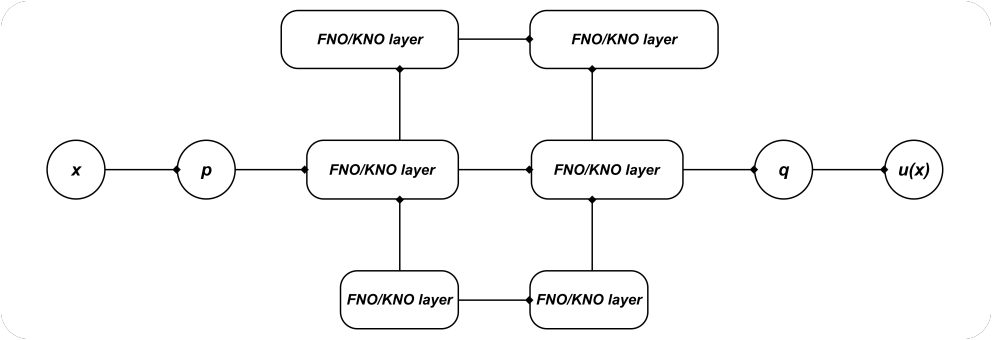

Fig. 7 Schematic of a U-Net with FNO/KNO layers

the Fourier transform should be applied to. To solve this obstacle, and to work with mesh-free data, input data should fit to the convolution layers. For that purpose, we artificially increase the data dimension and perform embeddings through two linear layers to transform the data dimension from $[n_{\text{points}}, n_{\text{dim}}]$ to $[n_{\text{points}}, \text{width}, \text{width}]$, where n_{dim} is a dimension of input data. For outputs we use two linear layers for inverse transformation from $[n_{\text{points}}, \text{width}, \text{width}]$ to $[n_{\text{points}}, n_{\text{dim}}]$. That approach helps to find additional patterns in the mesh-free data.

3.5 U-Net

U-Net is a type of neural network architecture that was originally developed for the task of image segmentation [30], which involves identifying and separating different objects within an image. The name U-Net comes from the shape of the network, which resembles the letter U. The U-Net architecture is characterized by a series of convolutional layers that gradually reduce the spatial resolution of the input image, followed by a series of upsampling layers that gradually restore the resolution back to the original size. The network also includes skip connections, which allow information from earlier layers to be passed directly to later layers, bypassing the intermediate layers. This helps the network to better capture both low-level and high-level features of the image, which is particularly important for image segmentation tasks.

To solve partial differential equations, the authors propose to use FNO layers instead of usual convolution layers [31]. We have also complemented this architecture with KNO layers. Its schematic representation of U-Net with FNO layers (or KNO layers) is shown in Figure 7.

3.6 Sinusoidal activation function

In [32] it was shown that PINNs have the so-called spectral bias that prevents them from learning high-frequency functions. There it was also proposed to add to neural network special blocks which mimic the solution features in the Fourier space. A similar idea was also discussed in [33]. In [34] it was proposed to use as the sinusoidal activation function for the output of the first layer, which looks as follows:

$$\sigma(x) = \sin(2\pi(x + b)), \tag{9}$$

where b is the bias. The argument in favor of this approach takes into account the process of the PINN convergence. Namely, it was shown that whenever the activation function is relatively flat (e.g. ReLU or tanh), PINN tends to provide a solution that can be very close to satisfying many PDEs and falling into a local minimum of the PINN loss that only minimizes PDE residuals. As a consequence, it takes more time for optimizers to minimize also other terms of the loss function thereby increasing the total training time. It was also shown empirically that the sinusoidal mapping of inputs prevents the loss function to reach a local minimum and be 'trapped' there instead of converging to a global minimum.

4 Simulations

All simulations are performed on Nvidia V100 GPU with 16GB memory. To make comparison consistent, we limit the training time by 10 minutes. The default configuration of PINN consists of 8 residual linear layers with 300 neurons in each layer. FNO consists of 3 blocks incorporating 4 linear layers and taking 32 modes followed by the GELU activation functions. We consider two models based on SPINN. The first model of SPINN has 4 linear layers with 300 neurons in each layer followed by the tahnshrink activation functions. The second model of SPINN consists of 4 linear layers with 32 neurons in each layer followed by FNO. The models of FO-PINN and FO-SPINN have the same configuration like models of PINN and SPINN. The GELU activation functions are employed in this case. We use AdamW optimizer to train for 10 minutes with an initial learning rate of 0.001. The ReduceLROnPlateau scheduler multiplies the learning rate by 0.9 if the loss is not decreased after 50 consecutive epochs. The weights of the network are initialized by using the Xavier method [35]. The ensure consistency, a total of 2500 sampling points are chosen for all cases. These points are randomly and uniformly distributed across the domain, and remain fixed throughout the training process without any changes.

4.1 Poisson equation

The Poisson equation is a type of elliptic partial differential equation that is widely used in theoretical physics. The solution of the Poisson equation is the potential field caused by a given charge density distribution. Under a known potential field, an electrostatic field can be calculated.

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u(x, y) = f(x, y), \tag{10}$$

where u is the unknown function, and f is the source function that reads as follows:

$$f(x, y) = (1 - x^2)(2y^3 - 3y^2 + 1). \tag{11}$$

Results of experiments are shown in Table 1. The best result is marked in bold. Almost all cases of training with sinusoidal activation the final mean squared error will be smaller than without the use of the sinusoidal activation function.

Table 1 MSE for the Poisson equation

	Classic	+sin	+FNO	+FNO+sin	+KNO	+KNO+sin
	3.1e-01	1.2e-01	4.6e-05	3.7e-05	5.0e-05	1.5e-05
FO	1.9e-02	9.2e-05	2.3e-05	4.5e-06	2.7e-05	5.6e-06
SO	1.2e-03	5.9e-05	1.0e-04	6.1e-05	7.3e-05	7.2e-05
Separable	7.5e-01	1.5e-01	5.6e-06	6.8e-07	1.6e-06	**5.5e-07**
Separable+FO	9.0e-02	1.0e-01	9.9e-02	2.2e-01	9.6e-02	2.2e-01
Separable+SO	1.6e-04	1.5e-04	7.1e-05	4.8e-05	8.5e-05	6.4e-05
U-Net	-	-	5.5e-04	9.1e-05	4.2e-04	1.4e-04
U-Net+FO	-	-	2.3e-04	1.6e-04	1.0e-03	7.0e-05
U-Net+SO	-	-	8.8e-05	2.7e-04	2.0e-04	1.3e-04
Separable+U-Net	-	-	1.3e-03	1.0e-04	2.3e-04	7.4e-05
Separable+U-Net+FO	-	-	7.5e-02	1.0e-01	8.2e-02	1.3e-01
Separable+U-Net+SO	-	-	1.0e-03	4.9e-04	1.4e-03	3.2e-04

Table 2 MSE for PINN solutions of the reaction-diffusion equation

	Classic	+sin	+FNO	+FNO+sin	+KNO	+KNO+sin
	6.1e-01	4.1e-01	3.9e-01	3.8e-01	1.3e-04	1.3e-04
FO	3.9e-01	1.1e-03	3.6e-04	2.8e-04	4.0e-04	2.6e-04
SO	4.8e-01	4.5e-01	3.3e-01	2.7e-01	3.3e-03	2.1e-02
Separable	2.3e+00	4.0e-01	1.5e-04	1.4e-04	1.4e-04	1.3e-04
Separable+FO	2.6e+00	4.0e-01	4.1e-01	4.1e-01	4.1e-01	4.1e-01
Separable+SO	1.7e+01	1.5e+00	1.3e-03	6.8e-02	5.5e-04	3.9e-03
U-Net	-	-	4.2e-04	3.3e-01	**6.7e-05**	2.0e-04
U-Net+FO	-	-	3.8e-01	4.6e-04	3.7e-01	3.0e-01
U-Net+SO	-	-	2.9e-02	9.3e-02	1.0e-01	3.8e-02
Separable+U-Net	-	-	5.1e-04	3.1e-04	2.8e-01	2.5e-03
Separable+U-Net+FO	-	-	4.5e+00	3.9e-01	2.1e+00	3.9e-01
Separable+U-Net+SO	-	-	6.4e-02	1.9e-01	6.4e-02	8.5e-02

4.2 Reaction-diffusion equation

The reaction-diffusion equation is a mathematical equation describing how two chemicals might react to each other as they diffuse through a medium together.

$$\frac{\partial u}{\partial t} + \nu \frac{\partial^2 u}{\partial x^2} = \rho u(1-u), \tag{12}$$

where $\rho = 5$, $\nu = 3$ and u are the PDE parameters and the source function, respectively.

The results are shown in Table 2 . The best result is marked in bold. The combined use of SPINN with the first- or second-order modifications reduced the accuracy as compared to the classic PINN. The best configuration is the U-Net with KNO blocks. The classic PINN involving KNO blocks also shows good results outperforming the classic PINN and other methods implemented separately. FNO without the use of additional techniques converges to the minimum best.

4.3 Helmholtz equation

The Helmholtz equation is one of the varieties of PDE. That equation represents time-independent form of source's wave in time-independent domain.

$$\nabla^2 u = -k^2 f. \tag{13}$$

For our tests, we consider

$$-u_{xx} - u_{yy} - k_0^2 u = f(x, y), \tag{14}$$

where k and f are the wave number and source of elliptic PDE that is widely used in theoretical physics. The solution of the Helmholtz is the potential field caused by a given charge density distribution. Under a known potential field, an electrostatic field can be calculated.

The corresponding source functions are given by

$$f(x, y) = \sin(k_0 x) \sin(k_0 y) \tag{15}$$

in the domain $\Omega = [0, 1]^2$, with the Dirichlet boundary conditions $u(x, y) = 0$, $(x, y) \in \partial\Omega$. The value of k_0 can be $4\pi, 16\pi$ or 24π. The distribution with different k_0 is shown in Figure 8. As k_0 increases, the solution becomes more complex in shape. That causes difficulties for neural networks according to the F-principle [15], that says that neural networks tend to fit the data by a low-frequency function.

The MSE values for all cases are shown in Tables 3, 4 and 5. The best result is marked in bold. The best configuration for $k_0 = 4\pi$ appeared to be SPINN with the first-order modification including FNO blocks and the sinusoidal activation function. The separate use of these options, as well as KNO blocks leads to less accurate results as those from the classic PINN. For $k_0 = 16\pi$ the accuracy of the classic PINN decreases, while the configurations involving KNO blocks become accurate. In particular, SPINN with the first-order modification, KNO blocks and sinusoidal activation function shows the best performance. Finally, for $k_0 = 24\pi$ SPINN with the second-order modification, KNO blocks and sinusoidal activation function is the winner outperforming by far other the rest configurations. As modifications of PINN introduce additional overhead and the training time is limited by 10 minutes, different configurations, different configurations manage to go through a different number of learning epochs. Note that the overhead changes with k_0. For instance, for $k_0 = 4\pi$ the classic PINN is the fastest, while for $k_0 = 16\pi$ and $k_0 = 24\pi$ PINN with the second-order modification and that with KNO blocks and sinusoidal activation function, respectively, are the fastest ones. The separate use of SPINN does not improve the results as compared to the classic PINN.

4.4 Burgers' equation

Burgers' equation arises in various areas of applied mathematics, including fluid mechanics, nonlinear acoustics and gas dynamics. It is a fundamental PDE which

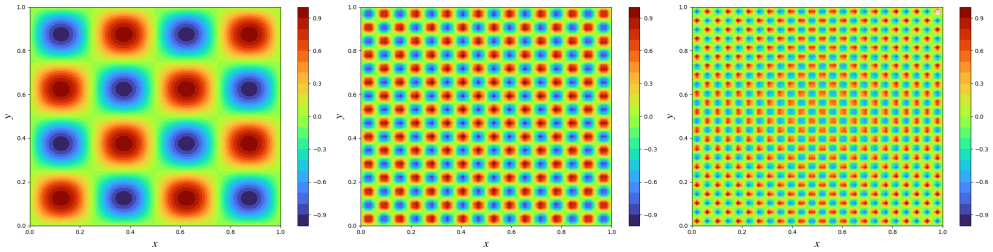

Fig. 8 Examples of Helmholtz equation with the source terms given by Eq. (15) with $k_0 = 4\pi$(left), $k_0 = 16\pi$ (middle) and $k_0 = 24\pi$ (right)

Table 3 MSE for the Helmholtz equation with $k_0 = 4\pi$

	Classic	+sin	+FNO	+FNO+sin	+KNO	+KNO+sin
	1.0e-04	1.2e-02	5.9e-02	8.9e-02	1.0e-01	1.0e-01
FO	6.2e-02	7.1e-02	6.4e-03	1.3e-04	5.7e-02	1.0e-01
SO	2.3e-02	4.4e-02	4.9e-03	2.7e-02	5.0e-02	4.6e-02
Separable	2.1e-01	3.7e-02	2.2e-02	7.1e-02	1.1e-01	1.0e-01
Separable+FO	1.8e-01	8.6e-04	7.3e-04	**3.5e-05**	1.8e-04	7.8e-04
Separable+SO	2.2e-01	1.3e-02	2.3e-01	8.5e-03	1.5e-02	4.4e-02
U-Net	-	-	1.5e-01	1.1e-01	1.0e-01	1.0e-01
U-Net+FO	-	-	6.2e-02	7.7e-02	6.4e-02	7.7e-02
U-Net+SO	-	-	1.7e-03	3.3e-04	3.2e-03	9.0e-03
Separable+U-Net	-	-	1.0e-01	9.7e-02	1.0e-01	9.8e-02
Separable+U-Net+FO	-	-	1.7e-01	8.0e-04	4.1e-02	1.1e-03
Separable+U-Net+SO	-	-	1.0e-03	2.1e-04	1.5e-03	7.5e-04

Table 4 The same as in Table 3, but for $k_0 = 16\pi$

	Classic	+sin	+FNO	+FNO+sin	+KNO	+KNO+sin
	1.4e-01	1.9e-02	2.1e-01	1.1e-01	1.8e-01	1.4e-01
FO	7.4e-02	2.0e-02	2.0e-01	5.1e-02	1.3e-01	3.4e-02
SO	1.5e-01	2.2e-02	1.5e-01	4.4e-02	1.5e-01	4.0e-02
Separable	2.5e-01	2.5e-01	2.4e-01	1.8e-01	1.1e-01	1.0e-01
Separable+FO	2.5e-01	2.5e-01	2.4e-01	3.4e-04	6.5e-02	**3.5e-05**
Separable+SO	3.5e-01	2.6e-01	2.4e-01	4.1e-04	2.2e-03	4.2e-05
U-Net	-	-	2.4e-01	1.4e-01	2.1e-01	1.5e-01
U-Net+FO	-	-	7.3e-02	3.0e-02	6.6e-02	2.4e-02
U-Net+SO	-	-	1.1e-01	2.0e-02	2.0e-01	5.4e-02
Separable+U-Net	-	-	2.4e-01	1.0e-01	2.4e-01	1.0e-01
Separable+U-Net+FO	-	-	2.8e-01	2.4e-01	2.4e-01	2.4e-01
Separable+U-Net+SO	-	-	2.4e-01	3.9e-03	2.4e-01	2.4e-01

describes the dynamics of viscous fluids or gases. It is a simplified version of the Navier-Stokes equation and can be derived from it by dropping the pressure gradient term. For our tests, we consider the following equation:

$$u_t + uu_x = (0.01/\pi)u_{xx} \tag{16}$$

Table 5 The same as in Table 3, but for $k_0 = 24\pi$

	Classic	+sin	+FNO	+FNO+sin	+KNO	+KNO+sin
	2.6e-01	1.6e-01	2.4e-01	2.1e-01	2.2e-01	2.6e-01
FO	2.4e-01	1.7e-01	2.6e-01	1.8e-01	2.4e-01	1.9e-01
SO	2.6e-01	1.8e-01	2.5e-01	1.4e-01	2.6e-01	2.1e-01
Separable	2.6e-01	2.7e-01	2.4e-01	2.4e-01	1.9e-01	1.0e-01
Separable+FO	2.6e-01	2.6e-01	2.4e-01	2.1e-01	5.4e-02	3.7e-04
Separable+SO	2.8e-01	2.6e-01	2.4e-01	2.8e-04	1.5e-01	**1.2e-04**
U-Net	-	-	2.3e-01	2.0e-01	2.1e-01	1.9e-01
U-Net+FO	-	-	2.7e-01	1.4e-01	2.6e-01	1.8e-01
U-Net+SO	-	-	2.4e-01	1.2e-01	2.0e-01	1.3e-01
Separable+U-Net	-	-	2.4e-01	1.1e-01	2.4e-01	1.0e-01
Separable+U-Net+FO	-	-	2.4e-01	2.8e-01	2.5e-01	2.5e-01
Separable+U-Net+SO	-	-	1.2e-01	4.6e-03	2.4e-01	3.0e-03

Table 6 MSE of Burgers' equation solution

	Classic	+sin	+FNO	+FNO+sin	+KNO	+KNO+sin
	1.1e-01	9.3e-02	2.7e-02	2.1e-02	**8.1e-03**	2.9e-02
FO	9.3e-02	8.0e-02	1.5e-01	9.7e-02	4.3e-02	3.0e-02
SO	1.5e-01	2.9e-01	1.6e-01	9.0e-02	8.6e-02	8.6e-02
Separable	1.7e-01	2.7e-01	1.9e-01	1.0e-01	6.0e-02	2.9e-02
Separable+FO	3.7e-01	3.7e-01	2.8e-01	2.3e-01	3.7e-01	3.7e-01
Separable+SO	1.2e-01	1.1e-01	1.3e-01	1.1e-01	9.0e-02	8.4e-02
U-Net	-	-	4.4e-02	2.3e-01	1.1e-01	3.3e-02
U-Net+FO	-	-	9.5e-02	8.1e-02	9.5e-02	8.0e-02
U-Net+SO	-	-	1.0e-01	1.0e-01	1.0e-01	1.0e-01
Separable+U-Net	-	-	4.0e-02	3.3e-02	3.6e-02	3.2e-02
Separable+U-Net+FO	-	-	3.8e-01	3.7e-01	3.7e-01	3.7e-01
Separable+U-Net+SO	-	-	1.3e-01	1.3e-01	1.2e-01	1.3e-01

in domain $x \in [0,1]^2$, $t \in [0,1]^2$with initial conditions $u(0,x) = -\sin(\pi x)$, and Dirichlet boundary conditions $u(t,-1) = u(t,1) = 0$.

The MSE values are summarized in Table 6. The best result is marked in bold. The configuration with KNO blocks appears to be the best one, while in combination with SPINN the second-order modification is more accurate than that with the first-order modification. An application of SPINN leads to worse results than those provided by the classic PINN.

5 Discussion

We tested several configurations of PINN and found that there was no clear winner in the comparison. For example, in the case of the Poisson equation, SPINN equipped with KNO and sinusoidal activation function performed the best, while using FO-PINN led to significantly worse results. However, this configuration was outperformed by the U-Net PINN with KNO blocks in the case of the reaction-diffusion equation. In fact, none of the options that were considered consistently resulted in an enhancement of PINN. Furthermore, we could not formulate explicitly the causes behind the success or failure of each individual option. The amount of overhead associated with utilizing

Table 7 MSE of Burgers' equation solution

Equation	Mesh	Solution time, s	MSE
Reaction-Diffusion	50x50	0.2	1e-4
Poisson	50x50	0.03	1e-7
Helmholtz, $k = 4\pi$	50x50	1.2	3e-11
Helmholtz, $k = 16\pi$	50x50	1.2	1e-6
Helmholtz, $k = 24\pi$	50x50	1.2	1
Helmholtz, $k = 24\pi$	2500x2500	1080 (18 min)	3e-6

various PINN options varies among different problems. This fact adds complexity to the examination of PINN configurations and the process of finding the optimal PINN configuration.

PINN is considered as a possible competitor of traditional solvers, such as FEM. For this reason, we apply FEM to aforementioned problems. We use DOLFINx [36] framework on python. DOLFINx framework uses combination of a Krylov subspace method[37] and a preconditioner from PETSc package [38]. In all simulations, the LU preconditioner with 1e-6 absolute and relative tolerances is used. Computations are performed on 1 core of Intel Xeon Gold 6151 3.0 GHz CPU. The results are summarized in Table 7. The corresponding MSE values are computed with respect to the analytical solutions. For considered cases, FEM provides stable results with MSE below 10^{-4} taking the same number of grid points, as PINN, namely 2500. The accuracy of PINN solutions becomes significantly worse when the number of sampling points in PINN is decreased. Let us recall that PINN takes 10 minutes on GPU. From this perspective and taking into account that our PINNs have around 10^5 trainable parameters against 10^3 unknowns in FEM, using piecewise linear functions as basic functions in FEM appears more effective than representing the solution of PDE through neural networks.

As the wavenumber in the Helmholtz equation gets higher, the number of required grid points in FEM also increases significantly. For example, when $k = 24\pi$, the number of points needs to be increased by a factor of 10^3. In this scenario, the performance of both FEM and PINN (in their most optimal configurations) can be deemed comparable, with a slight advantage leaning towards FEM. In this context, it is reasonable to assume that using PINN as a PDE solver is effective for cases that require fine discretization. However, for other cases, FEM appears to be a better option, as the solution can be obtained using fewer grid points (finite elements). This conclusion is only valid when considering a solution to a single PDE. In the case of complex problems like weather forecasting, it is reasonable to anticipate that PINN would offer significantly faster result predictions than numerical methods, with an accuracy that surpasses that of the numerical approach [39]. Moreover, in [40] it is suggested that the efficiency of PINN approaches increase with the dimension of the problem, whereas in FEM the complexity increases exponentially with increasing dimensions.

We finalize this section by mentioning that although this paper does not cover extensive numerical experiments necessary for comparing PINN and FEM, we present illustrative examples to provide readers with a sense of how these two methods compare to each other

6 Summary

In this paper, we explored several acceleration techniques for PINNs including noval PINN archtectures based on Koopman neural operators and the second-order PINNs. in order to improve their performance and efficiency. We have considered several types of methods, including those based on advanced network architectures, calculating derivatives by neural networks instead of solely use of automatic differentiation approach, and learning in functional spaces. These approaches have been implemented in a common framework aiming to streamline the process of developing and fine-tuning PINN models and provide an all-in-one solution for efficient PINN research. The best configuration for a specific problem can be found by performing tests across all available architectures.

In majority of considered cases SPINN architecture with sinusoidal activation function perform well and robust. To reduce the overhead due to automatic differentiation, FO-PINNs have been considered. We extended this method to include the second-order derivatives. Our results showed that the efficiency of SO-PINN was almost as good as FO-PINN in most cases. Nevertheless, SO-PINN performed better in the problems with high oscillatory solutions, such as the Helmholtz equation. We explored the use of functional spaces, including sinusoidal activation functions and FNO. In addition, we implemented PINNs based on KNO. Through our simulations, the combined use of neural operators together with the sinusoidal activation function may improve the accuracy of PINN. In most experiments, KNOs were more efficient than FNOs. The optimized configurations of PINN provided results with the error by 3-4 orders of magnitude less than the original PINN based on fully-connected layers.

Overall, our results suggest that the optimal configuration of PINN depends on the specific physics problem being addressed, and there is no single approach that works best for all cases. Nevertheless, it was shown that the combined use of several performance enhancement techniques can significantly improve PINN results. In particular, the SPINN configuration with sinusoidal activation function and KNO blocks seems to be a configuration to be tested. Our study highlights the potential benefits of combining multiple techniques to achieve the best results.

The most efficient configurations of PINN found in this study could not over-perform FEM. For instance, while PINN required minutes of training on GPU to obtain an accurate solution, FEM was capable to achieve accurate results in several seconds. Our experiments with the Helmholtz equation revealed that the performance gap between PINN and FEM diminishes as problem complexity increases. In our study, we specifically refer to the highly oscillatory nature of the solution as the complexity factor. Previous research suggests that a similar trend may be observed as the problem dimension increases. In our future work we aim to explore the potential of PINNs in solving complex problems that traditional solvers may struggle with. Specifically, we will focus on high-dimensional problems and parametric PDEs with free parameters.

7 Declarations

Jiexing Gao supervised the project. Fedor Buzaev and Jiexing Gao conceived the original idea. Fedor Buzaev and Ivan Chuprov designed the model and the

computational framework and analysed the data. Evgeniy proposed the FEM experiment in discussions.

All authors declare that they have no conflicts of interest.

Funding - Not applicable

Ethics approval - Not applicable

Code availability - Not applicable

Consent to participate - Not applicable

Consent for publication - Not applicable

Availability of data and material - Not applicable

References

[1] Kollmannsberger, S., D'Angella, D., Jokeit, M., Herrmann, L.: Physics-informed neural networks. In: Deep Learning in Computational Mechanics vol. 977, pp. 55–84. Springer, (2021). https://doi.org/10.1007/978-3-030-76587-3_5

[2] Berg, J., Nyström, K.: A unified deep artificial neural network approach to partial differential equations in complex geometries (2017) https://doi.org/10.1016/j.neucom.2018.06.056 arXiv:1711.06464

[3] Markidis, S.: The old and the new: Can physics-informed deep-learning replace traditional linear solvers? Frontiers in Big Data **4** (2021) https://doi.org/10.3389/fdata.2021.669097

[4] Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics **378**, 686–707 (2019)

[5] Wu, P., Pan, K., Ji, L., Gong, S., Feng, W., Yuan, W., Pain, C.: Navier–stokes generative adversarial network: a physics-informed deep learning model for fluid flow generation. Neural Computing and Applications **34**(14), 11539–11552 (2022) https://doi.org/10.1007/s00521-022-07042-6

[6] Xu, Z., Guo, Y., Saleh, J.H.: A physics-informed dynamic deep autoencoder for accurate state-of-health prediction of lithium-ion battery. Neural Computing and Applications **34**(18), 15997–16017 (2022) https://doi.org/10.1007/s00521-022-07291-5

[7] Mishra, S., Molinaro, R.: Physics informed neural networks for simulating radiative transfer. Journal of Quantitative Spectroscopy and Radiative Transfer **270**, 107705 (2021) https://doi.org/10.1016/j.jqsrt.2021.107705

[8] Chen, Y., Lu, L., Karniadakis, G.E., Negro, L.D.: Physics-informed neural networks for inverse problems in nano-optics and metamaterials (2019) arXiv:1912.01085 [physics.comp-ph]

[9] Cai, S., Wang, Z., Wang, S., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks for heat transfer problems. Journal of Heat Transfer **143**(6) (2021) https://doi.org/10.1115/1.4050542

[10] Ryck, T.D., Jagtap, A.D., Mishra, S.: Error estimates for physics informed neural networks approximating the Navier-Stokes equations (2022)

[11] Hennigh, O., Narasimhan, S., Nabian, M.A., Subramaniam, A., Tangsali, K., Fang, Z., Rietmann, M., Byeon, W., Choudhry, S.: NVIDIA SimNet™: An AI-accelerated multi-physics simulation framework. In: Computational Science – ICCS 2021, pp. 447–461. Springer (2021). https://doi.org/10.1007/978-3-030-77977-1_36

[12] Gupta, J.K., Brandstetter, J.: Towards multi-spatiotemporal-scale generalized pde modeling. arXiv preprint arXiv:2209.15616 (2022)

[13] Basir, S., Senocak, I.: Critical investigation of failure modes in physics-informed neural networks (2022) https://doi.org/10.2514/6.2022-2353 arXiv:2206.09961

[14] Grossmann, T.G., Komorowska, U.J., Latz, J., Schönlieb, C.-B.: Can Physics-Informed Neural Networks beat the Finite Element Method? arXiv (2023). https://doi.org/10.48550/ARXIV.2302.04107 . https://arxiv.org/abs/2302.04107

[15] Xu, Z.-Q.J.: Frequency Principle in Deep Learning with General Loss Functions and Its Potential Application. arXiv (2018). https://doi.org/10.48550/ARXIV.1811.10146 . https://arxiv.org/abs/1811.10146

[16] Jiang, X., Wang, D., Chen, X., Zhang, M.: Physics-informed neural network for optical fiber parameter estimation from the nonlinear schrödinger equation. Journal of Lightwave Technology, 1–11 (2022) https://doi.org/10.1109/jlt.2022.3199782

[17] Goswami, S., Bora, A., Yu, Y., Karniadakis, G.E.: Physics-Informed Deep Neural Operator Networks (2022)

[18] Goswami, S., Anitescu, C., Chakraborty, S., Rabczuk, T.: Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. Theoretical and Applied Fracture Mechanics **106**, 102447 (2020) https://doi.org/10.1016/j.tafmec.2019.102447

[19] Chen, X., Gong, C., Wan, Q., Deng, L., Wan, Y., Liu, Y., Chen, B., Liu, J.: Transfer learning for deep neural network-based partial differential equations solving. Advances in Aerodynamics **3**(1) (2021) https://doi.org/10.1186/s42774-021-00094-7

[20] Tang, H., Yang, H., Liao, Y., Xie, L.: A transfer learning enhanced the physics-informed neural network model for vortex-induced vibration. arXiv (2021). https:

//doi.org/10.48550/ARXIV.2112.14448 . https://arxiv.org/abs/2112.14448

[21] Wight, C.L., Zhao, J.: Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks (2020)

[22] Cho, J., Nam, S., Yang, H., Yun, S.-B., Hong, Y., Park, E.: Separable PINN: Mitigating the curse of dimensionality in physics-informed neural networks (2022) arXiv:2211.08761 [cs.LG]

[23] Gladstone, R.J., Nabian, M.A., Meidani, H.: FO-PINNs: A First-Order formulation for Physics Informed Neural Networks (2022)

[24] Sharma, R., Shankar, V.: Accelerated Training of Physics-Informed Neural Networks (PINNs) using Meshless Discretizations. arXiv (2022). https://doi.org/10.48550/ARXIV.2205.09332 . https://arxiv.org/abs/2205.09332

[25] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier Neural Operator for Parametric Partial Differential Equations. arXiv (2020). https://doi.org/10.48550/ARXIV.2010.08895 . https://arxiv.org/abs/2010.08895

[26] Lu, L., Jin, P., Karniadakis, G.E.: Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators (2019) https://doi.org/10.1038/s42256-021-00302-5 arXiv:1910.03193

[27] Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., Anandkumar, A.: Physics-Informed Neural Operator for Learning Partial Differential Equations (2021)

[28] Konuk, T., Shragge, J.: Physics-guided deep learning using fourier neural operators for solving the acoustic VTI wave equation. In: 82nd EAGE Annual Conference Exhibition. European Association of Geoscientists & Engineers (2021). https://doi.org/10.3997/2214-4609.202113304 . https://doi.org/10.3997/2214-4609.202113304

[29] Xiong, W., Huang, X., Zhang, Z., Deng, R., Sun, P., Tian, Y.: Koopman neural operator as a mesh-free solver of non-linear partial differential equations (2023)

[30] Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation (2015)

[31] Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., Benson, S.M.: U-FNO – An enhanced Fourier neural operator-based deep-learning model for multiphase flow (2021)

[32] Wang, S., Wang, H., Perdikaris, P.: On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed

neural networks. Computer Methods in Applied Mechanics and Engineering **384**, 113938 (2021) https://doi.org/10.1016/j.cma.2021.113938

[33] Huang, X., Alkhalifah, T., Song, C.: A modified physics-informed neural network with positional encoding. In: First International Meeting for Applied Geoscience: Energy Expanded Abstracts. Society of Exploration Geophysicists, (2021). https://doi.org/10.1190/segam2021-3584127.1 . https://doi.org/10.1190/segam2021-3584127.1

[34] Wong, J.C., Ooi, C., Gupta, A., Ong, Y.-S.: Learning in sinusoidal spaces with physics-informed neural networks. IEEE Transactions on Artificial Intelligence, 1–15 (2022) https://doi.org/10.1109/tai.2022.3192362

[35] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy (2010). https://proceedings.mlr.press/v9/glorot10a.html

[36] DOLFINx. https://github.com/FEniCS/dolfinx. accessed: 12.11.2022 (2017)

[37] Vorst, H.A.: Iterative Krylov Methods for Large Linear Systems. Cambridge University Press, Cambridge; New York (2003). https://www.worldcat.org/title/iterative-krylov-methods-for-large-linear-systems/oclc/50717963&referer=brief_results

[38] Balay, S.: PETSc users manual: Revision 3.10. Technical report (September 2018). https://doi.org/10.2172/1483828 . https://doi.org/10.2172/1483828

[39] Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., Hassanzadeh, P., Kashinath, K., Anandkumar, A.: FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators (2022)

[40] Weinan, Yu, B.: The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**(1), 1–12 (2018) https://doi.org/10.1007/s40304-018-0127-z