Titre: Title:	Physics-informed neural networks with trainable sinusoidal activation functions for approximating the solutions of the Navier-Stokes equations
Auteurs: Authors:	Amirhossein Khademi, & Steven Dufour
Date:	2025
Type:	Article de revue / Article
Référence: Citation:	Khademi, A., & Dufour, S. (2025). Physics-informed neural networks with trainable sinusoidal activation functions for approximating the solutions of the Navier-Stokes equations. Computer Physics Communications, 314, 109672 (15 pages). https://doi.org/10.1016/j.cpc.2025.109672

Titre de la revue: Journal Title:	Computer Physics Communications (vol. 314)
Maison d'édition: Publisher:	Elsevier B.V.
URL officiel: Official URL:	https://doi.org/10.1016/j.cpc.2025.109672

Computer Physics Communications 314 (2025) 109672

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Computational Physics

Physics-informed neural networks with trainable sinusoidal activation functions for approximating the solutions of the Navier-Stokes equations

Amirhossein Khademi *, Steven Dufour

MAGI, Department of Mathematics and Industrial Engineering, Polytechnique Montreal, 2500, Chemin de Polytechnique, H3T 1J4, Montreal, Quebec, Canada

ARTICLE INFO

The review of this paper was arranged by Prof. Andrew Hazel

Dataset link: https://github.com/AmirhosseinnnKhademi/TSA-PINN

Keywords:
Deep learning
Activation functions
Physics-informed neural networks
Partial differential equations
Navier-Stokes equations
Machine learning

ABSTRACT

We present TSA-PINN, a novel Physics-Informed Neural Network (PINN) that leverages a Trainable Sinusoidal Activation (TSA) mechanism to approximate solutions to the Navier-Stokes equations. By incorporating neuron-wise sinusoidal activation functions with trainable frequencies and a dynamic slope recovery mechanism, TSA-PINN achieves superior accuracy and convergence. Its ability to dynamically adjust activation frequencies enables efficient modeling of complex fluid behaviors, reducing training time and computational cost. Our testing goes beyond canonical problems, to study less-explored and more challenging scenarios, which have typically posed difficulties for prior models. Various numerical tests underscore the efficacy of the TSA-PINN model across five different scenarios. These include steady-state two-dimensional flows in a lid-driven cavity at two different Reynolds numbers; a cylinder wake problem characterized by oscillatory fluid behavior; and two time-dependent three-dimensional turbulent flow cases. In the turbulent cases, the focus is on detailed near-wall phenomena—including the viscous sub-layer, buffer layer, and log-law region—as well as the complex interactions among eddies of various scales. Both numerical and quantitative analyses demonstrate that TSA-PINN offers substantial improvements over conventional PINN models. This research advances physics-informed machine learning, setting a new benchmark for modeling dynamic systems in scientific computing and engineering.

1. Introduction

Machine learning (ML), particularly deep learning via multilayer neural networks, has shown promise for performing nonlinear mapping tasks through the use of surrogate models. But, such models often lack physical interpretability, which is crucial in applied science, where adherence to governing equations is paramount for generalization. Many scientific disciplines do not rely on extensive datasets, leading to models that are faster but not necessarily predictive. Hence, it is important to integrate existing knowledge, inherent physics principles, and domain expertise to constrain these models during training, with the limited data available. Raissi et al. [1] introduced Physics-Informed Neural Networks (PINN), with subsequent variants further developed to applied to fluid mechanics problem [2,3]. PINN included physical laws in the ML model using a weak loss function that incorporates the residuals of the conservation equations and boundary condition constraints. Despite their promising performance, PINN do not perform well in high-dimensional scenarios. In fluid dynamics, particularly for approximating solutions to the Navier-Stokes equations, PINN models are characterized by issues of convergence and extended training durations.

Based on the foundational work on data clustering of Mao et al. [4], Jagtap et al. [5] proposed domain decomposition techniques, namely conservative PINN (CPINN), to make it possible to use independent models in separate subdomains, to set the stage for the parallelization of PINN models. This method was further extended to also cover temporal decomposition [6], and finally led to the parallel PINN [7]. A time-sweeping and information propagation scheme was later incorporated in the scientific ML framework, to optimize training schedules on subdomains [8]. This approach led to computational speed-ups. Recently, Khademi and Dufour [9] refined the regularity and smoothness of solutions in the discretized PINN framework by introducing additional loss terms that account for the H^1 norm of the error on the propagation of information. They also enhanced expressibility of the model by integrating transformer networks (subdomain-wise) into the architecture, inspired by the work of Wang et al. [10]. Lorin and Yang [11] explored the capabilities of PINN framework to solve the time-dependent Dirac equation, showcasing its efficiency in quantum relativistic physics without

* Corresponding author.
E-mail addresses: amirhossein.khademi@polymtl.ca, amir.hn.khademi@gmail.com (A. Khademi), steven.dufour@polymtl.ca (S. Dufour).

https://doi.org/10.1016/j.cpc.2025.109672
Received 23 November 2024; Received in revised form 16 April 2025; Accepted 8 May 2025
Available online 15 May 2025

requiring derivative discretization. They also propose a quasi-optimal Schwarz Waveform Relaxation (SWR) domain decomposition method accelerated with PINN for solving the time-dependent Schrödinger equation. By leveraging Dirichlet-to-Neumann transmission operators and PINN, they achieve faster convergence rates with reduced computational costs compared to classical SWR methods [12]. Zhang et al. [13] introduce Trans-Net, a neural network framework that uses temporal domain decomposition to efficiently solve partial differential equations. This method improves accuracy and reduces training time by leveraging pre-trained networks from previous subdomains for subsequent calculations. Finally, Ren et al. [14] introduce SeismicNet, a physics-informed neural network model for seismic wave modeling in semi-infinite domains. This approach enhances scalability and accuracy using absorbing boundary conditions and temporal domain decomposition, demonstrating improvements over traditional methods without needing labeled data.

Zhang et al. [15] introduced a PINN model to study incompressible fluid flows around a cylinder, governed by the Navier-Stokes equations. This model incorporates the geometry of the cylinder, boundary and initial conditions, and fluid properties. By integrating Fourier features, the PINN demonstrated improved predictive accuracy across various design parameters and temporal scenarios. Lou et al. [16] illustrated the capabilities of PINN models for tackling inverse multiscale flow challenges. Implementing PINNs in inverse modeling across continuum and rarefied regimes via the Boltzmann-Bhatnagar-Gross-Krook (BGK) model, they demonstrated that PINN-BGK offer a versatile solution, facilitating both forward and inverse modeling.

To improve training efficiency, Chiu et al. [17] integrated numerical discretizations with automatic differentiation and applied this technique to flow mixing problems, lid-driven cavity flows, and channel flows over a backward-facing step. This integration significantly reduces the number of required collocation points, accelerating convergence rates. Psaros et al. [18] introduced a meta-learning framework into the PINN, leading to improved performance on tasks outside the typical training distribution, using computational resources more efficiently. Concurrently, Penwarden et al. [19] explored model-agnostic meta-learning and transfer learning concepts before implementing a specialized meta-learning adaptation tailored to PINNs. This approach has been verified in a variety of settings. Geometry-aware PINNs (GA-PINNs) incorporate a variational auto-encoder alongside a boundary-constrained network, facilitate the study of applications in complex, non-parametrizable geometries [20]. Spline-PINNs [21] merge PINNs with convolutional neural networks (CNN), using Hermite spline kernels. This combination is designed to train PINNs with minimal data, offering quick and continuous solution inference that extend to changing boundary conditions. This approach involves longer training phases across a spectrum of parametric variables, but allows to perform rapid inferences for various parametric values.

Numerical stiffness presents a considerable obstacle for PINN models, leading to disproportionate gradients during the back-propagation process in training. A broad range of solutions, detailed in several studies [22–25], have been explored, mainly focusing on regularization techniques, penalization strategies, or by adapting learning rates to balance the loss term coefficients. Manually adjusting these coefficients is not only time-consuming, but it also tends to be less effective, particularly because the solutions are prone to instabilities from small variations of these coefficients. Wang et al. [10] introduced a method that adjusts learning rates dynamically. This method calculates the coefficients using a moving average, taking into consideration the relative magnitude of the various loss terms. Extending this approach to time-sensitive issues led to the concept of causal PINN [26] was developed to manage dynamic systems characterized by complex behaviors like chaos or turbulence. Here, weighting coefficients are applied to loss terms in a manner that prioritizes the minimization of a specific loss term at a given time, contingent on the satisfactory minimization of previous losses. Song et al. [27] proposed the LA-PINN model, which incorporates a novel loss-attention mechanism. This model uses different attention-driven networks for each type of loss, dynamically adjusting weights at individual training points throughout the training, enhancing the performance of the model in areas of numerical stiffness.

Using the PINN frameworks to approximate solutions of high-dimensional and nonlinear problems presents substantial challenges that are not typically encountered in other applications. Problems involving high Reynolds numbers exhibit chaotic behavior and rapid changes. Standard PINN models often fall short for accuracy and computing efficiency in capturing these complex and nonlinear phenomena. These complexities necessitate the development of sophisticated and customized models. Eivazi et al. [28] incorporate Reynolds-Averaged Navier-Stokes (RANS) equations in the loss function of a PINN model to enable the PINN to deal with turbulence. To validate, the model was tested on problems involving a Zero Pressure Gradient (ZPG), an Adversial Pressure Gradient (APG), a NACA4412 airfoil, and a Periodic hill. Xu et al. [29] utilized parameterized Navier-Stokes equations as constraints. Using RANS model with eddy viscosity, showed that missing data in randomly specified regions can be inferred. Similar approaches were used in other studies to approximate turbulent flows [30,31]. Patel et al. [32] used the Spalart-Allmaras (SA) turbulence model, augmented with a PINN approach for data assimilation in turbulent flow scenarios. This hybrid method significantly enhances mean flow reconstruction accuracy when compared to traditional RANS solvers. Wang et al. [33] used a hybrid turbulence modeling approach that integrates RANS and Large Eddy Simulation (LES) models with deep learning. It introduces trainable spectral filters and uses a specialized U-net architecture for turbulence prediction, achieving reductions in prediction error and predicting physical fields that respect conservation laws, like mass conservation, while accurately emulating turbulent kinetic energy fields. Sliwinski and Rigas [34] introduced a PINN methodology for the reconstruction of the mean velocity and the Reynolds stress fields in turbulent flows, applying it within the context of the RANS equations. The paper uses sparse velocity data to successfully infer unknown closure quantities like the curl of unsteady RANS forcing, enhancing the assimilation of Reynolds stresses to complete the flow fields. The results underscore the ability of PINNs to accurately interpolate flow from sparse measurements, highlighting their potential to reduce experimental and computational costs in fluid dynamics studies. Turbulence filtering methods simplify computations by averaging flow characteristics, but this approach makes PINNs to overlook transient and fluctuating components, and phenomena with inherently unsteady behaviors, such as sharp gradients and near-wall phenomena. This simplification compromises the accuracy of this approach for capturing complex flow dynamics. Jin et al. [35] used PINNs to directly model 3D time-dependent turbulent channel flows at a Reynolds number around 1000, without using a turbulence model. They focused on optimizing the weighting of the loss function to balance data and physics, and they achieved results that were in good agreement with Direct Numerical Simulation (DNS) data, but at the expense of very long training time. Finally, Khademi et al. [36] introduced a time-marching approach with generative modeling—called DG-PINN—for modeling high-dimensional, time-dependent, 3D turbulent channel flow governed by the Navier–Stokes equations. By respecting causality, leveraging temporal discretization, and sharing generative information across time steps, DG-PINN reduced memory usage through a parallel setup while preserving approximation accuracy as time progresses, thereby addressing a key limitation of PINNs in high dimensional problems.

It is important to focus on the fundamental aspects of the Deep Neural Network models. A critical component in every Artificial Neural Network (ANN) model is the activation function. Without this function, the model would effectively just act as a linear mapping between the input and output spaces, regardless of the number of layers or neurons it contains. There is no one-size-fits-all activation function for neural networks; selection often hinges on specific problems. Various methods were explored accordingly, such as Yu et al. [37]'s adaptive sig-

moidal activation function for multilayer networks, Qian et al. [38]'s data-driven method of combining activation functions for convolutional neural networks, and Dushkoff and Ptucha [39]'s technique allowing neurons to choose from multiple activation functions. Abbasi and Andersen [40] introduced tailored activation functions derived from the physical principles governing the problem being modeled, differing from standard activation functions like tangent hyperbolic (tanh) or sigmoid functions. By integrating them into PINN models, the networks become more constrained by the underlying physics, which improves their predictive accuracy and efficiency. Jagtap et al. [41] used adaptive activation functions to enhance the modeling of both smooth and high-gradient solutions of the partial differential equations like the nonlinear Klein-Gordon, Burgers, and Helmholtz equations. These adaptive functions, equipped with a scalable hyper-parameter, dynamically adjust the loss function's topology during optimization. In another study [42] they introduced scalable parameters at both the layer and neuron levels, optimizing them via stochastic gradient descent to enhance training speeds and to prevent suboptimal convergence. Their methodology showed accelerated training and improved solution accuracy for both forward and inverse problems.

Existing Physics-Informed Neural Networks (PINNs), despite their considerable promise, often struggle to efficiently and accurately model complex fluid dynamics, especially in scenarios involving high-dimensional problems, rapid temporal changes, and chaotic phenomena. Traditional PINNs frequently encounter issues with slow convergence, extensive training durations, and difficulties in capturing intricate near-wall behaviors and multi-scale interactions inherent in turbulent flows. Addressing these bottlenecks necessitates a more fundamental investigation into neural network components—particularly activation functions, which significantly influence model expressivity and performance. Motivated by these challenges and the need to improve upon the limitations of current methods, this study introduces a novel PINN model employing trainable sinusoidal activation functions (TSA-PINN). By incorporating neuron-specific sinusoidal components with frequencies initialized to predefined values and subsequently optimized via gradient descent, the proposed method aims to enhance the expressivity and training efficiency of PINNs. The efficacy of TSA-PINN is evaluated through comprehensive tests involving the steady-state lid-driven cavity flow at various Reynolds numbers, the time-dependent cylinder wake problem exhibiting vortex shedding, and a more complex, realistic scenario of 3D turbulent channel flow. This approach not only closely matches reference solutions but also demonstrates significant improvements over standard PINN models in capturing complex fluid behaviors, thus filling a crucial research gap and setting a foundation for future explorations into advanced activation strategies in PINNs.

2. Methodology

2.1. Physics-informed neural networks

PINNs integrate observational data and known governing equations in a NN model to approximate solutions of physical systems. Specifically, they estimate the state vector $\hat{\boldsymbol{u}}(\boldsymbol{x},t)$ such that it approximates the actual system states $\boldsymbol{u}(\boldsymbol{x},t)$, where $\boldsymbol{x}\in\mathbb{R}^d$ represents spatial coordinates within the domain and $t\in[0,T]$ denotes time.

The governing dynamics of the system are encapsulated by the nonlinear operator $\mathcal{N}$, and the system's behavior is modeled through the differential equation:

$$\frac{\partial \boldsymbol{u}}{\partial t}+\mathcal{N}(\boldsymbol{u};\boldsymbol{x},t)=0. \tag{1}$$

PINN models aim at minimizing a loss function that includes both a data fidelity term and a physics-informed term. The data fidelity term is designed to minimize the discrepancy between the predicted and the reference solutions

$$L_{\text{data}}=\sum_{i=1}^{N_{\text{data}}}\left\|\hat{\boldsymbol{u}}(\boldsymbol{x}_i,t_i)-\boldsymbol{u}(\boldsymbol{x}_i,t_i)\right\|^2, \tag{2}$$

where N_{data} is the number of available data points and $(\boldsymbol{x}_i,t_i)$ represents the spatio-temporal coordinates of these points.

The physics-informed term ensures that the predictions adhere to the governing partial differential equations (PDEs),

$$L_{\text{phys}}=\sum_{j=1}^{N_{\text{phys}}}\left\|\mathcal{N}(\hat{\boldsymbol{u}}(\boldsymbol{x}_j,t_j);\boldsymbol{x}_j,t_j)\right\|^2, \tag{3}$$

where N_{phys} denotes the number of points at which the physical laws are enforced.

The goal during training is to minimize the combined loss function

$$L_{\text{PINN}}=L_{\text{data}}+L_{\text{phys}}, \tag{4}$$

where we want the approximate solution $\hat{\boldsymbol{u}}(\boldsymbol{x},t)$ converges towards the true solution $\boldsymbol{u}(\boldsymbol{x},t)$. This process ensures that the model not only fits the observed data, but also satisfies the underlying physical principles that govern the system.

2.2. Neural networks

The universal approximation theorem establishes that even a simple multi-layer perceptron with a single hidden layer can approximate any continuous function to an arbitrary degree of precision by increasing the number of neurons [43–45]. The widely used architecture in PINN models is the Feedforward Neural Network (FFNN), consisting of multiple fully connected layers. Each neuron's output in a given layer is described by

$$f_i(\boldsymbol{x};\boldsymbol{w}_i,\boldsymbol{b}_i)=\alpha(\boldsymbol{w}_i\cdot\boldsymbol{x}+\boldsymbol{b}_i),\quad \text{for } i=1,2,\ldots,n, \tag{5}$$

where x denotes the input vector, and w_i and b_i are the weight vector and bias for the i-th neuron, respectively. The nonlinear activation function α enables the network to capture complex relationships within the data.

In approximating solutions of the Navier-Stokes equations, the FFNN receives spatial and temporal coordinates $X=(t,x,y,z)$ as inputs. The network outputs consist of the velocity field $U(X)=(u,v,w)$ and the pressure field $P(X)$, as expressed by

$$\boldsymbol{U}\theta(\boldsymbol{X}),\boldsymbol{P}\theta(\boldsymbol{X})=f_{\text{NN}}(\boldsymbol{X},\theta), \tag{6}$$

where $f_{NN}(X,\theta)$ is the neural network function approximating the velocity and pressure fields, and $\theta=w,b$ denotes all trainable parameters of the network. Training the network involves optimizing θ by minimizing the discrepancy between predicted and actual physical fields, formulated in the loss function (equations (1), (2), (3), and (4)). The final FFNN representation is written as:

$$f_{\text{NN}}(\boldsymbol{x},\theta)=(a_L\circ\alpha\circ a_{L-1}\circ\cdots\circ\alpha\circ a_1)(x), \tag{7}$$

where $a(x)=wx+b$.

2.3. Trainable sinusoidal activation of physics-informed neural networks

2.3.1. Trainable sinusoidal activation mechanism

The purpose of an activation function is to determine whether a neuron should activate or remain inactive. In the absence of a nonlinear activation function, the model would simply perform a linear transformation using the weights and biases, which corresponds to a linear regression model. Let us assume that no nonlinear activation function is used, which means that $f(z)=z$. In this case, the output of each layer simply becomes (see Fig.1)

$$a_i=z_i=w_ia_{i-1}+b_i, \tag{8}$$

starting with the input layer

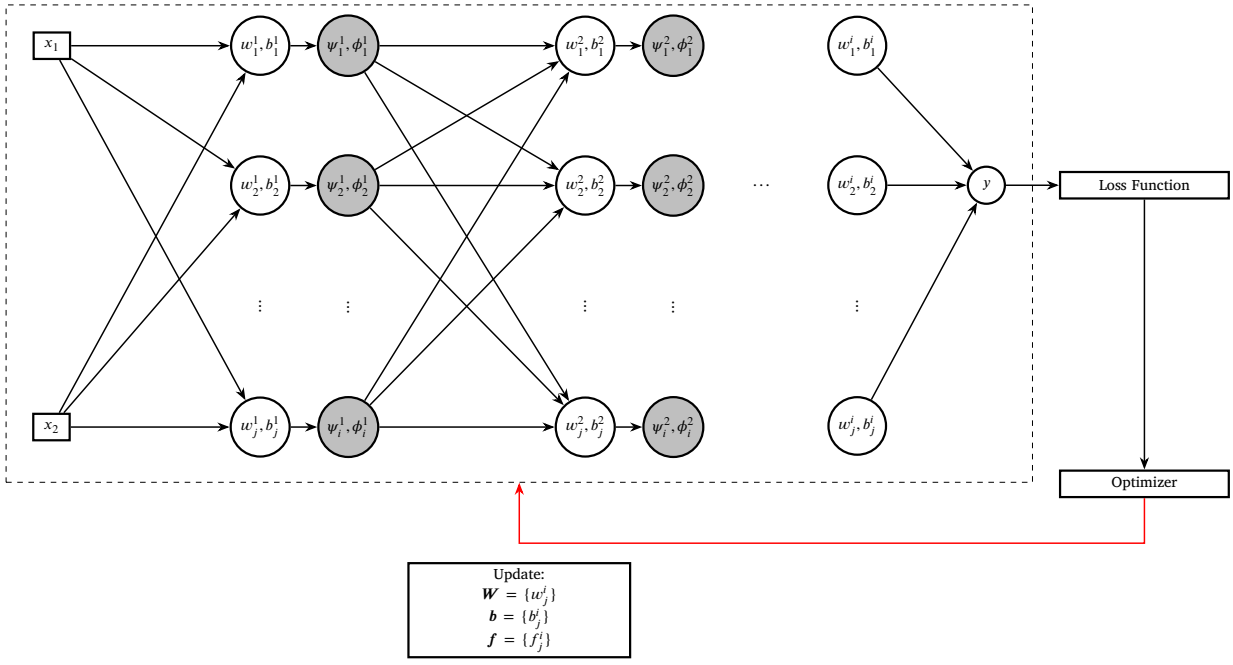


Fig. 1. Schematics of the TSA-PINN. The white circles represent the multiplication of weights and the addition of biases. The gray circles indicate the application of activation functions. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$a_1 = w_1 x + b_1, \tag{9}$$

the second layer

$$\begin{aligned} a_2 &= w_2 a_2 + b_2 = w_2(w_1 x + b_1) + b_2 \\ &= w_1 w_2 x + w_2 b_1 + b_2 \end{aligned} \tag{10}$$

and the third layer

$$\begin{aligned} a_3 &= w_3 a_2 + b_3 = w_3(w_2 w_1 x + w_2 b_1 + b_2) + b_3 \\ &= w_3 w_2 w_1 x + w_3 w_2 b_1 + w_3 b_2 + b_3. \end{aligned} \tag{11}$$

This process continues until the final layer L is reached

$$\begin{aligned} a_L = w_L w_{L-1} \dots w_2 w_1 x + (w_L w_{L-1} \dots w_2 b_1 + w_L w_{L-1} \\ \dots w_3 b_2 + \dots + b_L). \end{aligned} \tag{12}$$

At the final layer, the output of the network can be written as

$$\boldsymbol{y} = \boldsymbol{w}_{\text{total}} x + \boldsymbol{b}_{\text{total}} \tag{13}$$

where $\boldsymbol{w}_{\text{total}} = w_L w_{L-1} \dots w_2 w_1$ is a matrix resulting from multiplying all the weight matrices and $\boldsymbol{b}_{\text{total}} = (w_L w_{L-1} \dots w_2 b_1 + w_L w_{L-1} \dots w_3 b_2 + \dots + b_L)$ is the combination of all the bias terms. This expression is a linear transformation of the input $\boldsymbol{x}$. There is no nonlinearity introduced in the network, so regardless of the depth of the network (number of layers), the output remains a linear function of the input (13). For a standard fully connected neural network (FFNN), the pre-activation output of neuron i in layer k is given by:

$$z_i^{(k)} = \mathbf{w}_i^{(k)} \cdot \mathbf{a}^{(k-1)} + b_i^{(k)}, \tag{14}$$

where:

- $\mathbf{a}^{(k-1)} \in \mathbb{R}^{n_{k-1}}$ is the input vector from layer $k-1$,
- $\mathbf{w}_i^{(k)} \in \mathbb{R}^{n_{k-1}}$ is the weight vector for neuron i in layer k,
- $b_i^{(k)} \in \mathbb{R}$ is the bias of neuron i,
- $z_i^{(k)} \in \mathbb{R}$ is the scalar pre-activation output.

In this work, we use a neuron-wise sinusoidal activation function with a trainable frequency $f_i^{(k)} \in \mathbb{R}$. The activation output of each neuron is computed as:

$$\psi_i^{(k)} = \sin(f_i^{(k)} z_i^{(k)}), \tag{15}$$

$$\phi_i^{(k)} = \cos(f_i^{(k)} z_i^{(k)}), \tag{16}$$

$$a_i^{(k)} = \zeta_1 \psi_i^{(k)} + \zeta_2 \phi_i^{(k)} = \zeta_1 \sin(f_i^{(k)} z_i^{(k)}) + \zeta_2 \cos(f_i^{(k)} z_i^{(k)}), \tag{17}$$

where $\zeta_1, \zeta_2 \in \mathbb{R}$ are trainable or fixed scalar coefficients shared across the layer.

Let $\mathbf{a}^{(k-1)} \in \mathbb{R}^{n_{k-1}}$ be the input vector to layer k. The vector of pre-activations in the layer is:

$$\mathbf{z}^{(k)} = \mathbf{W}^{(k)} \mathbf{a}^{(k-1)} + \mathbf{b}^{(k)}, \tag{18}$$

where:

- $\mathbf{W}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$ is the weight matrix of layer k,
- $\mathbf{b}^{(k)} \in \mathbb{R}^{n_k}$ is the bias vector,
- $\mathbf{z}^{(k)} \in \mathbb{R}^{n_k}$ is the pre-activation vector.

The elementwise sinusoidal activations are applied using a vector of trainable frequencies $\mathbf{f}^{(k)} \in \mathbb{R}^{n_k}$:

$$\boldsymbol{\psi}^{(k)} = \sin(\mathbf{f}^{(k)} \odot \mathbf{z}^{(k)}), \tag{19}$$

$$\boldsymbol{\phi}^{(k)} = \cos(\mathbf{f}^{(k)} \odot \mathbf{z}^{(k)}), \tag{20}$$

where $\odot$ denotes the Hadamard (element-wise) product.

The final output of layer k with $N = n_k$ neurons is then given explicitly by:

$$\mathbf{a}^{(k)} = \begin{bmatrix} a_1^{(k)} \\ a_2^{(k)} \\ \vdots \\ a_N^{(k)} \end{bmatrix} = \begin{bmatrix} \zeta_1 \sin(f_1^{(k)} z_1^{(k)}) + \zeta_2 \cos(f_1^{(k)} z_1^{(k)}) \\ \zeta_1 \sin(f_2^{(k)} z_2^{(k)}) + \zeta_2 \cos(f_2^{(k)} z_2^{(k)}) \\ \vdots \\ \zeta_1 \sin(f_N^{(k)} z_N^{(k)}) + \zeta_2 \cos(f_N^{(k)} z_N^{(k)}) \end{bmatrix}. \tag{21}$$

This formulation enables each neuron to adapt its activation frequency independently, allowing the network to represent functions

with varying and potentially high-frequency components across different neurons.

2.3.2. Slope recovery

The slope recovery term $S(a)$ adjusts the slope of the activation functions dynamically, which is crucial for maintaining active and effective gradient propagation across the network. By incorporating this term, inspired by Jagtap et al. [42], the network is forced to rapidly enhance the activation slope, consequently accelerating the training process. We have

$$S(a) = \frac{1}{\frac{1}{L-1}\sum_{k=1}^{L-1}\exp\left(\frac{1}{N_k}\sum_{i=1}^{N_k} f_i^k\right)}, \tag{22}$$

where:

- L represents the total number of layers;
- N_k denotes the number of neurons in the k-th layer;
- f_i^k is the trainable frequency for the i-th neuron in the k-th layer.

The slope recovery term is included into the loss function to regulate the impact of trainable frequencies on training dynamics. The augmented loss function with the slope recovery term is written as

$$L_{\text{PINN}} = L_{\text{data}} + L_{\text{phys}} + \lambda S(a), \tag{23}$$

where λ is a hyperparameter that determines the weight of the slope recovery term in the total loss, restricting the model to optimize frequency parameters.

2.3.3. The loss function

To establish the loss function associated with the TSA-PINN model, a problem involving a 3D time-dependant turbulent channel flow is considered. The incompressible Navier-Stokes equations, that govern the flow, are written in the Velocity-Pressure (VP) form as:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p - \frac{1}{Re}\Delta\boldsymbol{u} = 0, \qquad \text{in } \Omega; \tag{24}$$

$$\nabla\cdot\boldsymbol{u} = 0, \qquad \text{in } \Omega; \tag{25}$$

$$\boldsymbol{u} = u_\Gamma, \qquad \text{on } \Gamma_D; \tag{26}$$

$$\frac{\partial \boldsymbol{u}}{\partial n} = 0, \qquad \text{on } \Gamma_N. \tag{27}$$

In this context, the non-dimensional time is t. The non-dimensional velocity vector is denoted by $\boldsymbol{u}(x,y,z,t) = [u,v,w]^T$, and P denotes the non-dimensional pressure. The Reynolds number, Re, is a parameter used to characterizing the dynamics of the flow by comparing inertial forces to viscous forces. It is defined as $Re = \frac{\rho u L}{\mu}$, where ρ is the fluid density, u is the characteristic velocity, L represents the characteristic length scale, and μ is the dynamic viscosity. The Dirichlet and Neumann boundary conditions are given by equations (26) and (27), respectively. The residuals associated with the conservation of the momentum and continuity equations, (24) and (25) respectively, can be expressed as:

$$R_x = \partial_t u + u\partial_x u + v\partial_y u + w\partial_z u + \partial_x p - \frac{1}{Re}(\partial^2_{xx}u + \partial^2_{yy}u + \partial^2_{zz}u); \tag{28}$$

$$R_y = \partial_t v + u\partial_x v + v\partial_y v + w\partial_z v + \partial_y p - \frac{1}{Re}(\partial^2_{xx}v + \partial^2_{yy}v + \partial^2_{zz}v); \tag{29}$$

$$R_z = \partial_t w + u\partial_x w + v\partial_y w + w\partial_z w + \partial_z p - \frac{1}{Re}(\partial^2_{xx}w + \partial^2_{yy}w + \partial^2_{zz}w); \tag{30}$$

$$R_c = \partial_x u + \partial_y v + \partial_z w. \tag{31}$$

Here, R_x, R_y, R_z, and R_c denote the residuals for the momentum equations in the x, y, and z directions, and the divergence-free constraint, respectively. To compute the partial differential operators, automatic differentiation is used [46]. This approach involves calculating the derivatives of the outputs from the computational graph with respect to the variables x, y, z, and t, to approximate the derivatives in the governing equations. In the context of TSA-PINN, the approximation problem is reformulated as an optimization problem involving the network parameters $\boldsymbol{w}, \boldsymbol{b}$ and $\boldsymbol{f}$. The goal is to minimize a loss function related to the approximation of the solution. The loss function is expressed as:

$$L = L_{\text{IC}} + L_{\text{BC}} + L_{\text{R}} + \lambda L_{\text{S}}, \tag{32}$$

and the loss terms are written as:

$$L_{\text{IC}} = \frac{1}{N_{\text{I}}}\sum_{n=1}^{N_{\text{I}}}\left|\boldsymbol{u}_\theta^n - \boldsymbol{u}_{\text{IC}}^n\right|^2; \tag{33}$$

$$L_{\text{BC}} = \frac{1}{N_{\text{B}}}\sum_{n=1}^{N_{\text{B}}}\left|\boldsymbol{u}_\theta^n - \boldsymbol{u}_{\text{BC}}^n\right|^2; \tag{34}$$

$$L_{\text{R}} = \frac{1}{N_{\text{R}}}\left(\sum_{n=1}^{N_{\text{R}}}\left|R_x^n\right|^2 + \sum_{n=1}^{N_{\text{R}}}\left|R_y^n\right|^2 + \sum_{n=1}^{N_{\text{R}}}\left|R_z^n\right|^2 + \sum_{n=1}^{N_{\text{R}}}\left|R_c^n\right|^2\right), \tag{35}$$

where L_{IC}, L_{BC}, and L_{R} denote the errors associated with the approximations of the initial conditions (IC), the boundary conditions (BC), and the residuals of the governing PDEs, respectively. The last term of equation (32) represents the slope recovery term, as introduced in equation (22). The optimization problem discovers the optimal values of network parameters such that to minimize the loss associated with the approximation:

$$\boldsymbol{W}^* = \arg\min_w(L(w)); \tag{36}$$

$$\boldsymbol{b}^* = \arg\min_b(L(b)); \tag{37}$$

$$\boldsymbol{f}^* = \arg\min_f(L(f)). \tag{38}$$

This minimization problem is approximated using a gradient descent approach. The model parameters are updated as:

$$w^{m+1} = w^m - \eta\nabla_{w^m}L^m(w); \tag{39}$$

$$b^{m+1} = b^m - \eta\nabla_{b^m}L^m(b); \tag{40}$$

$$f^{m+1} = f^m - \eta\nabla_{f^m}L^m(f), \tag{41}$$

where, at the m-th iteration, η denotes the learning rate and L^m is the loss function.

3. Results and discussion

To validate the TSA-PINN, it is applied to approximate the Navier–Stokes equations in various scenarios:

1. 2D steady-state lid-driven cavity problem at $Re = 100$;
2. 2D steady-state lid-driven cavity problem at $Re = 3200$;
3. 2D time-dependent Cylinder Wake;
4. 3D time-dependent turbulent channel flow: near-wall region;
5. 3D time-dependent turbulent channel flow: over a larger domain.

To estimate the error associated with each scenario, the relative L_2 norm of the error at all evaluation points is used

$$\text{Error}_i = \frac{\|\hat{\boldsymbol{U}}_i - \boldsymbol{U}_i\|_2}{\|\boldsymbol{U}_i\|_2} \times 100, \tag{42}$$

where the subscript i denotes the index of the variable and $\|\cdot\|_2$ represents the L_2 norm. $\hat{\boldsymbol{U}}$ and $\boldsymbol{U}$ indicate the vectors of the approximated and the reference solutions, respectively. The standard PINN model uses the tanh activation function. For both models, TSA-PINN and standard PINN, weights and biases are initialized using the Glorot normal method

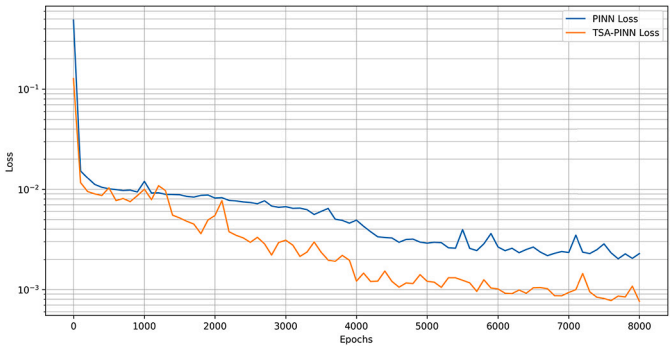


Fig. 2. Lid-driven cavity problem at $Re = 100$: Comparison of loss decays.

[47]. Trainable frequencies are initialized using $\sigma = 1.0$ unless specified otherwise. A gradient descent method is used in all cases, with the ADAM optimizer [48]. TensorFlow is used for automatic differentiation and computational graph construction [49].

3.1. 2D lid-driven cavity problem at $Re = 100$

The first test case is a steady-state flow in a two-dimensional lid-driven cavity, governed by the 2D steady-state incompressible Navier-Stokes equations (24) and (25). For this problem, we set the Reynolds number to $Re = 100$, and the system is expected to converge to a steady-state solution [50]. Spatial coordinates $x \in [0, 1]$ and $y \in [0, 1]$ are provided as inputs to the network, which outputs the stream function ψ and the pressure field P. All the variables are non-dimensional. By adopting the stream-function formulation, the solutions to the Navier-Stokes equations are explored in a set of divergence-free functions

$$u_x + v_y = 0. \tag{43}$$

In this setting, the velocity components are

$$u = \partial_y \psi, \tag{44}$$

and

$$v = -\partial_x \psi. \tag{45}$$

This assumption automatically satisfies the continuity constraint. It is important to note that no training data from within the domain are available for this problem. The training relies entirely on unsupervised learning, using 4,000 collocation points within the domain, and 500 boundary condition points along the domain boundaries. In this setting, the residuals of the governing equations are:

$$R_x = u\partial_x u + v\partial_y u + \partial_x p - \frac{1}{Re}(\partial^2_{xx} u + \partial^2_{yy} u); \tag{46}$$

$$R_y = u\partial_x v + v\partial_y v + \partial_y p - \frac{1}{Re}(\partial^2_{xx} v + \partial^2_{yy} v). \tag{47}$$

These terms are included in the loss function to satisfy the physics-informed part. The learning rate is set to exponential decay, starting from an initial value of $\eta = 10^{-3}$ with a decay rate of 0.9 every 1000 steps. For this case, training consists in 8,000 iterations. To estimate the error, we used a spatial mesh grid of 400×400 points. Finally, the neural network architecture used for this simulation involves 5 hidden layers with 50 neurons in each layer.

One primary advantage of the TSA-PINN compared to the standard PINN is its enhanced model expressibility, which leads to faster convergence rates, thanks to the use of neuron-wise sinusoidal activation functions with trainable frequencies. Fig. 2 illustrates that the loss decay of the TSA-PINN, is significantly faster when compared to standard PINNs, by nearly half an order of magnitude. This improvement is further illustrated in Fig. 3. While the simulation using the standard PINN struggles to accurately capture the velocity fields (Fig. 3-a), the TSA-PINN exhibits minimal error and accurately captures all phenomena (Fig. 3-b).

Table 1
Error and runtime comparison (2D lid-driven cavity problem at $Re = 100$): Standard PINN vs. TSA-PINN.

Method	Epochs	Run-time (s)	Rel. L_2-err.
Standard PINN	8,000	1,161	1.51×10^{-1}
TSA-PINN	8,000	1,438	2.74×10^{-2}

Table 2
Error and runtime comparison (2D lid-driven cavity problem at $Re = 3200$): Standard PINN vs. TSA-PINN.

Method	Epochs	Run-time (s)	Rel. L_2-err.	layers-neurons
PINN	6,000	2,150	1.09×10^{-1}	8×100
TSA-PINN	6,000	2,691	3.92×10^{-2}	8×100

Fig. 4 shows a quantitative analysis of the approximations made by both the TSA-PINN (Fig. 4-b) and the standard PINN (Fig. 4-a). This analysis quantitatively reaffirms the earlier conclusion, highlighting the superior accuracy of the TSA-PINN. The approximations made by the TSA-PINN align almost exactly with the reference solution. The standard PINN approximations exhibit noticeable errors. Table 1 gives a comparison of the L_2 norm of the approximation errors between the TSA-PINN and standard PINNs. The TSA-PINN achieves higher accuracy compared to the standard PINN, where the errors reduced by an order of magnitude. This increase in accuracy comes at the cost of approximately 24% additional computational time.

3.2. 2D lid-driven cavity problem at $Re = 3200$

To further evaluate the robustness and capability of TSA-PINN in handling more challenging flow problems, we applied the method to the two-dimensional lid-driven cavity problem at a higher Reynolds number, $Re = 3200$, governed by the incompressible Navier-Stokes equations. Following the recommendations of Wang et al. [51], and to mitigate discontinuities at the two corners of the moving lid boundary, we reformulated the top boundary condition as:

$$u(x, y) = 1 - \frac{cosh(C_0(x - 0.5))}{cosh(0.5C_0)}, \tag{48}$$

$$v(x, y) = 0, \tag{49}$$

where $x \in [0, 1]$, $y = 1$, and $C_0 = 50$. We use three sets of evaluation points: 7000 points for the PDE residuals, 1000 for the boundary conditions, and 100 for the training data, which were sampled from the results of a direct numerical simulation.

For the lid-driven cavity problem at $Re = 3200$, our results highlight the superior performance of TSA-PINN compared to the standard PINN framework. In Fig. 5, the velocity contours and approximation errors are presented for both methods. In panel (a), the standard PINN results reveal less accurate flow feature capture, particularly in the near-wall regions, whereas panel (b) demonstrates that TSA-PINN better resolves these flow structures, yielding lower approximation errors. The quantitative validation of the solution is provided in Fig. 6, where the approximated solutions along selected cross-sections are compared to the reference data. TSA-PINN's predictions are in close agreement with the reference, affirming its ability to accurately capture the essential flow dynamics at high Reynolds numbers.

Fig. 7 displays the loss histories for the two approaches. TSA-PINN shows a smoother and more stable convergence behavior with a lower final loss value compared to the standard PINN, which indicates enhanced training efficiency and improved numerical stability. Finally, Table 2 summarizes the error metrics and runtime comparisons between

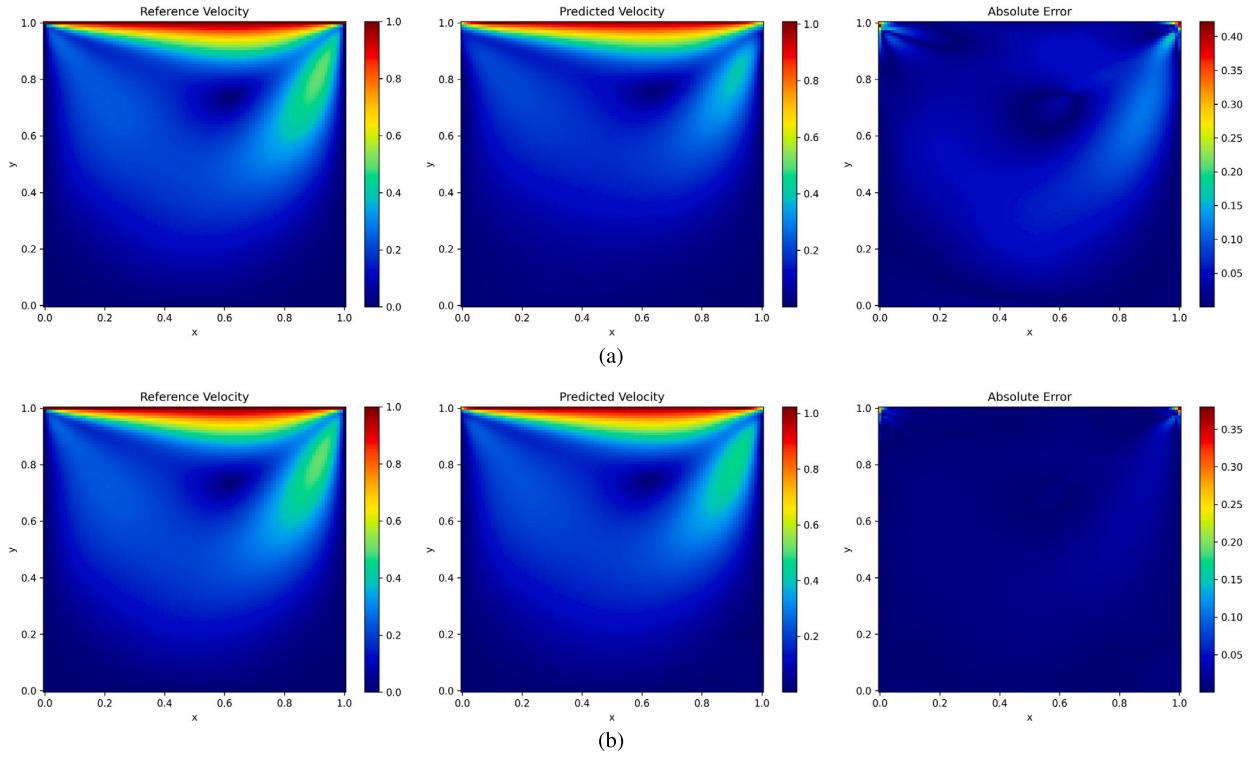


Fig. 3. Lid-driven cavity problem at $Re = 100$: Velocity contours and approximation error. (a) Standard PINN; (b) TSA-PINN.

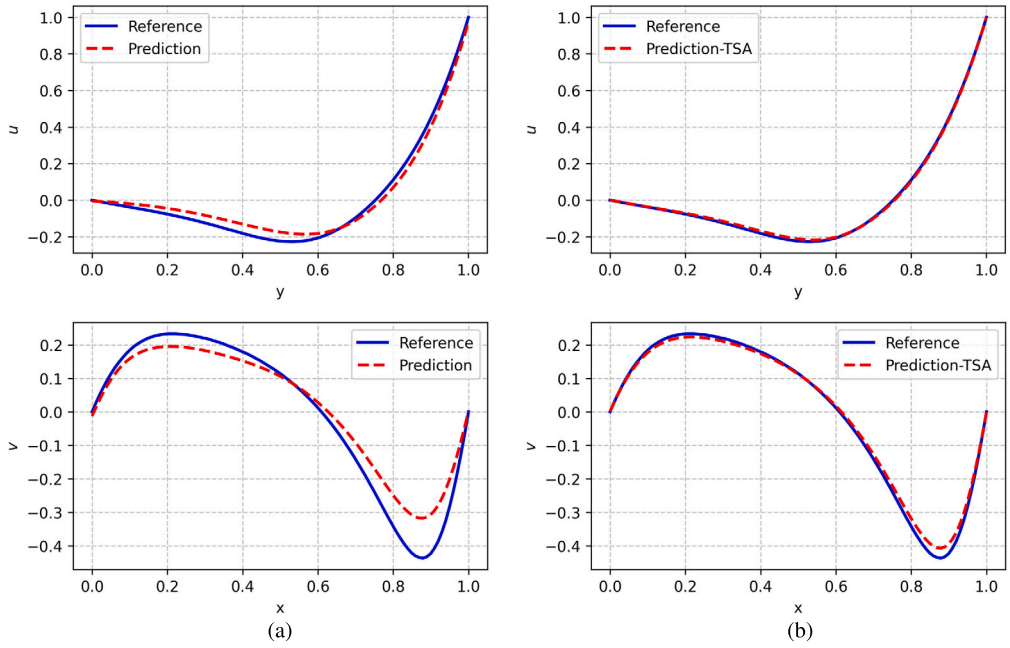


Fig. 4. Lid-driven cavity problem at $Re = 100$: Quantitative comparison of the solutions at $x_{\text{constant}} = 0.7$ and $y_{\text{constant}} = 0.7$. (a) Standard PINN; (b) TSA-PINN.

the standard PINN and TSA-PINN. Although TSA-PINN incurs a slight increase in training time, the significant reduction in approximation error makes it a more robust and efficient option for simulating the complex flow phenomena encountered at $Re = 3200$.

3.3. 2D time-dependent cylinder wake

The second problem is a time-dependent simulation of 2D vortex shedding behind a circular cylinder at $Re = 100$. This problem is governed by the 2D incompressible Navier–Stokes equations (24) and (25). The inlet condition is specified as a free-stream non-dimensional velocity $u_\infty = 1$, and the kinematic viscosity is set to $\nu = 0.01$. The center of the cylinder, with a diameter $D = 1$, is positioned at $(x, y) = (0, 0)$. This configuration gives rise to asymmetrical swirling vortices due to vortex shedding, commonly known as the Karman vortex street. For reference data generation, a high-fidelity DNS approach is used [2]. The dimensions of the spatial domain are $[1, 8] \times [-2, 2]$. The time interval is $[0, 20]$ and the time-step is $\Delta t = 0.01$. The inputs to the neural network model are the spatio-temporal coordinates x, y, and t (all parameters are non-dimensional). The two-dimensional output represents

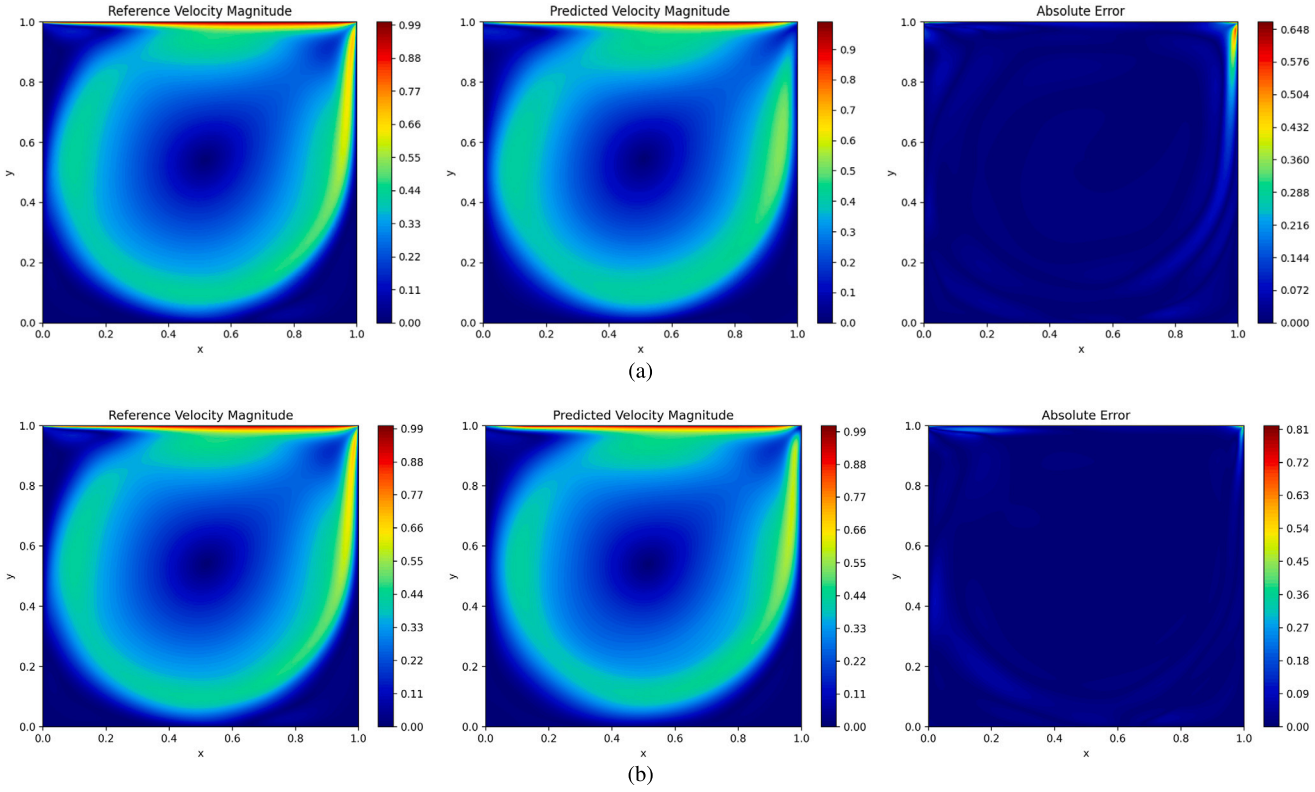


Fig. 5. Lid-driven cavity problem at $Re = 3200$: Velocity contours and approximation error. (a) Standard PINN; (b) TSA-PINN.

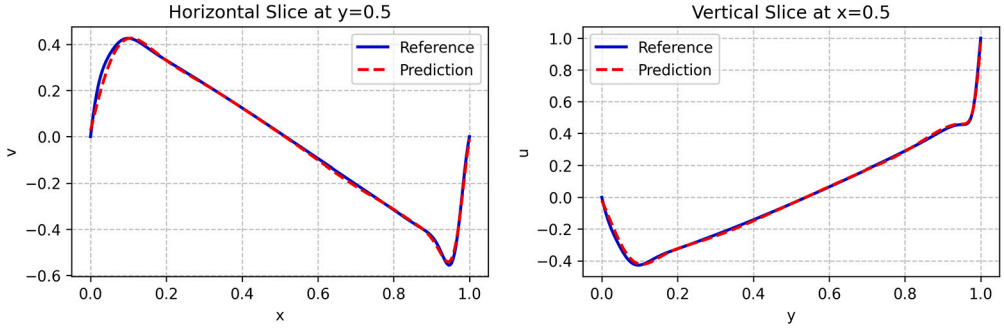


Fig. 6. Lid-driven cavity problem at $Re = 3200$: Quantitative validation of the solution approximations at $x_{\text{constant}} = 0.5$ and $y_{\text{constant}} = 0.5$.

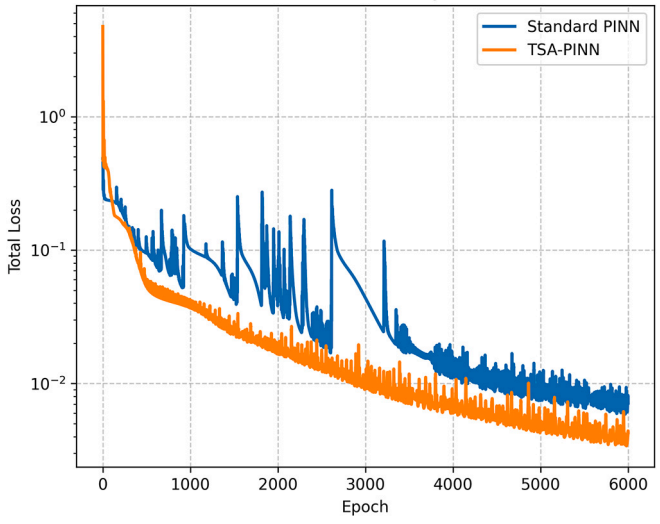


Fig. 7. Lid-driven cavity problem at $Re = 3200$: Comparison of loss histories.

the stream function $\psi(x, y, t)$ and the pressure $p(x, y, t)$. Labeled training data are available within the domain. At each of the 200 time-steps, 1,000 collocation points and 400 boundary condition points are used to evaluate the model's loss. The neural network architecture used for this simulation involves 4 hidden layers with 50 neurons in each layer. The learning rate is configured for exponential decay, starting from an initial value of $\eta = 5 \times 10^{-3}$, with a decay rate of 0.95 every 1,000 steps. The training process used 4,000 iterations. To estimate the L_2 norm of the error during training, a mesh grid of 200×200 is used to estimate the error at points not used during the training at each time-step. The residuals of the governing equations for this problem are:

$$R_x = \partial_t u + u\partial_x u + v\partial_y u + \partial_x p - \frac{1}{Re}(\partial^2_{xx} u + \partial^2_{yy} u); \tag{50}$$

$$R_y = \partial_t v + u\partial_x v + v\partial_y v + \partial_y p - \frac{1}{Re}(\partial^2_{xx} v + \partial^2_{yy} v). \tag{51}$$

First, we analyze the role of the initialization for the trainable frequencies in the TSA-PINN model. Despite the adaptability of frequencies within the TSA-PINN, the initialization influences the optimization. Initial values for the frequencies are better to align with the spectral characteristics of the target function, enabling the optimizer to converge

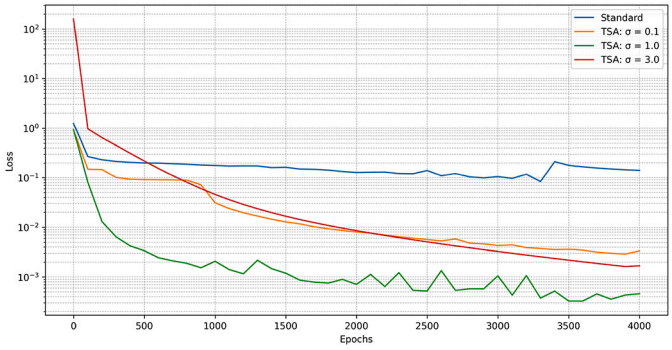


Fig. 8. Cylinder wake problem: Comparison of the loss history.

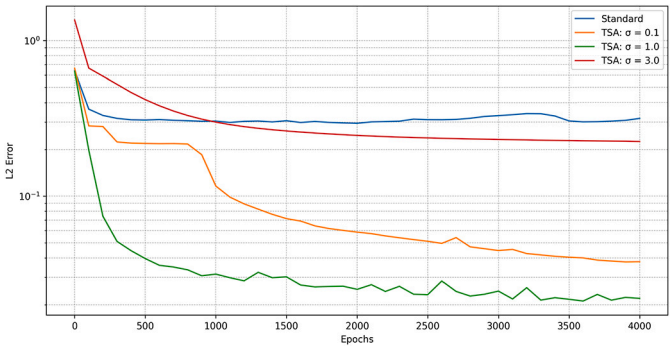


Fig. 9. Cylinder wake problem: Comparison of the error history.

more efficiently towards optimal parameter values. Fig. 8 illustrates the loss decay comparison between the standard PINN and the TSA-PINN initialized with various frequency values σ. The loss history, illustrated in Fig. 8, indicates that TSA-PINN models, when initialized with optimal frequencies, converge more efficiently towards accurate solutions. This observation illustrates the importance of the initial frequency settings. The model with $\sigma = 1.0$ shows the most rapid initial decrease of the loss, suggesting that this value is relatively the optimal frequency for this problem. Models initialized with $\sigma = 0.1$ and $\sigma = 3.0$ converge more slowly compared to the TSA-PINN with $\sigma = 1.0$, indicating a suboptimal starting point, requiring more epochs to adjust towards effective frequency values. Even for the models with $\sigma = 0.1$ and $\sigma = 3.0$, despite not being initialized optimally, their trainable frequencies lead to improved convergence after additional iterations, compared to the standard PINN. This illustrates the capability of the TSA-PINN to adapt and refine frequency parameters dynamically during training.

In Fig. 9 we illustrate the error dynamics of TSA-PINN models with varying σ parameters compared to the standard PINN model. The TSA-PINN configured with $\sigma = 1.0$ consistently exhibits the lowest error rates, leading to rapid error reduction and superior model accuracy. With various initial σ settings, TSA-PINN models systematically outperform the standard PINN, which lacks frequency adaptability. This advantage, evident even when σ values are not optimally set, highlights the superior performance of the TSA-PINN. The robustness and consistent efficacy of TSA-PINN across a range of σ configurations show its superiority over traditional PINN approaches. It is worthy to note that the process of determining optimal values for the initialization of trainable frequencies often involves trial and error, similar to selecting the optimal network size. Both are considered problem-specific hyperparameters.

Fig. 10 shows the velocity contours at $t = 10.0$ s, comparing the predictions from a standard PINN model and a TSA-PINN model against reference data. It is evident that the standard PINN struggles to capture the detailed flow structures around the cylinder, particularly in the wake region, within this limited number of epochs. Conversely, the TSA-PINN model exhibits improved prediction capabilities. The velocity components show better details and very close agreement with the reference data, capturing all features of vortex shedding and wake for-

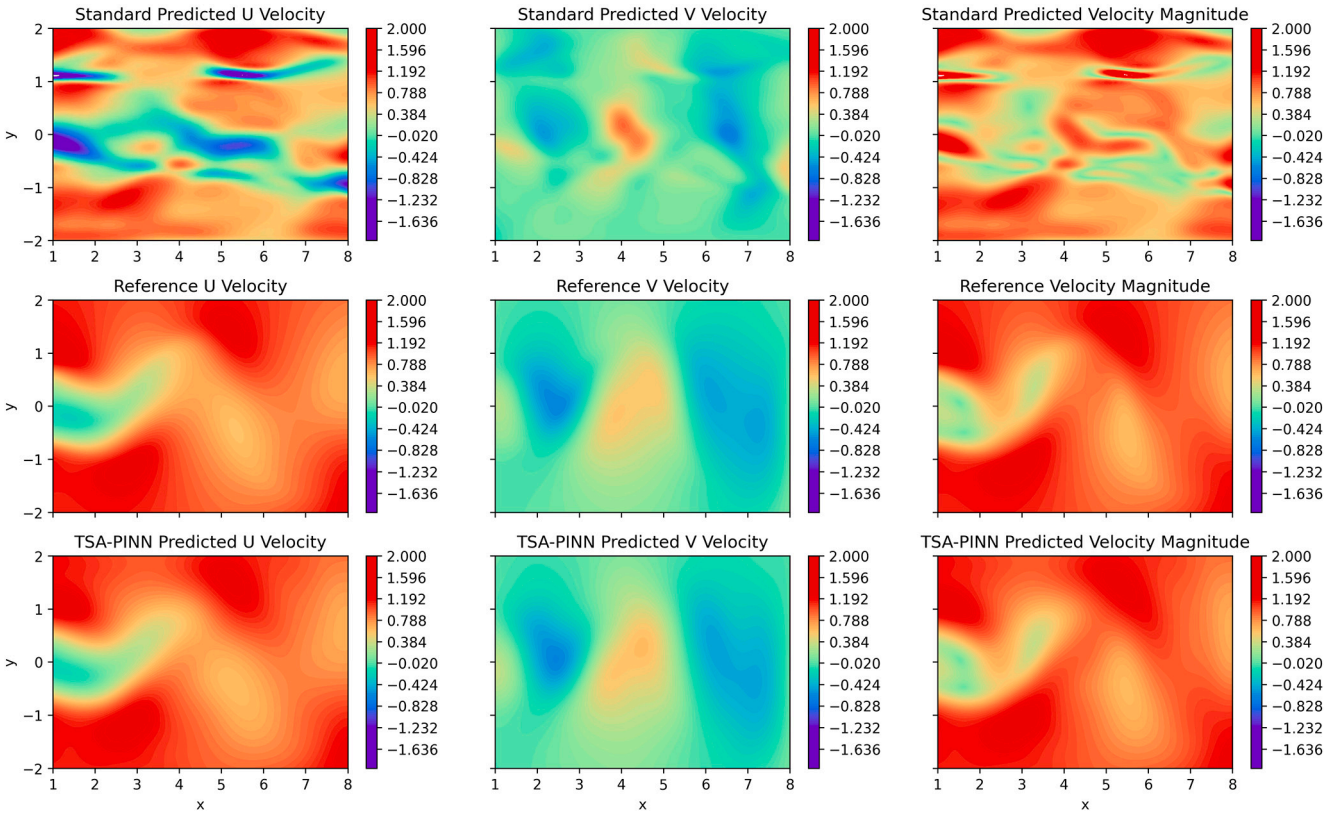


Fig. 10. Cylinder wake problem: Comparison of velocity contours for Standard PINN (top) and TSA-PINN (bottom) against the reference solution (middle).

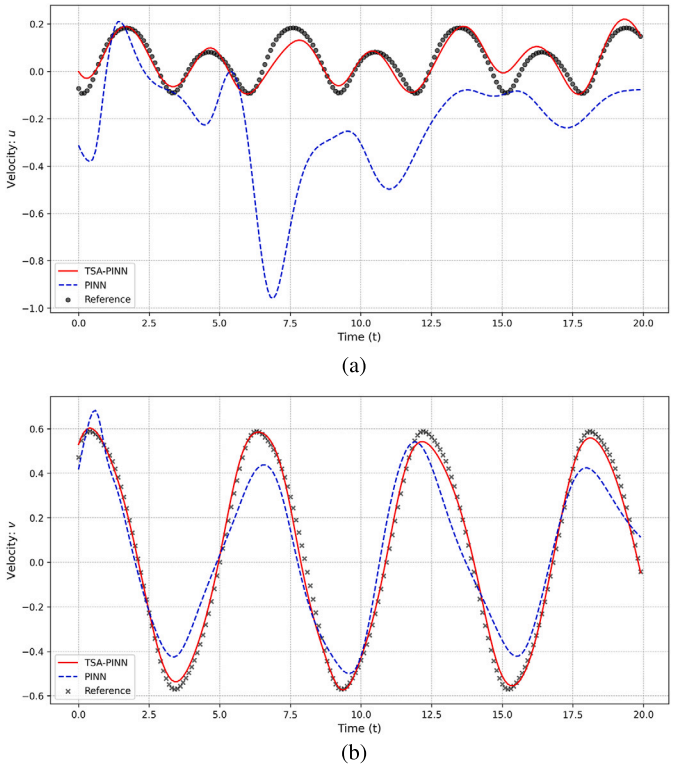


Fig. 11. Cylinder wake problem: Quantitative comparison of the solutions at $x = 2.0$, $y = 0.04$. (a) u vs. t; (b) v vs. t.

Table 3
Error and runtime comparison for the cylinder wake problem: Standard PINN vs. TSA-PINN.

Method	Run-time (s)	L_2-err.: u	L_2-err.: v
PINN	156	6.41×10^{-1}	6.78×10^{-1}
TSA-PINN	201	1.96×10^{-2}	6.63×10^{-2}

mation with high accuracy. This comparative analysis highlights the superior performance of the TSA-PINN in modeling complex fluid dynamics, demonstrating their effectiveness in capturing intricate flow features that the standard PINN fails to resolve, within such a limited number of training epochs.

Fig. 11 illustrated the time evolution of the velocity components at a fixed point in the wake of a cylinder, providing a quantitative comparison of TSA-PINN and standard PINN models against the reference data. This comparison serves as a validation of the models, demonstrating their ability to capture the dynamics of the fluid over time. The TSA-PINN model exhibits remarkable agreement with the reference data, accurately replicating the sinusoidal oscillations in both velocity components, within a very fast and short training process (Table 3). This precision in capturing the amplitude and phase of the flow's oscillations highlights the efficacy of TSA-PINN's trainable sinusoidal activation functions, which are better suited to this type of fluid dynamics problem where the underlying solutions exhibit sinusoidal behavior. In contrast, the standard PINN fails to accurately match the reference solution. The quantitative results from this comparison not only validate the TSA-PINN model, but also underscore its superiority in terms of accuracy and robustness. The enhanced performance of TSA-PINNs shows its ability to adjust its activation frequencies dynamically. Table 3 gives a detailed comparison of the L_2-norm of approximation errors and runtime between standard PINNs and TSA-PINNs. The data clearly shows that the TSA-PINN significantly enhances accuracy, with smaller L_2-errors in both velocity components. Specifically, the L_2-error for the u velocity component of the TSA-PINN is 1.96×10^{-2} compared to 6.41×10^{-1} for the standard PINN, and for the v velocity component, it is 6.63×10^{-2}, compared to 6.78×10^{-1}, demonstrating nearly an order of magnitude improvement in accuracy. This demonstrates that while TSA-PINNs require a merely negligible additional computational resources, the payoff in terms of accuracy is considerable. While the standard PINN converges to a relatively accurate solution, it requires substantially more epochs compared to the TSA-PINN. Our comparative study shows that for a limited training duration, TSA-PINNs achieve comparable precision, much more rapidly. These results establish TSA-PINNs as a more efficient alternative, achieving superior accuracy and making it a better choice for applications demanding high accuracy within limited computational resources.

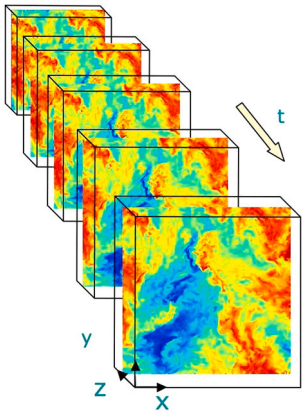


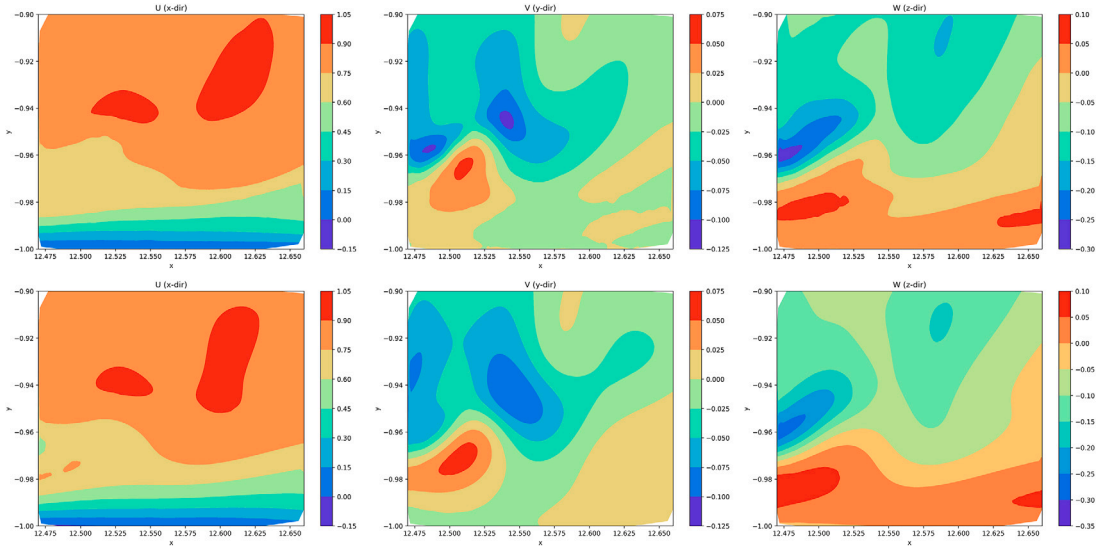
Fig. 12. Schematics of the 3D turbulent channel flow problem [52–54] from https://turbulence.pha.jhu.edu.

3.4. 3D time-dependent turbulent channel flow: near-wall region

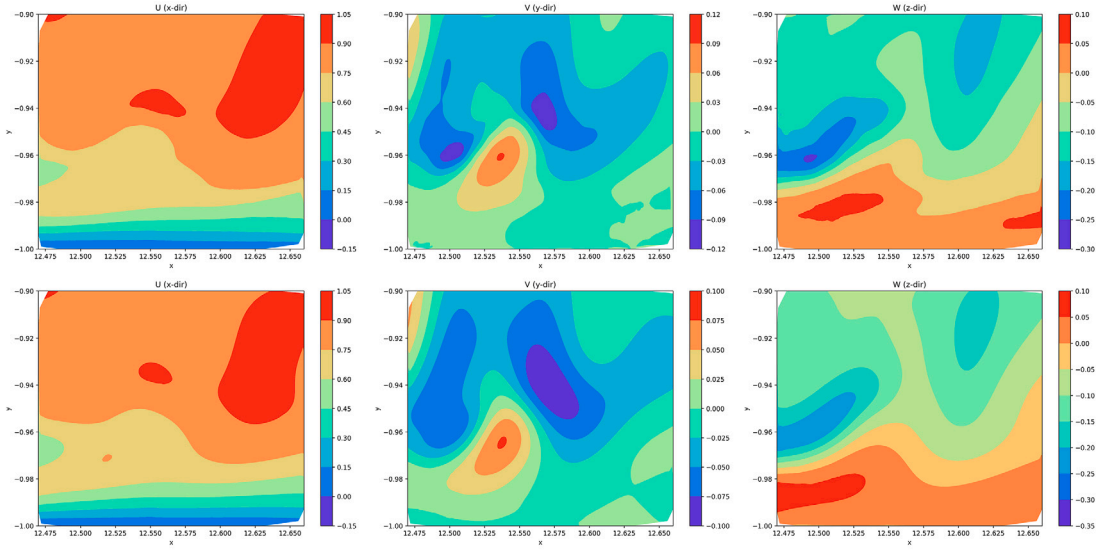
For the third test and validation scenario, we extend the application of our model to a more complex and realistic challenge by conducting a time-dependent simulation of a 3D turbulent channel flow at $Re = 999.4$. This problem is governed by the incompressible Navier–Stokes equations (24) and (25). This case represents a less commonly explored problem in the literature on PINN, applied to fluid dynamics problems, as most studies typically focus on more canonical problems. The training and testing data used for the VP form of the Navier-Stokes equations are sourced from the turbulent channel flow database provided by Perlman et al. [52], Li et al. [53], Graham et al. [54], available at https://turbulence.pha.jhu.edu/.

The spatial domain for the simulation is $[0, 8\pi] \times [-1, 1] \times [0, 3\pi]$, for x, y, and z respectively, with a time-step size of 0.0065 as provided by the online database. The viscosity is $\nu = 5 \times 10^{-5}$, and the initial conditions represent a fully developed turbulence. The network's input and output dimensions consist in four variables each: (x, y, z, t) serve as inputs, while the corresponding outputs are the velocity components and pressure, (u, v, w, P). The schematics of the problem domain are depicted in Fig. 12. To study the performance of the TSA-PINN in a near-wall region, a smaller domain is considered with dimensions $[12.47, 12.66] \times [-1.00, -0.90] \times [4.69, 4.75]$ for x, y, and z respectively. The analysis includes 17 discrete time-steps within this region, focusing on capturing the detailed dynamics near the boundary. According to the description provided by the database [54], the specified region includes the viscous sub-layer, the buffer layer, and the log-law region:

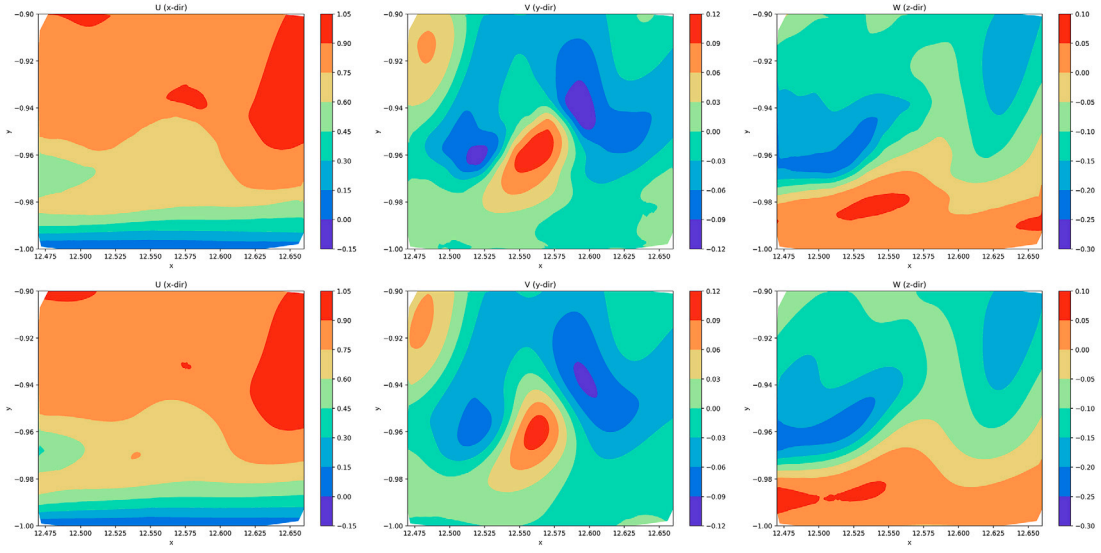
- The *viscous sub-layer*, or laminar sub-layer, is positioned next to the wall, where the flow is predominantly viscous and turbulence is minimal. In this region, the velocity profile is nearly linear, with the shear stress being predominantly viscous. Viscous effects dominate, and the velocity gradient is directly proportional to the shear stress, characterizing this layer as very thin compared to others.



(a)



(b)



(c)

Fig. 13. 3D turbulent channel (Problem 4): Velocity contours at $z_{\text{constant}} = 4.69$. (a) Top: reference solution at $t = 0$; bottom: TSA-PINN at $t = 0$; (b) Top: reference at $t = 5 \times 0.0065$; bottom: TSA-PINN at $t = 5 \times 0.0065$; (c) Top: reference at $t = 10 \times 0.0065$; bottom: TSA-PINN at $t = 10 \times 0.0065$.

Table 4
Error comparison for the 3D turbulent channel (Problem 4): Standard PINN vs. TSA-PINN.

Method	Time-step index	L_2-err.: u component	L_2-err.: v component	L_2-err.: w component
PINN	0	7.31×10^{-2}	1.98×10^{-1}	1.48×10^{-1}
TSA-PINN	0	3.06×10^{-2}	9.63×10^{-2}	8.12×10^{-2}
PINN	5	8.91×10^{-2}	2.48×10^{-1}	1.92×10^{-1}
TSA-PINN	5	3.57×10^{-2}	1.43×10^{-1}	1.08×10^{-1}
PINN	10	1.01×10^{-1}	2.98×10^{-1}	2.48×10^{-1}
TSA-PINN	10	4.26×10^{-2}	1.95×10^{-1}	1.48×10^{-1}

- Located above the viscous sub-layer and below the fully turbulent zone, the *buffer layer* serves as a transition zone where the effects of viscosity and turbulence intermingle. This layer is significant for both viscous and turbulent shears, and it is characterized by a complex velocity profile that neither follows a simple linear nor logarithmic law. It marks the region where turbulence production and dissipation are approximately balanced.
- Located above the *buffer layer*, the *log-law region* is governed by fully turbulent flow, where the viscosity effects are negligible when compared to the inertial effects of turbulence. The mean velocity profile within this region follows a logarithmic distribution as a function of the distance from the wall,

$$u^+ = \frac{1}{k}\log(y^+) + B, \tag{52}$$

where u^+ denotes the non-dimensional velocity, y^+ is the non-dimensional distance from the wall, k is the von Kármán constant (approximately 0.41), and B is a constant dependent on the flow characteristics and surface roughness, here considered to be 5.2.

In this simulation, at each time-step, the model's loss is computed using 10,000 collocation points, 6,644 boundary condition points, and 10,000 initial condition points at the first time-step. The neural network architecture contains 8 hidden layers, each containing 200 neurons. The training use a piece-wise decay of learning rate strategy with the ADAM optimizer, initially set at 10^{-3} for the first 150 epochs, then decaying to 10^{-4} for the next 200 epochs, 5×10^{-5} for another 200 epochs, 5×10^{-6} for an additional 250 epochs, and finally 10^{-6} for the last 150 epochs, with 150 iterations per epoch. The segmented learning strategy in using the ADAM optimizer leverages the benefits of adjusting learning rates at different phases of training. By segmenting the learning strategy, the model initially uses a higher learning rate to converge quickly to a reasonable solution space and then gradually decreases the rate to refine the solution. To assess the L_2-norm of the error post-training, mesh grids of 1500×1500 at $z =$ constant planes are used to evaluate the error at points not encountered during training at each time-step. The residuals of the governing equations for this problem, critical for understanding the dynamics and performance of the model, are formulated in equations (28), (29), (30), and (31).

It is worth noting that, in the turbulent channel flow problems (i.e., the third and fourth test cases), the training duration was kept identical for both TSA-PINN and the standard PINN. Under this constraint, the standard PINN was able to complete approximately 25% more training epochs. Nevertheless, the TSA-PINN consistently outperformed the standard version, demonstrating superior performance.

The comparisons of instantaneous velocity fields with the reference DNS solution and the TSA-PINN predictions are illustrated in Fig. 13 at three different time frames. These comparisons indicate that the TSA-PINN's approximations align well with the reference solution, particularly in the early stages. Due to the initial conditions specified at the first time frame, the accuracy is initially better and tends to decrease as time progresses. This observation is confirmed in Table 4, which also clearly shows the superior efficiency of TSA-PINN when compared to the standard PINN. Furthermore, the performance of the TSA-PINN in capturing the near-wall phenomena (viscous sub-layer, buffer region, and log-law region) is analyzed. Fig. 14 illustrates how the TSA-PINN's approximations for the mean velocity profile correspond to the expected behavior in the near-wall region. The quantitative results show that the model's predictions up to $y^+ = 10$ follow a linear model where $U^+ = y^+$, suggesting a linear relationship between the mean velocity profile and the vertical distance in wall-units. In the range $10 < y^+ < 30$, the buffer layer is observed where the velocity profile becomes more complex and does not adhere to a simple linear or logarithmic law. Beyond $y^+ > 30$, the mean velocity profile predicted by the TSA-PINN adheres to a logarithmic distribution relative to the distance from the wall, confirming that the predictions are consistent with the reference solution [54].

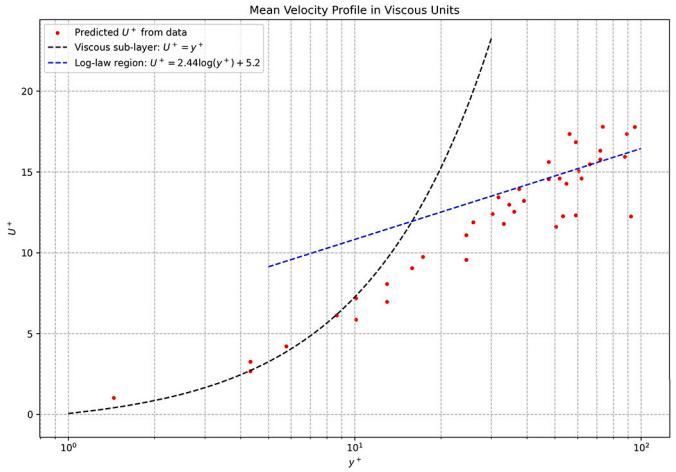


Fig. 14. 3D turbulent channel (Problem 4): Mean velocity profile in viscous units. Standard values of $k = 0.41$ and $B = 5.2$ are used in the log-law for reference.

3.5. 3D time-dependent turbulent channel flow: over a larger domain

As an extension of case 4, this final validation and testing scenario (case 5) involves a substantially larger domain, covering half of the channel height. This subdomain $[12.47, 12.66] \times [-1.00, -0.0031] \times [4.61, 4.82]$ for x, y, and z respectively, includes analysis over 8 discrete time-steps. The larger domain facilitates interactions among diverse scales of eddies, with significant computational challenges. The complexity introduced by these varying scales necessitates an increased number of loss evaluation points, thereby demanding more computational resources and power. At each time increment, the evaluation of the model's loss incorporates 40,000 collocation points, 26,048 boundary condition points, and 147,968 initial condition points at the first time-step. The computational architecture used consists of 10 hidden layers, each with 300 neurons. A piece wise decay of the learning rate strategy is used with the ADAM optimizer, with a learning rate of 10^{-3} for the initial 100 epochs, then sequentially decreasing to 10^{-4} for 250 epochs, 10^{-5} for another 250 epochs, 5×10^{-6} for an additional 250 epochs, and ultimately with to 10^{-6} for the final 150 epochs, maintaining 150 iterations per epoch.

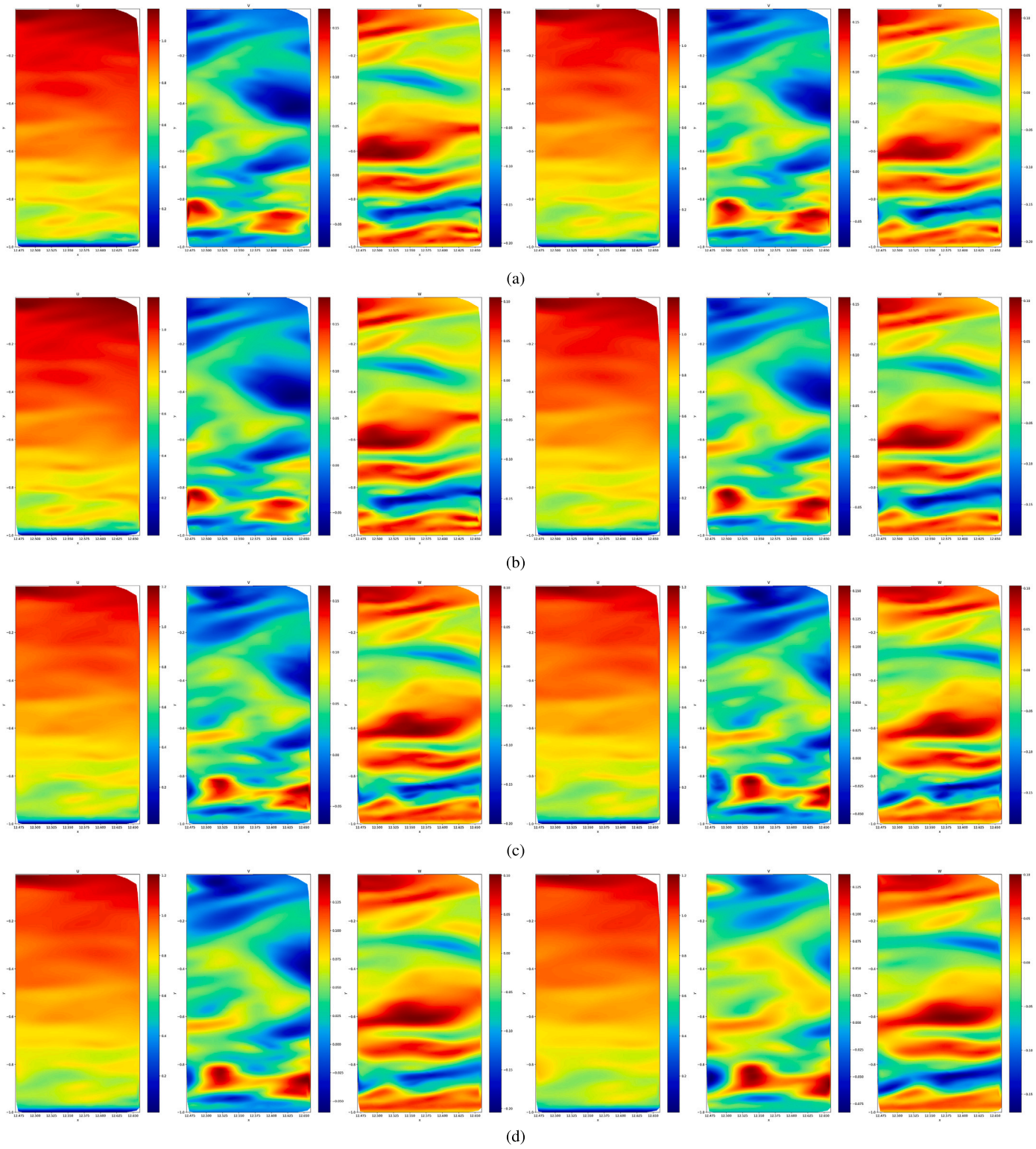


Fig. 15. 3D turbulent channel (Problem 5): Velocity contours at $z_{\text{constant}} = 4.61$. (a) Reference solution at $t = 0$ (left) and $t = 4 \times 0.0065$ (right); (b) TSA-PINN at $t = 0$ (left) and $t = 4 \times 0.0065$ (right); (c) Reference at $t = 8 \times 0.0065$ (left) and $t = 12 \times 0.0065$ (right); (d) TSA-PINN at $t = 8 \times 0.0065$ (left) and $t = 12 \times 0.0065$ (right).

It is worth mentioning that the wall-normal and span-wise velocities exhibit greater relative L_2-errors when compared to the stream-wise velocity. This is because of the substantially larger values of the stream-wise velocity relative to the other two components. Fig. 15 illustrates comparisons of instantaneous velocity fields between the reference DNS solution and TSA-PINN approximations for four different time frames. The TSA-PINN's approximations align well with the reference solution, especially at time frames within the training range. The final time-step, $t = 12 \times 0.0065$, extends beyond the training scope of $t \in [0, 8 \times 0.0065]$. Therefore, TSA-PINN's predictions are highly accurate within the interpolation domain and remain acceptably accurate in the extrapolation phase.

Fig. 16 and Table 5 quantitatively confirm this observation where the L_2-norm of the errors associated with the models' numerical approxi-

Table 5
Error comparison for the 3D turbulent channel (Problem 5): Standard PINN vs. TSA-PINN.

Method	Time-step index	L_2-err.: u component	L_2-err.: v component	L_2-err.: w component
PINN	0	6.25×10^{-3}	1.17×10^{-1}	1.08×10^{-1}
TSA-PINN	0	3.95×10^{-3}	6.57×10^{-2}	5.49×10^{-2}
PINN	4	5.41×10^{-3}	9.44×10^{-2}	9.83×10^{-2}
TSA-PINN	4	3.48×10^{-3}	5.43×10^{-2}	5.57×10^{-2}
PINN	8	6.91×10^{-3}	1.30×10^{-1}	1.35×10^{-1}
TSA-PINN	8	5.17×10^{-3}	8.24×10^{-2}	8.72×10^{-2}
PINN	12	1.00×10^{-2}	2.10×10^{-1}	1.85×10^{-1}
TSA-PINN	12	8.68×10^{-3}	1.74×10^{-1}	1.49×10^{-1}
PINN	16	1.46×10^{-2}	3.43×10^{-1}	3.05×10^{-1}
TSA-PINN	16	1.80×10^{-2}	3.15×10^{-1}	2.70×10^{-1}

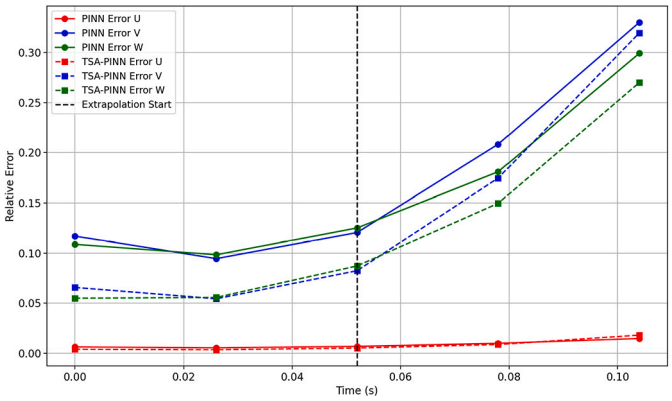


Fig. 16. 3D turbulent channel (Problem 5): Comparison of the error history.

mations are compared to the reference data. Within the interpolation domain, TSA-PINN's approximations are closely aligned with the reference solution, maintaining the error below 10^{-3} for u and 10^{-2} for v and w. Beyond the training domain, as illustrated using a vertical dashed line at 0.06 seconds in Fig. 16, an increase in error rates for all velocity components is observed. The error, which had remained low within the interpolation region, grows sharply post-extrapolation. This larger rise in error highlights the challenges associated with generalizing data beyond the temporal bounds of the training data. Although the standard PINN exhibits similar trends, the TSA-PINN performs better, confirming its superior efficiency and effectiveness. These analyses show the benefits of TSA-PINNs over standard PINNs, particularly their enhanced capability to accurately capture complex fluid dynamics.

4. Conclusion

In this work, we introduced a transformative approach to approximate the solutions to the Navier-Stokes equations using a novel Physics-Informed Neural Network model, incorporating a Trainable Sinusoidal Activation Function (TSA-PINN). Central to our approach is the integration of an innovative trainable neuron-wise sinusoidal activation mechanism and a dynamic slope recovery mechanism. The initial alignment of activation frequencies with the target function, coupled with their trainable nature, allows for continuous optimization during training, ensuring that the model dynamically converges to optimal settings. The slope recovery mechanism, an addition to our model, adjusts the frequencies of the activation functions in real time, stabilizing the gradient flow and preventing issues such as gradients vanishing or exploding. This leads to faster convergence, and bolsters the model's robustness. Our numerical experiments span from steady-state two-dimensional to time-dependent three-dimensional turbulent flows with high Reynolds numbers. In the first and second validation experiments—lid-driven cavity flows at $Re = 100$ and $Re = 3200$—TSA-PINN demonstrated good agreement with the reference solutions, achieving improved accuracy and more effective convergence compared to the standard PINN. The third test case, the cylinder wake problem, showcased that the TSA-PINN is even more efficient in scenarios where the target phenomena exhibit an oscillating behavior. This efficiency is attributed to the sinusoidal nature (a combination of sine and cosine functions) of the activation function and of the trainable frequency, allowing the model to adapt dynamically to the oscillatory nature of the flow dynamics. We went beyond the "canonical" problems to study less-explored, more challenging and realistic models, where other methods often struggle to find accurate approximations. This allowed us to show that the TSA-PINN provides improvements over PINN models. The TSA-PINN's ability to dynamically adjust the frequencies in sets of neuron-specific sinusoidal activation functions allows it to effectively capture flow dynamics that are typically challenging, especially in high-dimensional chaotic systems where other models often fail. Our findings show that TSA-PINNs not only deliver superior accuracy, but also achieve this with fewer training epochs when compared to other methods. Specifically, our application of TSA-PINNs to three-dimensional, time-dependent turbulent channel flows illustrates their proficiency in capturing complex phenomena, such as rapidly-changing behaviors near walls (as seen in the 4th test case) and the nonlinear behaviors arising at various scales of eddies (as showed in the 5th test case). Future work will focus on further optimizing the architecture and training processes of TSA-PINNs and expanding the application scope to study additional physical systems. This research not only advances the field of physics-informed machine learning, but also paves the way for more sophisticated, efficient, and accurate models, using advanced and dynamic activation functions in scientific computing and engineering simulations.

CRediT authorship contribution statement

Amirhossein Khademi: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Steven Dufour:** Writing – review & editing, Supervision.

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

All the datasets and source codes to reproduce the results in this study are available on GitHub at https://github.com/AmirhosseinnnKhademi/TSA-PINN

References

[1] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part i): data-driven solutions of nonlinear partial differential equations, arXiv preprint, arXiv:1711.10561, 2017.

[2] 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.

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

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

[5] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, Comput. Methods Appl. Mech. Eng. 365 (2020) 113028.

[6] A.D. Jagtap, G.E. Karniadakis, Extended physics-informed neural networks (xpinns): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, in: AAAI Spring Symposium: MLPS, 2021, pp. 2002–2041.

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

[8] M. Penwarden, A.D. Jagtap, S. Zhe, G.E. Karniadakis, R.M. Kirby, A unified scalable framework for causal sweeping strategies for physics-informed neural networks (pinns) and their temporal decompositions, J. Comput. Phys. 493 (2023) 112464.

[9] A. Khademi, S. Dufour, A novel discretized physics-informed neural network model applied to the Navier-Stokes equations, Phys. Scr. (2024).

[10] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (2021) A3055–A3081.

[11] E. Lorin, X. Yang, Time-dependent Dirac equation with physics-informed neural networks: computation and properties, Comput. Phys. Commun. 280 (2022) 108474.

[12] E. Lorin, X. Yang, Quasi-optimal domain decomposition method for neural network-based computation of the time-dependent Schrödinger equation, Comput. Phys. Commun. 299 (2024) 109129.

[13] D. Zhang, Y. Li, S. Ying, Trans-net: a transferable pretrained neural networks based on temporal domain decomposition for solving partial differential equations, Comput. Phys. Commun. 299 (2024) 109130.

[14] P. Ren, C. Rao, S. Chen, J.-X. Wang, H. Sun, Y. Liu, Seismicnet: physics-informed neural networks for seismic wave modeling in semi-infinite domain, Comput. Phys. Commun. 295 (2024) 109010.

[15] T. Zhang, B. Dey, P. Kakkar, A. Dasgupta, A. Chakraborty, Frequency-compensated pinns for fluid-dynamic design problems, arXiv preprint, arXiv:2011.01456, 2020.

[16] Q. Lou, X. Meng, G.E. Karniadakis, Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-bgk formulation, J. Comput. Phys. 447 (2021) 110676.

[17] P.-H. Chiu, J.C. Wong, C. Ooi, M.H. Dao, Y.-S. Ong, Can-pinn: a fast physics-informed neural network based on coupled-automatic–numerical differentiation method, Comput. Methods Appl. Mech. Eng. 395 (2022) 114909.

[18] A.F. Psaros, K. Kawaguchi, G.E. Karniadakis, Meta-learning pinn loss functions, J. Comput. Phys. 458 (2022) 111121.

[19] M. Penwarden, S. Zhe, A. Narayan, R.M. Kirby, A metalearning approach for physics-informed neural networks (pinns): application to parameterized pdes, J. Comput. Phys. 477 (2023) 111912.

[20] J. Oldenburg, F. Borowski, A. Öner, K.-P. Schmitz, M. Stiehm, Geometry aware physics informed neural network surrogate for solving Navier–Stokes equation (gapinn), Adv. Model. Simul. Eng. Sci. 9 (2022) 8.

[21] N. Wandel, M. Weinmann, M. Neidlin, R. Klein, Spline-pinn: approaching pdes without data using fast, physics-informed Hermite-spline cnns, Proc. AAAI Conf. Artif. Intell. 36 (2022) 8529–8538.

[22] S. Wang, H. Wang, P. Perdikaris, Improved architectures and training algorithms for deep operator networks, J. Sci. Comput. 92 (2022) 35.

[23] O. Hennigh, S. Narasimhan, M.A. Nabian, A. Subramaniam, K. Tangsali, Z. Fang, M. Rietmann, W. Byeon, S. Choudhry, Nvidia simnet™: an ai-accelerated multi-physics simulation framework, in: International Conference on Computational Science, Springer, 2021, pp. 447–461.

[24] A. Kendall, Y. Gal, R. Cipolla, Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7482–7491.

[25] Y. Liu, W. Liu, X. Yan, S. Guo, C.-a. Zhang, Adaptive transfer learning for pinn, J. Comput. Phys. 490 (2023) 112291.

[26] S. Wang, S. Sankaran, P. Perdikaris, Respecting causality for training physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 421 (2024) 116813.

[27] Y. Song, H. Wang, H. Yang, M.L. Taccari, X. Chen, Loss-attentional physics-informed neural networks, J. Comput. Phys. 501 (2024) 112781.

[28] H. Eivazi, M. Tahani, P. Schlatter, R. Vinuesa, Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations, Phys. Fluids 34 (2022).

[29] H. Xu, W. Zhang, Y. Wang, Explore missing flow dynamics by physics-informed deep learning: the parameterized governing systems, Phys. Fluids 33 (2021).

[30] C. Cheng, H. Meng, Y.-Z. Li, G.-T. Zhang, Deep learning based on pinn for solving 2 dof vortex induced vibration of cylinder, Ocean Eng. 240 (2021) 109932.

[31] S. Ghosh, A. Chakraborty, G.O. Brikis, B. Dey, Rans-pinn based simulation surrogates for predicting turbulent flows, arXiv preprint, arXiv:2306.06034, 2023.

[32] Y. Patel, V. Mons, O. Marquet, G. Rigas, Turbulence model augmented physics-informed neural networks for mean-flow reconstruction, Phys. Rev. Fluids 9 (2024) 034605.

[33] R. Wang, K. Kashinath, M. Mustafa, A. Albert, R. Yu, Towards physics-informed deep learning for turbulent flow prediction, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1457–1466.

[34] L. Sliwinski, G. Rigas, Mean flow reconstruction of unsteady flows using physics-informed neural networks, Data-Cent. Eng. 4 (2023) e4.

[35] X. Jin, S. Cai, H. Li, G.E. Karniadakis, Nsfnets (Navier-Stokes flow nets): physics-informed neural networks for the incompressible Navier-Stokes equations, J. Comput. Phys. 426 (2021) 109951.

[36] A. Khademi, E. Salari, S. Dufour, Simulation of 3d turbulent flows using a discretized generative model physics-informed neural networks, Int. J. Non-Linear Mech. 170 (2025) 104988.

[37] C.-C. Yu, Y.-C. Tang, B.-D. Liu, An adaptive activation function for multilayer feedforward neural networks, in: 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM'02. Proceedings, vol. 1, IEEE, 2002, pp. 645–650.

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

[39] M. Dushkoff, R. Ptucha, Adaptive activation functions for deep networks, Electron. Imaging 28 (2016) 1–5.

[40] J. Abbasi, P.Ø. Andersen, Physical activation functions (pafs): an approach for more efficient induction of physics into physics-informed neural networks (pinns), Neurocomputing (2024) 128352.

[41] 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.

[42] A.D. Jagtap, K. Kawaguchi, G. Em Karniadakis, Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, Proc. Royal Soc. A 476 (2020) 20200334.

[43] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.

[44] D.S. Berman, A.L. Buczak, J.S. Chavis, C.L. Corbett, A survey of deep learning methods for cyber security, Information 10 (2019) 122.

[45] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics–informed neural networks: where we are and what's next, J. Sci. Comput. 92 (2022) 88.

[46] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018) 1–43.

[47] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[48] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.

[49] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., {TensorFlow}: a system for {Large-Scale} machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, pp. 265–283.

[50] C.-H. Bruneau, M. Saad, The 2d lid-driven cavity problem revisited, Comput. Fluids 35 (2006) 326–348.

[51] S. Wang, B. Li, Y. Chen, P. Perdikaris, Piratenets: physics-informed deep learning with residual adaptive networks, J. Mach. Learn. Res. 25 (2024) 1–51.

[52] E. Perlman, R. Burns, Y. Li, C. Meneveau, Data exploration of turbulence simulations using a database cluster, in: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, 2007, pp. 1–11.

[53] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, G. Eyink, A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence, J. Turbul. (2008) N31.

[54] J. Graham, K. Kanov, X. Yang, M. Lee, N. Malaya, C. Lalescu, R. Burns, G. Eyink, A. Szalay, R. Moser, et al., A web services accessible database of turbulent channel flow and its use for testing a new integral wall model for les, J. Turbul. 17 (2016) 181–215.