POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Design of a Physics-Informed Neural Network to Approximate Solutions of the Navier-Stokes Equations

AMIRHOSSEIN KHADEMI
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques de l'ingénierie

Avril 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Design of a Physics-Informed Neural Network to Approximate Solutions of the Navier-Stokes Equations

présentée par **Amirhossein KHADEMI**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Youssef DIOUANE, président
Steven DUFOUR, membre et directeur de recherche
Yann-Meing LAW-KAM-CIO, membre
Philippe MIRON, membre externe

DEDICATION

To Jayran, for her love that is beyond the power of words...
To my parents, Mehraneh and Mahdi, for their love without any expectations...
To my brothers, Masoud and Mohammadreza, for being my best friends forever...

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research director, Prof. Steven Dufour, for providing me with the opportunity to embark on this once-in-a-lifetime journey. I appreciate your guidance in focusing on this field of science, as well as your patience and support throughout my research.

Special thanks to Prof. Serge Prudhomme for his valuable time and compassionate advice beyond course hours.

I am also grateful to Prof. Mahdi Soltanpour for his encouragement and belief in my capabilities, and to Dr. Erfan Salari for all the fruitful discussions we shared.

Last but not least, I would like to thank my committee members for dedicating their time to read this manuscript and for their presence at my defense.

RÉSUMÉ

Cette thèse introduit des avancées dans l'application de l'apprentissage profond (Deep Learning, DL), notamment à travers les réseaux de neurones informés par la physique (Physics-Informed Neural Networks, PINN), pour approcher les solutions des équations de Navier-Stokes, qui sont cruciales dans l'étude de la dynamique des fluides. En exploitant les principes de l'apprentissage machine combinés aux lois physiques, cette recherche aborde les défis posés par les méthodes numériques traditionnelles pour approximer les solutions aux problèmes complexes de dynamique des fluides. L'innovation principale de ce travail réside dans le développement et la mise en œuvre de techniques basées sur les PINN améliorées qui intègrent la décomposition de domaine, des architectures de réseaux avancées incluant des réseaux transformateurs, une planification temporelle combinée à un modélisation générative, et de nouveaux mécanismes d'activation au sein du modèle PINN. Cette approche ne se contente pas d'approximer précisément les solutions aux équations aux dérivées partielles (EDP) dans un cadre complet, mais améliore également la capacité des modèles PINN à gérer des flux à haute dimension, chaotiques et turbulents, typiquement rencontrés en dynamique des fluides, régis par les équations de Navier-Stokes. La dissertation est structurée autour de trois articles clés, chacun contribuant au thème général de l'amélioration des PINN pour la dynamique des fluides:

1. **Décomposition de Domaine et Réseaux Transformateurs :** Cette étude approfondit une technique de décomposition de domaine combinée à des réseaux transformateurs qui projettent les espaces d'entrée dans des espaces de caractéristiques de plus haute dimension. Cela facilite des modèles plus flexibles et localement adaptatifs, favorisant une convergence plus rapide vers une solution précise.

2. **Discrétisation Temporelle et Modélisation Générative :** Ce segment introduit la discrétisation temporelle combinée à une modélisation générative pour générer et propager des informations à travers les étapes temporelles, améliorant ainsi l'adaptabilité du PINN aux systèmes fluidiques dynamiques.

3. **Mécanisme d'Activation Sinusoïdal Entraînable :** Le dernier article discute de l'intégration d'un mécanisme d'activation sinusoïdal entraînable au sein du PINN. Les fréquences neuronales du mécanisme d'activation sinusoïdal sont entraînées par une approche de descente de gradient, permettant au modèle de s'aligner rapidement avec la fonction cible et de capturer des comportements chaotiques et oscillants, avec une

performance nettement améliorée.

À travers des expériences computationnelles rigoureuses, incluant des applications à des problèmes de référence tels que la cavité entraînée par le couvercle et le sillage de cylindre, cette recherche démontre l'efficacité des méthodes proposées pour fournir des solutions précises, efficaces et robustes. Les résultats montrent des améliorations significatives dans l'adaptabilité des modèles, les taux de convergence, et la capacité de capturer des comportements fluides complexes, étendant ainsi l'applicabilité des PINN à des scénarios de dynamique des fluides plus exigeants et réalistes. Les problèmes de validation dans cette étude élargissent l'utilisation des méthodes basées sur les PINN des problèmes canoniques traditionnels à des scénarios de dynamique des fluides plus complexes. Plus précisément, l'approche est appliquée à deux scénarios de test au sein d'un écoulement canalisé turbulent en 3D dépendant du temps : une région proche de la paroi pour une analyse détaillée des phénomènes proche de la paroi, tels que la sous-couche visqueuse et la région de la loi logarithmique, et un domaine plus large conçu pour simuler divers phénomènes influencés par des tourbillons à échelles multiples. En résumé, cette thèse non seulement fait avancer l'état de l'art dans l'application de l'apprentissage profond à la dynamique des fluides mais ouvre également de nouvelles voies pour l'exploration de problèmes complexes à haute dimension où les techniques de modélisation basées sur l'apprentissage profond traditionnelles sont insuffisantes. L'applicabilité étendue des modèles PINN démontrée dans ce travail suggère un avenir prometteur pour leur utilisation dans un éventail plus large de problèmes scientifiques et d'ingénierie.

ABSTRACT

This dissertation introduces advancements in the application of Deep Learning (DL) techniques, particularly through Physics-Informed Neural Networks (PINN), to approximate solutions to the Navier-Stokes equations, which are of high importance for the study of fluid dynamics. By leveraging the principles of machine learning, combined with physical laws, this research addresses the shortcomings posed by traditional numerical methods for modeling complex fluid dynamics problems. The core innovation of this work lies in the development and implementation of enhanced PINN-based techniques that integrate domain decomposition, advanced network architectures including transformer networks, time marching combined with generative modeling, and novel trainable activation mechanisms. This approach not only gives accurate approximations to Partial Differential Equations (PDEs), but also enhances the capability of PINN models to handle high-dimensional, nonlinear, and turbulent flows typically encountered in fluid dynamics, governed by the Navier-Stokes equations. The thesis is structured around three articles, each contributing to the overall theme of enhancing PINNs for fluid dynamics:

1. **Domain Decomposition and Transformer Networks:** this study elaborates on a domain decomposition technique, combined with transformer networks that project input spaces into higher-dimensional feature spaces. This facilitates more flexible and locally adaptive models, promoting faster convergence to an accurate approximation.

2. **Temporal Discretization and Generative Modeling:** this paper introduces a time discretization strategy combined with generative modeling to generate and propagate information across time steps, thereby improving the adaptability of the PINN to dynamical fluid systems.

3. **Trainable Sinusoidal Activation Mechanism:** the final article proposes the integration of trainable sinusoidal activation mechanism within PINNs. The neuron-wise frequencies are trained using a gradient descent approach, advancing the model to rapidly align with the target function and capture nonlinear and oscillating behaviors, with a significantly improved performance.

Through rigorous computational experiments, including applications of benchmark problems such as the lid-driven cavity problem and the cylinder wake problem, this research shows the effectiveness of the proposed methods in providing accurate, efficient, and robust approximations. The results show significant improvements in model adaptability, convergence

rates, and the ability to capture complex fluid behaviors, thus extending the applicability of PINNs to more challenging and realistic fluid dynamics scenarios. The validation problems in this study expand the use of PINN-based methods from traditional canonical problems to more complex fluid flows. The methodology is applied to two 3D time-dependent turbulent channel flows: a smaller near-wall region problem for analyzing near-wall phenomena, such as the viscous sub-layer and the log-law region, and a larger domain designed to study various phenomena influenced by multi-scale eddies. In summary, this thesis enhances current methodologies in deploying deep learning for fluid dynamics challenges and paves the way for more advanced approaches in tackling complex, high-dimensional problems where conventional deep learning models fall short. The extended applicability of PINN models, as shown in this work, suggests a promising future for their use in a broader range of scientific and engineering problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Networks
BC	Boundary Conditions
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
DD-PINN	Domain Decomposition Physics-Informed Neural Network
DG-PINN	Discretized Generative Physics-Informed Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DNS	Direct Numerical Simulation
DOF	Degree Of Freedom
DT	Decision Tree
FEM	Finite Element Method
FFNN	Feed Forward Neural Network
FC-NN	Fully Connected Neural Network
GAN	Generative Adversarial Network
IC	Initial Conditions
LES	Large Eddy Simulation
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NN	Neural Networks
NTK	Neural Tangent Kernel
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
ReLU	Rectified Linear Unit
RANS	Reynolds Averaged Navier-Stokes
RF	Random Forest
RL	Reinforcement Learning
ROM	Reduced Order Modeling
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SIMPLE	Semi-Implicit Method for Pressure Linked Equations

SVM	Support Vector Machine
TRF	Transformer Networks
TSA	Trainable Sinusoidal Activation
TSA-PINN	Trainable Sinusoidal Activation Physics-Informed Neural Network

CHAPTER 1 INTRODUCTION

Fluid mechanics has long had to manage vast arrays of data, stemming from experimental work, field data, and extensive numerical analyses. Recent decades have seen a surge of the field known as "big data" within this discipline, driven by enhanced computing capabilities and sophisticated experimental techniques. Today's scientific landscape is marked by an explosion in data across various fields, coupled with advances in computational technologies, reduced computational and storage costs, abundant open-source resources, and benchmark problems. These developments are complemented by substantial industry investments in data-driven techniques, recently increased interest in using Machine Learning (ML) to study fluid mechanics. Machine learning is offering new ways to handle problems such as nonlinear regression, reduced-order modeling, processing experimental data, optimizing shapes, modeling turbulence closures, and controlling processes. This shift towards data-centric methods resemble the mid-20th century's pivot to numerical solutions for fluid dynamics equations. However, it is crucial to balance enthusiasm for ML, knowing that its integration into fluid mechanics presents ongoing hurdles.

1.1 Context

1.1.1 Machine Learning

ML [4] involves estimating relationships among inputs, outputs, and system parameters based on limited data. We distinguish between the system generating data samples and the algorithms. It is important to understand that ML estimates are fundamentally probabilistic. The learning process primarily focuses on minimizing a risk functional,

$$R(w) = \int L[y, \phi(x, y, w)] p(x, y)\, dx\, dy, \tag{1.1}$$

where x (inputs) and y (outputs) are samples from the distribution p, $\phi(x, y, w)$ outlines the model's structure and the parameters w, and the loss function L is designed to find the best approximations with the highest possible accuracy. This risk functional is governed by the probability distribution $p(x, y)$. Learning algorithms are generally classified into three types: supervised, unsupervised, and semisupervised, differentiated by the level of external guidance accessible to the model.

1.1.2 Supervised Learning

Supervised learning [5] leverages corrective data for training the model. Commonly, this means using labeled training data where labels match the model's output. The primary goal is to minimize the cost function dependent on this training data to determine the model's unknown parameters. Historically linked to methods of regression and interpolation introduced by Gauss, supervised learning often employs a loss function expressed as

$$L[y, \phi(x, y, w)] = \|y - \phi(x, y, w)\|^2. \tag{1.2}$$

Key tasks in supervised learning include:

- Classification, using methods such as Naive Bayes Classifiers [6], Support Vector Machines (SVM) [7], Decision Trees (DT) [8], and Random Forests (RF) [9], etc;
- Regression, including Linear Regression [10], Neural Network Regression [11], and Support Vector Regression [12].

1.1.3 Unsupervised Learning

Unsupervised learning [13] involves automatically discovering the underlying patterns in data without external guidance or ground-truth labels. It focuses on feature extraction through global criteria, dealing with tasks like dimensionality reduction, quantization, and clustering. Common approaches in unsupervised learning include:

- Dimensionality reduction, such as proper orthogonal decomposition, principal component analysis, and autoencoders;
- Clustering and vector quantization;
- Anomaly detection.

1.1.4 Semisupervised Learning

Semisupervised learning [14] mixes supervised and unsupervised approaches, typically using a limited amount of labeled data alongside a larger pool of unlabeled data. This category includes:

- Generative adversarial networks (GANs);

- Reinforcement learning (RL), where models learn through rewards and penalties in a simulated setting.

1.1.5 Physics-Informed Neural Networks

While data-driven frameworks have achieved significant success in numerous domains, they function as black-box models without embedded physical constraints. Such methodologies excel at exploring multidimensional physical problems by capturing complex spatio-temporal relationships that remain elusive due to inherent complexity. Despite performing admirably within their training domains, these models encounter substantial limitations: (1) they require extensive datasets, which may not be readily available, and substantial computational resources to synthesize relationships; (2) their performance beyond the training scope raises reliability concerns, potentially leading to significant discrepancies in model generalization. Although these models are not inherently oriented towards physical principles, a substantial dataset can still guide the machine learning model to discern physical relationships. However, with limited data, this approach can be faced with numerous challenges. This is therefore important to integrate physical laws into the machine learning model to foster the development of physics-guided phenomena. Addressing the shortcomings of data-driven models, particularly in scenarios with sparse data, has spurred the development of an enhanced machine learning framework that embeds the physical characteristics of systems. This method, known as the Physics-Informed Machine Learning (PIML) [15], or the PINN, incorporates physical laws, including the partial differential equations, directly within the neural network to foster models that are consistent with physical reality. The primary advantages of using PINN models includes their universal approximation capability to represent any PDE without the need for discretization steps, enhanced generalization across different scenarios, meaningful physical interpretations of the approximations, mitigation of the curse of dimensionality, and reduced dependence on extensive training datasets. In the PINN configurations, various types of PDEs, such as stochastic, differential, integer-order, and integro-differential equations, along with their respective initial conditions (ICs) and boundary conditions (BCs), are integrated into the neural network's loss function,

$$L_{\text{PINN}} = L_{\text{PDE}} + L_{\text{data}} + L_{\text{IC/BC}}, \tag{1.3}$$

where, L_{data} quantifies the loss from training data, including losses from the initial and boundary conditions of a PDE, while L_{PDE} accounts for the loss associated with adherence to the physical phenomena described by the PDE.

1.1.6 Neural Networks

Neural networks [16] form the backbone of contemporary machine learning and have shown remarkable success in a wide range of applications, from image recognition to natural language processing. At their core, neural networks are mathematical models that emulate the way the human brain processes information. They consist of layers of interconnected nodes or neurons, where each node represents a mathematical operation that transforms its input into an output. The specific arrangement of neurons within a network gives rise to different network types. The most common models are the Multi-Layer Perceptron (MLP) or Feed Forward Neural Networks (FFNN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN). A typical FFNN architecture is made of an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to several neurons in the next layer through pathways associated with weights, which are adjusted during the training process to minimize the error between the predicted outputs and the actual outputs. The transformation at each neuron is governed by a nonlinear activation function, which introduces the necessary nonlinearity into the network, enabling it to learn complex patterns,

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{w}\boldsymbol{x} + \boldsymbol{b}), \tag{1.4}$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{w}$ represents the weight matrix, $\boldsymbol{b}$ is a bias vector, and σ denotes the activation function. Common choices for σ include the sigmoid, tanh, and ReLU (Rectified Linear Unit) functions. The training of neural networks typically involves forward propagation of the input through the network to generate the output, followed by backpropagation of the error to adjust the weights. The aim is to minimize a loss function, which quantifies the difference between the network's prediction and the actual data. This process is iteratively repeated across multiple epochs or passes through the training dataset. The mathematical representation of the learning process in a neural network is typically formulated as the optimization problem

$$\min_{\boldsymbol{w},\boldsymbol{b}} \sum_{i=1}^{N} L(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{w}, \boldsymbol{b})), \tag{1.5}$$

where N is the number of training samples, $\boldsymbol{y}_i$ is the actual output, $f(\boldsymbol{x}_i; \boldsymbol{w}, \boldsymbol{b})$ is the predicted output by the network, and L is the loss function. The flexibility and power of neural networks come from their ability to approximate virtually any function, a property known as the universal approximation theorem. This theory says that given enough neurons and layers, a neural network can capture any underlying pattern in the data, making it a formidable tool in the machine learning arsenal.

1.2 Research Objectives

The main objective of this thesis is to develop a ML-based methodology using Physics-Informed Neural Networks to approximate solutions to the Navier-Stokes equations. The specific objectives are defined as follows:

1. To design and implement a PINN-based model for studying partial differential equations, with enhancements like domain decomposition and transformer networks;
2. To develop and validate a new PINN-based method applicable to high-dimensional problems governed by the Navier-Stokes equations;
3. To further extend the PINN-based approach to a wide range of complex problems, including nonlinear and high-dimensional 3D scenarios governed by the Navier-Stokes equations.

The methodologies to achieve these objectives are:

1. By incorporating the residuals of the Navier-Stokes equations along with initial and boundary conditions into the loss function, the approach transforms the problem into an optimization task. This framework utilizes gradient descent to refine the network's parameters for minimal loss, indicating precise solutions. Simultaneously, domain discretization and transformer networks enhance the PINN framework's flexibility and error mitigation. Complementary loss terms are added to maintain solution regularity across discrete models. Domain-specific transformers elevate the feature space for improved detection of data nuances. This model's efficacy will be assessed using the Burgers and Navier-Stokes equations.
2. Development and implementation of a novel methodology that combines temporal domain discretization, generative modeling in the Sobolev function space, and a gating mechanism to address high-dimensional challenges. This approach expands the application range beyond conventional test cases to more complex and realistic scenarios where traditional PINN models often fall short. The efficacy of this methodology will be demonstrated through its application to simulating time-dependent 3D turbulent channel flows governed by the incompressible Navier-Stokes equations.
3. Expansion of the search space of standard neural network models by integrating novel and improved network blocks, neuron-wise trainable sinusoidal activation mechanism,

designed to accurately approximate solutions to high-dimensional problems characterized by chaotic and oscillatory behaviors. The proposed model's performance will be assessed against realistic and challenging high-dimensional scenarios, testing the limits of conventional deep learning approaches.

1.3 Thesis Organization

This thesis is structured into six chapters.

Chapter 1: Introduction This chapter outlines the fundamentals, states the main and specific objectives of the research, and provides thesis organization.

Chapter 2: Literature Review This chapter provides a comprehensive background on existing techniques, focusing on Physics-Informed Neural Networks. It aims to present an overview, give preliminary references, discuss limitations of current PINN methods, and propose potential research directions.

Chapter 3: Improved PINN Model Using Domain Decomposition and Transformer Networks This chapter outlines the development and validation of both the standard PINN model and an advanced PINN model featuring domain decomposition and transformer networks. The efficacy of these models is showcased through their application to the Burgers equation as well as steady-state and time-dependent Navier-Stokes equations.

Chapter 4: Novel Domain Decomposition Technique for PINNs Chapter 4 introduces a domain decomposition technique for PINNs that decomposes the global domain into multiple overlapping subdomains. This allows for parallel solutions of subdomains and includes a generative modeling within the Sobolev function space. The chapter further explores a gating mechanism that manages interconnections between parallel models for optimal global training. The approach is validated using a complex real-world application involving turbulent channel flow governed by the 3D time-dependent Navier-Stokes equations.

Chapter 5: Modified Neural Network Architecture with Trainable Activation Mechanism This chapter proposes a modified neural network architecture that integrates a trainable activation mechanism and a frequency-based slope recovery mechanism within the PINN framework. The chapter explores the application of this novel mechanism to problems such as steady-state lid-driven cavity flow problem and time-dependent cylinder wake problem, demonstrating vortex shedding. Additional validation is conducted on a more complex scenario involving a 3D time-dependent turbulent channel flow problem, assessing the model's performance in large-scale phenomena and near-wall behavior, and testing its

extrapolative capabilities beyond the trained domain.

Chapter 6: General Discussion and Future Works The concluding chapter synthesizes findings from the three methodological chapters. It offers a comprehensive discussion on the implications of the research, and it suggests directions for future work to expand the scope of this doctoral research.

CHAPTER 2 LITERATURE REVIEW

Deep learning has demonstrated promising results in nonlinear mapping through the deployment of data-driven methods. They are trained to minimize the difference between the approximated solutions and training data generated with a numerical solver [17–20]. Nonetheless, these ML-based models lack physical interpretability, an essential aspect in scientific fields where conformity to fundamental equations is critical for effective generalization. Therefore, it is essential to incorporate established knowledge, fundamental physics principles, and domain-specific expertise to appropriately constrain these models during training with limited and sparse data.

2.1 Physics-Informed Neural Networks

Traditional numerical methods such as the finite difference method (FDM), the finite-element method (FEM), and the finite volume method (FVM) have long been the standard for approximating the solutions of the Navier-Stokes equations. Both the FEM and the FVM fall under the broader category of numerical methods known as the method of weighted residuals (MWR), where solutions to PDEs are approximated through a combination of trial functions. Inspired by the recent advancements in the universal approximation capabilities of multilayer perceptrons (MLPs) [21,22], a deep learning-based adaptation of MWR was developed to tackle simpler PDEs. In this adapted method, an MLP, parameterized by its weights and biases, approximates the subspace of functions that likely contain the PDE solution. This approach replaces the traditional linear combination of trial functions with an MLP to estimate the PDE solutions [23–25]. Originally proposed by Dissanayake & Phan-Thien [23] in the 1990s, the MLP parameters are optimized by minimizing a loss function that includes the PDE's residuals at selected collocation points, using a gradient descent algorithm akin to standard MLP training practices. Since the loss function includes the PDE's residuals, the MLP's output derivatives respecting time or space must be computed at each loss function evaluation during training. Lagaris et al. [24] recommended using back-propagation to compute these derivatives accurately, using the chain rule sequentially through each layer, a technique allowing accurate derivative values with a minimal number of collocation points. Furthermore, Dissanayake & Phan-Thien [23] included the residuals of boundary conditions in the loss function to weakly constrain the network, ensuring the MLP approximately meets these conditions. Conversely, Lagaris et al. [24] enhanced model convergence by enforcing a strong constraint using a trial solution that combines an MLP with a function designed

to maintain the integrity of the boundary conditions. However, this approach is confined to rectangular domains and necessitates the use of specially designed functions. To address these restrictions, Lagaris et al. [25] expanded their methodology to accommodate more complex geometries. This proved too resource-intensive to gain traction. Both groups of researchers adopted different methods for aggregating PDE residuals at collocation points; while Lagaris et al. [24] summed the residuals directly, Dissanayake & Phan-Thien [23] calculated the integral of the residuals across the computational domain. Kumar & Yadav [26] provided an extensive review of MLP applications for solving differential equations from the 1990s through the 2000s. Raissi et al. [15] pioneered Physics-Informed Neural Networks and then further refined the model to tackle primarily fluid dynamics-related problems [27–29]. They have become popular due to the novel approach to solving both forward [30–32] and inverse problems [33–35] involving PDEs, utilizing neural networks. PINN models satisfy physical laws by incorporating a weakly enforced loss function that accounts for the residuals of physical equations and boundary conditions. In other words, physics-informed models are trained using an unsupervised approach to find approximations that minimize the residuals of the governing equations [36, 37]. This novel method not only lessens the reliance on training data, but also circumvents the traditional requirements for mesh generation and numerical discretization by leveraging automatic differentiation [38]. However, despite improved performance, PINNs still encounter difficulties with high-dimensional problems, frequently experiencing issues with convergence and prolonged training time.

2.1.1 Domain Decomposition

Building on the foundational data clustering work by Mao et al. [39], Jagtap et al. [40] introduced domain decomposition techniques under the umbrella of conservative Physics-Informed Neural Networks (CPINN), enabling the use of independent models across various subdomains, thereby facilitating PINN parallelization. These methodologies were further elaborated to involve temporal decomposition [41], setting the stage for the development of parallel PINN strategies [42]. Subsequently, a time-sweeping and information propagation strategy was integrated to optimize training across these subdomains [43], enhancing computational efficiency. Concurrently, Kharazmi et al. [44, 45] advanced the concept of domain discretization within PINN by incorporating a variational approach, which significantly improved accuracy and reduced training overhead. This led to the introduction of non-overlapping domain models using high-order polynomial projections, where the trial space is represented by a global neural network space, and the test space contains piecewise polynomials. In related developments, Lorin et al. [46] showed the PINN framework's effectiveness in solving the time-dependent Dirac equation, illustrating its utility in quantum relativis-

tic physics without the need for derivative discretization. They further introduced a quasi-optimal Schwarz Waveform Relaxation (SWR) domain decomposition technique enhanced by PINNs to address the time-dependent Schrödinger equation, achieving rapid convergence and reduced computational load in comparison to conventional SWR methods [47]. Additionally, Zhang et al. [48] developed Trans-Net, a framework that use temporal domain decomposition to approximate partial differential equations more efficiently, using pre-trained networks from prior subdomains to expedite subsequent computations. Ren et al. [49] proposed SeismicNet, a physics-informed model tailored for seismic wave modeling in semi-infinite domains, which uses absorbing boundary conditions and temporal domain decomposition to enhance both scalability and accuracy, offering a superior alternative to traditional techniques without the necessity for labeled data.

2.1.2 Improvement of Training Efficiency

To enhance training efficiency, the standard PINN can be adapted for higher-dimensional applications by altering the neural network's architecture. Wang et al. [50] introduced a feedforward neural network (FFNN) coupled with two transformer networks [51–53] that elevate input variables into a high-dimensional feature space. Concurrently, Sirignano & Spiliopoulos [54] developed a Deep Galerkin Method (DGM), inspired from LSTM models [55]. Both approaches are designed to capture complex local variations in solutions using more sophisticated neural network architectures. Chiu et al. [56] used numerical discretization with automatic differentiation, applying this integrated technique to complex fluid dynamics problems such as flow mixing, lid-driven cavity flows, and channel flows over a backward-facing step. This method significantly reduces the need for numerous collocation points, and improves convergence rates. Psaros et al. [57] implemented a meta-learning framework in the PINN context to enhance performances on tasks beyond the usual training distribution, and to optimize the use of computational resources. In a related study, Penwarden et al. [58] delved into the core principles of model-agnostic meta-learning and transfer learning, adapting these concepts specifically for PINN applications, which have been assessed across various scenarios. Further innovations include the introduction of Geometry-Aware PINN (GA-PINN) by Oldenburg et al. [59], which integrates a variational auto-encoder with a boundary-constraining network to handle complex, non-parameterizable geometries. Another novel approach, the Spline-PINN by Wandel et al. [60], combines PINNs with convolutional neural networks (CNN) using Hermite spline kernels. This strategy is tailored to enable PINNs to infer fast and continuous solutions adaptable to new domains. Although this approach requires extended training phases across various parametric variables, it facilitates swift inference on different parametric settings. Additionally, Mclenny et al. [61] proposed

the Self-Adaptive PINN (SA-PINN), which enhances training dynamics through an analysis using the neural tangent kernel (NTK). This technique adjusts network weights in response to loss increases, effectively training the model to minimize losses while optimizing weight values.

2.1.3 Addressing Numerical Stiffness

Numerical stiffness poses a significant challenge for PINNs, resulting in imbalanced gradient flow during the back-propagation phase of training and consequently leading to competing loss terms. Despite the lack of definitive guidelines for optimal weight selection [62], various approaches have been investigated, as highlighted in several studies [63–66], which primarily focus on implementing regularization techniques, penalization strategies, or adapting learning rates to balance the coefficients of loss terms. A common strategy is to scale each loss term by a parameter λ to equitably distribute each term's influence on the total loss [67]. However, manual adjustments of these coefficients, while common, are often time-consuming and may lead to instability due to minor variations in these coefficients. To address this, several approaches have been developed. Self-adaptive methods in PINNs [61], and the self-adaptive weight PINN [68] using adaptive coefficients for each loss term to automate the balancing process. Additionally, techniques like learning rate annealing [50], and the neural tangent kernel [69] have been introduced to dynamically adjust the contribution of each term to the overall loss. Wang et al. [50] proposed a dynamic adjustment of learning rates, inspired by the adaptive moment estimation method introduced by Kingma et al. [70], and the attention mechanism by Vaswani et al. [51]. This approach uses a moving average to compute coefficients, taking into account the relative magnitudes of different loss terms. Expanding on this adaptive strategy for addressing time-dependent problems, the concept of causal PINNs [71] was developed to better manage dynamic systems exhibiting complex behaviors such as turbulence. In this framework, weighting coefficients are strategically applied to loss terms to prioritize the minimization of specific losses at certain times, contingent upon the successful minimization of preceding losses. Song et al. [72] introduced the LA-PINN model, which features a novel loss-attention mechanism. This model employs different attention-driven networks for each type of loss, dynamically adjusting weights at individual training points throughout the training process. This targeted adjustment significantly enhances the model's ability to handle areas of numerical stiffness.

2.1.4 Turbulence Modeling and Turbulent Problems

Using PINNs to approximate solutions of turbulent problems presents significant challenges that are not typically encountered in laminar flows. Situations involving high Reynolds numbers or near-wall regions often display nonlinear behaviors and sharp gradients, for which standard PINN models frequently struggle to capture accurately. These complexities necessitate the development of sophisticated and customized models. PINN models have been applied to turbulence closure modeling [73]. Ling et al. [74] deployed an MLP alongside a specialized neural network to address Reynolds-averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES) turbulence models. This specialized network incorporates Galilean invariance [75] through an advanced multiplicative layer. It was observed that the specialized network was capable of predicting the anisotropy tensor based on an invariant tensor basis, achieving enhanced accuracy over the MLP. Similarly, Maulik et al. [76] introduced a closure framework for subgrid modeling of Kraichnan turbulence [77]. This framework utilizes an implicit mapping approach, taking inputs such as grid-resolved variables and eddy viscosities to determine the dynamic closure strength. Eivazi et al. [78] enhanced the PINN framework by incorporating RANS equations into the loss function, enabling the model to address turbulence with Reynolds stress terms derived directly from the neural network's output. This model was validated against various problems, including the Zero Pressure Gradient (ZPG), the Adversarial Pressure Gradient (APG), the NACA4412 airfoil, and periodic hills. Xu et al. [79] tackled the issue of missing flow dynamics in turbulent flows by integrating parameterized Navier-Stokes equations as constraints, and showed that missing data in randomly specified regions could be inferred using an eddy viscosity-enhanced RANS model. This methodology was also adopted in subsequent studies to approximate turbulent flow dynamics [80, 81]. Patel et al. [82] used the Spalart-Allmaras (SA) turbulence model, augmented with a PINN approach, for data assimilation in turbulent flow scenarios. This hybrid method significantly improved mean-flow reconstruction accuracy over traditional RANS solvers. Wang et al. [83] used a hybrid turbulence modeling approach that combines the RANS and LES models with deep learning, introducing trainable spectral filters and using a specialized U-net architecture for turbulence prediction. This approach not only reduced prediction errors but also ensured that predicted physical fields respected conservation laws, such as mass conservation, while accurately capturing turbulent kinetic energy fields. Sliwinski et al. [84] developed a PINN methodology for the reconstruction of mean velocity and Reynolds stress fields in turbulent flows, using sparse velocity data to infer unknown closure quantities like the curl of unsteady RANS forcing. This application enhanced the assimilation of Reynolds stresses, enriching the understanding of the pressure field within the studied flows. The results highlighted the ability of PINNs to interpolate flows accu-

rately from sparse measurements, underlining their potential to reduce both experimental and computational costs in fluid dynamics research. Lastly, Jin et al. [85] demonstrated the use of PINNs for directly simulating 3D time-dependent turbulent channel flows at high Reynolds numbers, without having to use a turbulence model. They focused on optimizing the weighting of the loss function to balance data fidelity and physics, achieving outcomes that aligned well with Direct Numerical Simulation (DNS) data, albeit requiring substantial training time. While turbulence filtering methods simplify computations by averaging flow characteristics, they lead PINN models to overlook transient and fluctuating components, and phenomena with inherently unsteady behaviors, such as sharp gradients and near-wall phenomena, thus compromising the accuracy of complex flow dynamics capturing.

2.1.5 Network Architecture and Activation Mechanism

It is important to emphasize the fundamental aspects of Deep Neural Network (DNN) models, with an important component being the activation mechanism. Without this function, a model would merely function as a linear map between input and output spaces, regardless of the complexity of its architecture. There is no universal activation function suitable for all neural networks; the choice typically depends on the specifics of the problem at hand. Various innovative approaches have been developed, such as the adaptive sigmoid activation function for multilayer networks by Yu et al. [86], a data-driven method by Qian et al. [87] for blending activation functions in convolutional networks, and a technique by Dushkoff et al. [88] that allows neurons to select from multiple activation functions. Additionally, Abbasi et al. [89] introduced activation functions derived from physical principles relevant to the problem being modeled, diverging from conventional functions like tanh or sigmoid. By incorporating these into PINN models, the networks are more tightly governed by the underlying physics, enhancing both predictive accuracy and efficiency. Jagtap et al. [90] used adaptive activation functions to improve the approximating of solutions, both smooth and non smooth, of partial differential equations such as the nonlinear Klein-Gordon, Burgers, and Helmholtz equations. These adaptive functions, which include a scalable hyper-parameter, dynamically modify the topology of the loss function during optimization. Still using this approach, they introduced scalable parameters at both the layer and the neuron levels, optimizing them using stochastic gradient descent methods to boost training speeds and avert suboptimal convergence [91]. Their methodologies have showed faster training processes and enhanced approximation accuracy in both forward and inverse problem scenarios.

2.2 Research Direction and Proposed Approaches

Although DL-based approaches, specifically PINN-based methodologies, have seen rapid growth and notable improvements for fluid flow simulations, several persistent limitations remain. These include long training times [92], limited accuracy in high-dimensional and turbulent problems [85, 93, 94], imbalanced gradient flows, competing loss terms [50, 62], and limited adaptability in multi-scale problems [95, 96]. These challenges have hindered the application of PINNs to study high-dimensional and realistic scenarios, making this area less explored. In this thesis, we introduce novel PINN-based methods that incorporate spatial and temporal domain discretizations, generative modeling and transfer learning approaches, transformer networks, and trainable activation mechanisms. These enhancements are specifically tailored to enhance the approximation of the Navier-Stokes equations. To verify and validate proposed models, various problems will be studied. These include the Burgers equation, steady-state lid-driven cavity flows, and time-dependent cylinder wakes exhibiting vortex shedding. Additionally, to broaden the applicability of the PINN framework, a 3D time-dependent turbulent channel flow problem will be studied. This scenario represents a less explored domain within the machine learning framework, providing a challenging test of the enhanced models' capabilities in handling complex fluid dynamics.

CHAPTER 3 ARTICLE 1: A NOVEL DISCRETIZED PHYSICS-INFORMED NEURAL NETWORK MODEL APPLIED TO THE NAVIER-STOKES EQUATIONS

Authors: Amirhossein Khademi and Steven Dufour

Journal of Physica Scripta, IOP, submitted on 2024-03-28, accepted on 2024-06-07, published on 2024-06-20

3.1 Introduction

In recent years, several scientific ML-based techniques have been proposed to approximate the solution of nonlinear PDE, thanks to advancements in computing power and optimized methodologies. A notable development is the introduction of automatic differentiation [38], which makes it possible to bypass numerical discretization and mesh generation, leading to a new machine learning-based technique known as PINN [15, 27]. Since the introduction of the PINN, it has been widely used to approximate solution to PDE [28, 50, 85, 97–106], particularly highly nonlinear ones with non-convex and oscillating behavior, such as Navier-Stokes equations, that pose challenges for traditional numerical discretization techniques. The problematic part of Navier-Stokes equations is because of the convective term in the equations introduces a nonlinearity due to the product of velocity components. PINN reduce the actual reliance on labeled data, and they incorporate relevant physics into the ML-based model, addressing the issue of non-physics-informed neural networks that lack awareness of the underlying physics. As a result, the discrete problem is transformed into a hybrid system of supervised and unsupervised learning, where the minimization of the loss function incorporates both physics-based and data-dependent constraints.

While PINN improved significantly, solving high-dimensional non-convex optimization problems can result in local minima, and in long training times, which are major limitations. Significant efforts have been made to address these limitations. Clustering training data [39] has led to the development of domain decomposition techniques that can be used with PINN, where local neural networks, each with specific hyperparameters, are used in the various subdomains. This approach provides greater flexibility in handling problems with various physical characteristics, such as regions where solutions have large gradients. Li et al. [107] further extended this concept by incorporating variational techniques and overlapping domain decompositions in PINN. On the other hand, variational PINN [44] and later *hp*-refinement strategies were developed using non-overlapping domain decompositions and projections onto

a space of high-order polynomials [45]. The conservative PINN (CPINN) [40] was then introduced, which is a domain decomposition-based PINN for conservation laws that enforces the continuity of states and fluxes across subdomain interfaces. This concept was further extended to general PDE using the extended PINN approach, which offers domain decompositions both in space and time [41]. Subsequently, a parallel PINN [42] was proposed to complement the scenario previously established by the CPINN. Aulakh et al. [108] introduced generalized finite-volume-based discretized loss functions integrated into pressure-linked algorithms for unsupervised training of neural networks (NN) to approximate Navier-Stokes equation solutions. They combined physics-informed loss functions with data-driven ones to train NN with partial or sparse observations. Additionally, they modified the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithm to support physics-based NN training by adding feedback loops at various solution steps. Finally, a novel approach was proposed in which residual-based adaptive sampling was utilized, effectively targeting regions with elevated residuals [109]. Robust agreement with both analytical solutions and validated computational fluid dynamics (CFD) calculations was demonstrated through the findings of this simulation. relevantly, given the importance of optimizing the distribution of training points, Qin et al. [110] introduced an adaptive mixing sampling approach within PINN, dynamically enhancing the distribution of training points and resulting in superior accuracy compared to conventional PINN methods.

In another line of research, Wang et al. [50] proposed an improved fully connected neural architecture by utilizing transformer networks (TRF) in combination with a standard fully connected neural network (FC-NN) to enhance the hidden states with residual connections. This idea came from neural network systems used for computer vision and natural language processing [51]. Wang explored the application of this improved architecture for studying fluid flow problems [111].

Even if various domain decomposition models were proposed to be used with PINN, building a domain decomposition model with specialized loss functions, and the study of the contribution of loss terms, have been minimally explored. With current domain-decomposed models, loss terms are similar to those used in a baseline PINN, and the addition of current extra loss terms may not adequately address the continuity and the regularity of the solution of the PDE. When approximating the solution to the Navier-Stokes equations, there is a significant “magnitude difference" in the predicted pressure field, despite the fact that the velocity and the pressure fields are accurately discretized.

In this study, we develop a novel improvement technique within the framework of the domain decomposition method for PINN by building complementary loss functions, using domain-specific TRF, and performing optimization within each subdomain independently. Our con-

tributions include:

- **New Physics-Informed Module:** Introduction of a novel physics-informed module designed to maintain solutions' continuity, differentiability, and promote regularity and smoothness by enforcing the H^1 norm of the MSE in interface loss terms. In this study, "solution regularity" encompasses the continuity, differentiability, and smoothness of the global solution.
- **Comprehensive Evaluation and Comparative Analysis of Loss Terms:** This study pioneers an in-depth investigation and comparative assessment of different loss terms, considering both computational efficiency and accuracy, marking the first instance of such analysis.
- **Subdomain-Specific Transformer Networks:** Utilization of domain-specific TRF to elevate the model's capacity by mapping input variables into a higher-dimensional feature space, facilitating enhanced capture of subtle data variations.
- **Enhanced Model's Capacity and Flexibility:** Incorporation of TRF within each subdomain amplifies the model's ability to discern complex patterns and relationships, leading to improved accuracy and convergence rates. Operating in a higher-dimensional space provides greater flexibility.
- **Convergence Analysis:** Following the application of TRF, we have performed a convergence analysis to study the impact of sub-network TRF on the overall convergence behavior of the model.
- **Error Analysis:** Error analysis conducted to investigate the mode of failure related to the weak interface constraints within the domain decomposition approach in the PINN framework.
- **Dominant Subdomain Approach:** An innovative approach is introduced to conduct a priori error analysis, employing a subdomain-wise strategy and implementing selective interface constraints to address the challenge of "pressure magnitude difference." This challenge has been a limitation in both baseline PINN approaches and basic domain decomposition PINN methods.

To verify the effectiveness of the proposed DD-PINN model, we have performed simulations and compared the results against the results of direct numerical simulation (DNS) of two test problems [27, 112]; the steady-state flow in a two-dimensional lid-driven cavity problem and

the time-dependent two-dimensional cylinder wake, both described by the two-dimensional (2D) incompressible Navier-Stokes equations. The results of numerical comparisons in both test cases demonstrate that the DD-PINN promotes global solutions regularity and smoothness, by preserving the continuity and differentiability of the solutions and improves approximation accuracy and the rate of convergence. The second test case notably underscores the significance of enforcing interface-specific continuity and differentiability constraints, leveraging the H^1 norm of MSE for these constraints. This refinement led to a substantial enhancement in the approximation of the pressure field compared to previous domain decomposition models like CPINN. To the best of our knowledge, this marks the first instance within the PINN framework where a method effectively addresses the challenge of accurately approximating the pressure field, which has been a limitation in both baseline PINN approaches and basic domain decomposition PINN methods.

3.2 Methodology

3.2.1 Overview

Neural Network

Based on the universal approximation theorem, it is stated that a multi-layer perceptron with a single hidden layer and a finite number of neurons has the capability to approximate any continuous function with an arbitrary precision [21, 113, 114]. From a mathematical standpoint, an Artificial Neural Network (ANN) can be characterized as a graph made of a collection of vertices that represent neurons, along with a collection of edges that represent the connections between them. A FC-NN belongs to the category of deep neural networks (DNN) that possess multiple hidden layers. This type of network is composed of neurons that carry out computations in a sequential manner across the various layers. Neurons in an FC-NN architecture are interconnected only with the neurons of the adjacent layer and not with those in the same layer. Every neuron in the network applies an activation function to the weighted sum of the inputs, plus a bias factor. So, a FC-NN architecture can be described mathematically as

$$f_i(\boldsymbol{x}_i; \boldsymbol{w}_i, \boldsymbol{b}_i) = \alpha_i(\boldsymbol{w}_i \cdot \boldsymbol{x}_i + \boldsymbol{b}_i), \tag{3.1}$$

where $\boldsymbol{x}$ is the input, $\boldsymbol{w}_i$ denotes the weight vector, $\boldsymbol{b}$ represents the bias vector and α is a non-linear activation function. This model can be seen as a composition of layers, using another notation [115], such that equation 3.1 can be also written as

$$\boldsymbol{u}_\theta(x) = C_K \circ \alpha \circ C_{K-1} ... \circ \alpha \circ C_1(x), \tag{3.2}$$

where, for any $k = 1, 2, ..., K$

$$C_k(x_k) = \boldsymbol{W}_k x_k + \boldsymbol{b}_k, \tag{3.3}$$

and where C_k and W_k are the transformation function and the weight matrix associated with the k_{th} hidden layer and $\circ$ denotes the composition operation.

PINN

PINN are a class of neural network-based algorithms designed for solving PDE in a data-driven manner. Traditional methods for solving PDE, such as finite difference or finite element methods, often require a grid-based discretization of the domain and are computationally expensive, especially for complex geometries or high-dimensional problems. PINN offer an alternative approach by leveraging the expressive power of neural networks to directly approximate the solution of the underlying PDE without the need for explicit discretization. At the core of PINN is the idea of enforcing the governing equations of the PDE as constraints on the neural network model. This is achieved by incorporating both the available boundary conditions and the PDE itself into the loss function, which the neural network aims to minimize during training. By doing so, PINN combine the flexibility of neural networks with the physical principles encoded in the PDE, leading to accurate and efficient solutions. Given randomly generated sets of training and collocation points $\{\boldsymbol{x}_u^i\}_{i=1}^{N_u}$ and $\{\boldsymbol{x}_F^i\}_{i=1}^{N_F}$ respectively, the PINN algorithm looks for an optimal set of parameters $\boldsymbol{\theta}$ for which the loss function that used to compute the MSE between the discrete approximation and reference values is minimal. The loss function for baseline PINN is given by

$$L = MSE_b + MSE_F, \tag{3.4}$$

where the mean squared errors are given by:

$$MSE_b(\boldsymbol{\theta}, \{\boldsymbol{x}_u^i\}_{i=1}^{N_u}) = \frac{1}{N_u}\sum_{i=1}^{N_u} |u^i - u_\theta(\boldsymbol{x})_u^i|^2; \tag{3.5}$$

$$MSE_F(\boldsymbol{\theta}, \{\boldsymbol{x}_F^i\}_{i=1}^{N_F}) = \frac{1}{N_F}\sum_{i=1}^{N_F} |F_\theta(\boldsymbol{x})_F^i|^2. \tag{3.6}$$

u and u_θ represent the boundary data and the neural network approximations of boundary data at corresponding training points and F_θ signifies the neural network model for approximating the governing equations. MSE_b and MSE_F correspond to the data mismatch due to boundary conditions and the magnitude of the PDE residuals evaluated at collocation points. In this setting, the PDE approximation problem is transformed into a minimization problem

to find network trainable parameters $\boldsymbol{\theta} = \{\theta_1, ..., \theta_N\}$, where N is the number of layers of the NN model.

Baseline Decomposed-Domain Physics-Informed Neural Network

Recent works focusing on domain decomposition in PINN have introduced additional terms to the loss function, expanding the conventional formulation to account for domain-specific characteristics,

$$L = MSE_b + MSE_F + MSE_{\bar{\boldsymbol{u}}} + MSE_R. \tag{3.7}$$

The MSEs are defined as:

$$MSE_b(\boldsymbol{\theta}_\Omega, \{\boldsymbol{x}_u^i\}_{i=1}^{N_u}) = \frac{1}{N_{u_\Omega}} \sum_{i=1}^{N_{u_\Omega}} \left| u^i - u_{\theta_\Omega}(\boldsymbol{x})_{u_\Omega}^i \right|^2; \tag{3.8}$$

$$MSE_F(\boldsymbol{\theta}_\Omega, \{\boldsymbol{x}_F^i\}_{i=1}^{N_F}) = \frac{1}{N_{F_\Omega}} \sum_{i=1}^{N_{F_\Omega}} \left| F_{\theta_\Omega}(\boldsymbol{x})_{F_\Omega}^i \right|^2; \tag{3.9}$$

$$MSE_{\bar{\boldsymbol{u}}}(\boldsymbol{\theta}_\Omega, \{\boldsymbol{x}_I^i\}_{i=1}^{N_I}) = \sum_{\forall \Omega^+} \left(\frac{1}{N_{I_\Omega}} \sum_{i=1}^{N_{I_\Omega}} \left| u_{\theta_\Omega}(\boldsymbol{x})_{I_\Omega}^i - \left\{ \{ u_{\theta_\Omega}(\boldsymbol{x})_{I_\Omega}^i \} \right\} \right|^2 \right); \tag{3.10}$$

$$MSE_R(\boldsymbol{\theta}_\Omega, \{\boldsymbol{x}_I^i\}_{i=1}^{N_I}) = \sum_{\forall \Omega^+} \left(\frac{1}{N_{I_\Omega}} \sum_{i=1}^{N_{I_\Omega}} \left| F_{\theta_\Omega}(\boldsymbol{x}_{I_\Omega}^i) - F_{\theta_{\Omega^+}}(\boldsymbol{x}_{I_\Omega}^i) \right|^2 \right). \tag{3.11}$$

The subdomain for which the loss function is defined is represented by Ω, while Ω^+ denotes an adjacent subdomain. With the decomposition of the domain, the loss function is modified to incorporate two additional terms: $MSE_{\bar{\boldsymbol{u}}}$ and MSE_R. These terms account for flux and solution continuity at the interfaces [41]. In $MSE_{\bar{\boldsymbol{u}}}$, the difference between the subdomain's solution and the average solution (average on two adjacent subdomains) is minimized at the interface. This average term is simply computed as the average of the combination of the solutions on two adjacent subdomains at the interface points. In MSE_R, the difference between the residuals of the PDE at the interface between two adjacent subdomains is minimized.

3.2.2 Discretized Domain Physics-Informed Neural Networks

Domain Discretization

In the domain decomposition technique, the domain is subdivided into several non-overlapping subdomains, as depicted in Figure 3.1. The entire domain can then be defined as

$$\Omega = \cup_{i=1}^{N_s} \Omega_i \ , \tag{3.12}$$

where

$$\Omega_i \cap \Omega_j = \partial\Omega_{ij} \ , \tag{3.13}$$

where N_s represents the total number of subdomains. This enables the utilization of networks with various architectures and hyperparameters in each subdomain. In this setting, if the approximation of the solution in the i^{th} subdomain is written as

$$\boldsymbol{u}_{\theta_i} = N^L(\boldsymbol{x}, \boldsymbol{\theta}_i), \tag{3.14}$$

where N^L is the neural network model with L layers, then the global solution is given by

$$\boldsymbol{u}_\theta = \cup_{i=1}^{N_s} \boldsymbol{u}_{\theta_i}. \tag{3.15}$$

The parameters of the network are updated independently in each subdomain and in parallel, using the gradient descent method. However, networks communicate with each other at each iteration at the interface of the subdomains, using the continuity and differentiability constraints.

In the proposed DD-PINN methodology, the enforcement of effective interface constraints between subdomains relies on subdomain-specific loss terms. These terms play a crucial role in ensuring continuity and differentiability at the interfaces between subdomains. Each subdomain is associated with its own neural network model, represented by parameters $\boldsymbol{\theta}i$. The solution within each subdomain is approximated by the neural network model specific to that subdomain, denoted as $\boldsymbol{u}\theta_i$. Therefore, the global solution $\boldsymbol{u}_\theta$, which encompasses the entire domain Ω, is obtained by aggregating the solutions from individual subdomains, as shown in Equation (3.15). To enforce interface constraints effectively, the loss function for each subdomain includes terms that penalize deviations from these constraints. Specifically, the loss function incorporates information from neighboring subdomains to ensure continuity and differentiability across interfaces. This is achieved by introducing additional terms in the loss function that quantify the discrepancy between the solutions and their derivatives

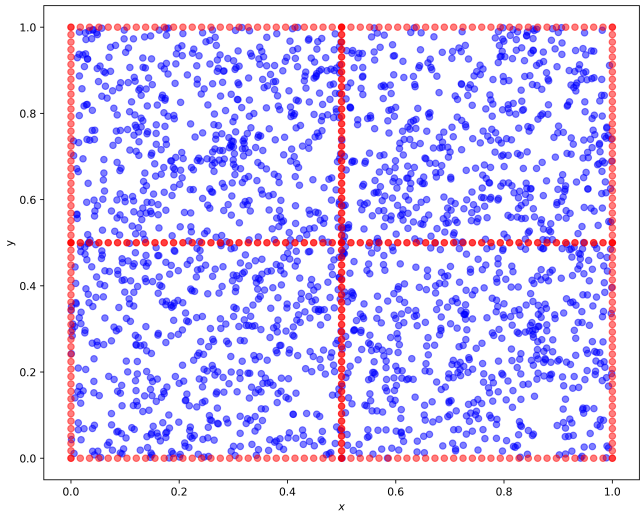


Figure 3.1 Decomposed domain; collocation points are in blue and the evaluation points of boundary conditions and interface constraints are in red.

at the interfaces.

For continuity enforcement, the loss function includes terms that penalize differences between the solutions at interface points shared by neighboring subdomains. These terms encourage the solutions to smoothly transition between subdomains, minimizing abrupt changes at the interfaces. Similarly, to ensure differentiability at the interfaces, additional terms are introduced in the loss function to penalize differences in the derivatives of the solutions. By minimizing these differences, the neural network models learn to produce solutions that exhibit smooth transitions in derivatives across subdomain boundaries.

During training, the parameters of the neural network models are updated independently for each subdomain using gradient descent methods. However, communication between subdomains occurs at each iteration, where the interface constraints are enforced through the subdomain-specific loss terms.

Problem Setup

To establish the methodology, a problem involving steady-state flow in a two-dimensional lid-driven cavity is considered. The incompressible Navier-Stokes equations govern this flow

which can be written in non-dimensional form as

$$\boldsymbol{u}\cdot\nabla\boldsymbol{u}+\nabla p-\frac{1}{Re}\Delta\boldsymbol{u}=0 \qquad \text{in } \Omega; \tag{3.16}$$

$$\nabla\cdot\boldsymbol{u}=0 \qquad \text{in } \Omega; \tag{3.17}$$

$$\boldsymbol{u}(x,y)=(1,0) \qquad \text{on } \Gamma_1; \tag{3.18}$$

$$\boldsymbol{u}(x,y)=(0,0) \qquad \text{on } \Gamma_0, \tag{3.19}$$

where $\boldsymbol{u}(x,y)=(u(x,y),v(x,y))$ represents a vector of velocity field, $p(x,y)$ denotes a pressure field and the spatial coordinates (x,y) are constrained within the domain $\Omega=[0,1]\times[0,1]$, where Ω corresponds to a two-dimensional square cavity. The top boundary of this cavity is denoted by Γ_1, while the remaining three sides of the boundary are represented by Γ_0. Additionally, Re stands for the Reynolds number characterizing the flow, which is the ratio of inertial forces to viscous forces in the flow, defined as $Re=\rho\ u\ L/\mu$, where ρ is the density of the fluid, u is the characteristic velocity, L is the characteristic length scale and μ is the dynamic viscosity of the fluid.

By adopting the stream-function formulation of the two-dimensional incompressible Navier-Stokes equations, the neural network model takes a set of (x,y) points as input and produces the associated scalar stream function $\psi(x,y)$ along with the scalar pressure field $p(x,y)$. In other words, the solutions to the Navier-Stokes equations are explored in a set of divergence-free functions,

$$u_x+v_y=0. \tag{3.20}$$

Following the stream-function formulation mentioned above, introducing a scalar stream function facilitates the direct computation of an incompressible velocity field

$$u=\partial\psi/\partial y,\ v=-\partial\psi/\partial x, \tag{3.21}$$

for the latent scalar stream function $\psi(x,y)$. This assumption makes it possible to automatically satisfy the continuity constraint.

To build the loss function, the residuals of the PDE are first needed, as expressed by the following:

$$F_{\theta_x}=u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+\frac{\partial p}{\partial x}-\frac{1}{Re}(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}); \tag{3.22}$$

$$F_{\theta_y}=u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+\frac{\partial p}{\partial y}-\frac{1}{Re}(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}), \tag{3.23}$$

where F_{θ_x} and F_{θ_y} represent the residuals of the momentum equation (3.16), in the x and y directions, respectively.

Loss Function of the DD-PINN

With the DD-PINN, the continuity and the differentiability conditions are applied along the interfaces. The loss function is then formulated as

$$\begin{aligned} L = &MSE_b + MSE_F + MSE_{\psi_{\text{int}}} + MSE_{p_{\text{int}}} + MSE_{\nabla(\psi_{\text{int}})} \\ &+ MSE_{\nabla(p_{\text{int}})}, \end{aligned} \tag{3.24}$$

where the subscript "int" is the index of the loss terms corresponding to each interface constraint. In this formulation, MSE_b and MSE_F correspond to the loss associated with the boundary conditions and the residuals of the PDE, $MSE_{\psi_{\text{int}}}$ and $MSE_{p_{\text{int}}}$ represents the loss associated with the continuity of the NN outputs at the interface. To be more detailed, sets of evaluation points are defined on any interface of subdomains, where we enforce the two subdomains to have equal approximations. As a result, the continuity of the global solutions (3.15) is preserved. Finally $MSE_{\nabla(\psi_{\text{int}})}$ and $MSE_{\nabla(p_{\text{int}})}$ correspond to the loss associated with the gradients of the NN outputs at the interface, respectively. By taking the gradient of the output of the NN into account, we enforce the H^1 norm of the MSE in the interface loss terms, rather than L^2 norm, which promotes the regularity of the solution. The loss function in the Ω^{th} subdomain, illustrated in Figure 3.2, is written as

$$MSE_b(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x},\boldsymbol{y})_u^i\}_{i=1}^{N_b}) = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|(u,v)^i - (u,v)_{\theta_\Omega}^i\right|_2^2; \tag{3.25}$$

$$MSE_F(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x},\boldsymbol{y})_F^i\}_{i=1}^{N_F}) = \frac{1}{N_F}\sum_{i=1}^{N_F}\left|F_{\theta_\Omega}(x,y)_F^i\right|_2^2; \tag{3.26}$$

$$\begin{aligned} &MSE_\psi(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x},\boldsymbol{y})_{\text{int}}^i\}_{i=1}^{N_{\text{int}}}) \\ &= \sum_{\forall\Omega^+}\left(\frac{1}{N_{\text{int}}}\sum_{i=1}^{N_{\text{int}}}\left|(\psi)_{\theta_\Omega}(x,y)_{\text{int}}^i - (\psi)_{\theta_{\Omega^+}}(x,y)_{\text{int}}^i\right|_2^2\right); \end{aligned} \tag{3.27}$$

$$\begin{aligned} &MSE_p(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x},\boldsymbol{y})_{\text{int}}^i\}_{i=1}^{N_{\text{int}}}) \\ &= \sum_{\forall\Omega^+}\left(\frac{1}{N_{\text{int}}}\sum_{i=1}^{N_{\text{int}}}\left|(p)_{\theta_\Omega}(x,y)_{\text{int}}^i - (p)_{\theta_{\Omega^+}}(x,y)_{\text{int}}^i\right|_2^2\right); \end{aligned} \tag{3.28}$$

$$
\begin{aligned}
&MSE_{\nabla(\psi)}(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x},\boldsymbol{y})^i_{\text{int}}\}_{i=1}^{N_{\text{int}}}) \\
&= \sum_{\forall\Omega^+} \left(\frac{1}{N_{\text{int}}} \sum_{i=1}^{N_{\text{int}}} \left| (\nabla(\psi))_{\theta_\Omega}(x,y)^i_{\text{int}} - (\nabla(\psi))_{\theta_{\Omega^+}}(x,y)^i_{\text{int}} \right|_2^2 \right);
\end{aligned} \tag{3.29}
$$

$$
\begin{aligned}
&MSE_{\nabla(p)}(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x},\boldsymbol{y})^i_{\text{int}}\}_{i=1}^{N_{\text{int}}}) \\
&= \sum_{\forall\Omega^+} \left(\frac{1}{N_{\text{int}}} \sum_{i=1}^{N_{\text{int}}} \left| (\nabla(p))_{\theta_\Omega}(x,y)^i_{\text{int}} - (\nabla(p))_{\theta_{\Omega^+}}(x,y)^i_{\text{int}} \right|_2^2 \right),
\end{aligned} \tag{3.30}
$$

where N_b, N_F, and N_{int} are the number of boundary points on Ω_i, the number of collocation points in Ω, and the number of interface points on the common interface with adjacent subdomains, respectively. u and v denote training data, in particular velocity, in x and y directions. Terms with the subscript Ω^+ correspond to any adjacent subdomain. Enforcing the continuity of the derivatives of the discrete solution gives C^1 continuity of the solutions across the interface. This means that not only do the degree of freedom (DOF) of the discrete solution match at the interface, but the first derivative of the solution is also continuous across the interface. Enforcing the continuity of the derivatives of the discrete solution across the interface can be useful in cases where the problem requires more smoothness across the interface. For example, in problems where the solution needs to be continuously differentiable, C^1 continuity can help improve the accuracy and the convergence rate of the numerical solution [40].

Analysis of number and size of subdomains

In our methodology, the number and size of subdomains are hyperparameters. Similar to the process of optimizing the architecture of neural networks, determining the ideal number and size of subdomains involves empirical exploration. These hyperparameters are influenced by various factors such as the number of interface points, training points, collocation points, choice of activation functions, subdomain geometry, function space constraints, and the architecture of the main neural network. As a result, selecting the appropriate subdomain size requires iterative experimentation. Adjusting the size of subdomains can impact training complexity and computational efficiency. Smaller subdomains simplify training by limiting the range of gradient changes, but require more points for assessing solution regularity, potentially increasing computational time. Also, using too many subdomains may underutilize the neural network's capabilities. Hence, determining the optimal size and number of subdomains is problem-specific and relies on prior understanding of the problem's complexity, coupled with iterative experimentation.

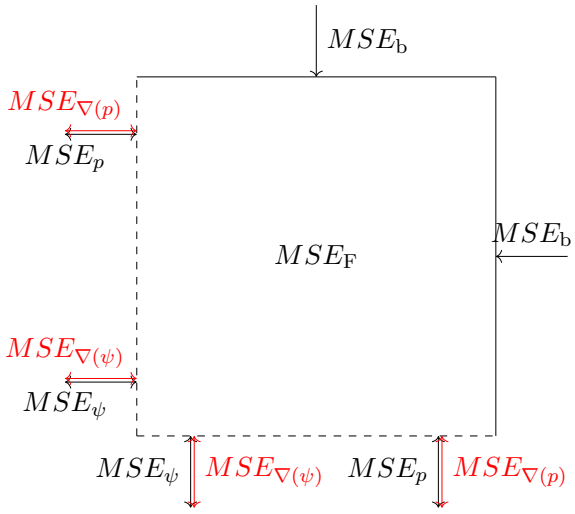


Figure 3.2 A sample subdomain Ω_i. Dashed lines show interfaces with adjacent subdomains.

3.2.3 DD-PINN with Transformer Networks

TRF, in fact, are a pair of encoders that map input variables to a high-dimensional feature space with higher DOF and update hidden layers with point wise multiplication (refer to the forward pass propagation rule 3.35). Mapping inputs to a higher dimensional feature space can enhance the accuracy of approximation by allowing the model to capture more complex relationships within the data. This process facilitates the extraction of intricate patterns and structures that may be latent or nonlinear in nature. By operating in a higher dimensional space, the neural network gains increased representational capacity, enabling it to better discern and model the underlying characteristics of the data. Following the introduction of the improved FC-NN by [50], in which transformer networks were incorporated inside the FC-NN, we adapt the DD-PINN and propose a new technique with the inclusion of TRF in each subdomain separately. This advancement enhances the hidden state by incorporating residual connections. As illustrated in Figure 3.3, this can be considered as a standard MLP with the incorporation of two encoders, where the inputs $\boldsymbol{X}$ are embedded into a feature space using TRF U and V, and then merged back in the standard FC-NN using pointwise

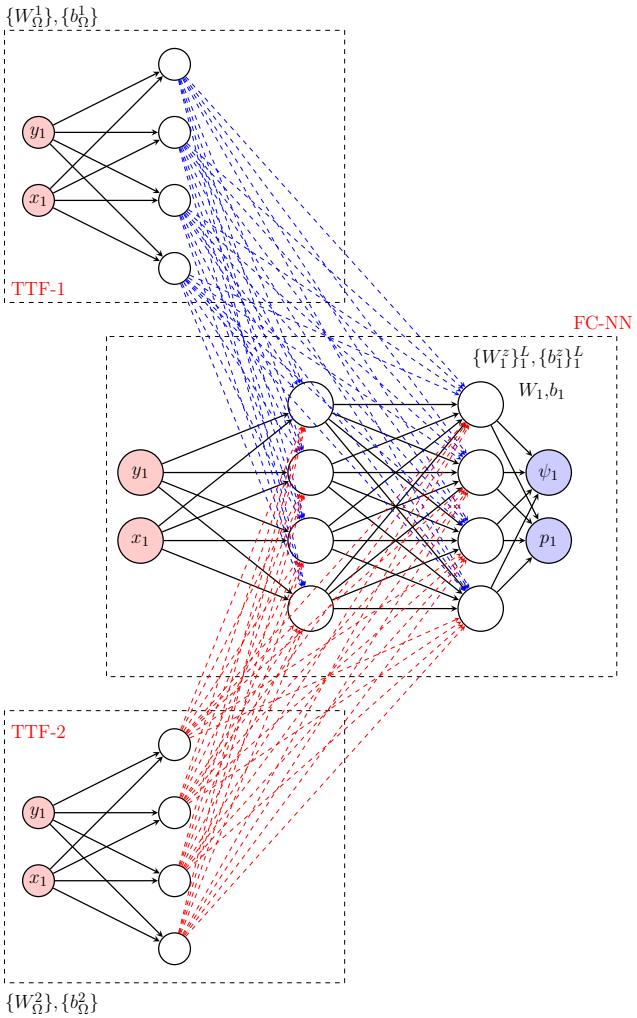


Figure 3.3 Schematic of the improved FC-NN with transformer networks at each subdomain

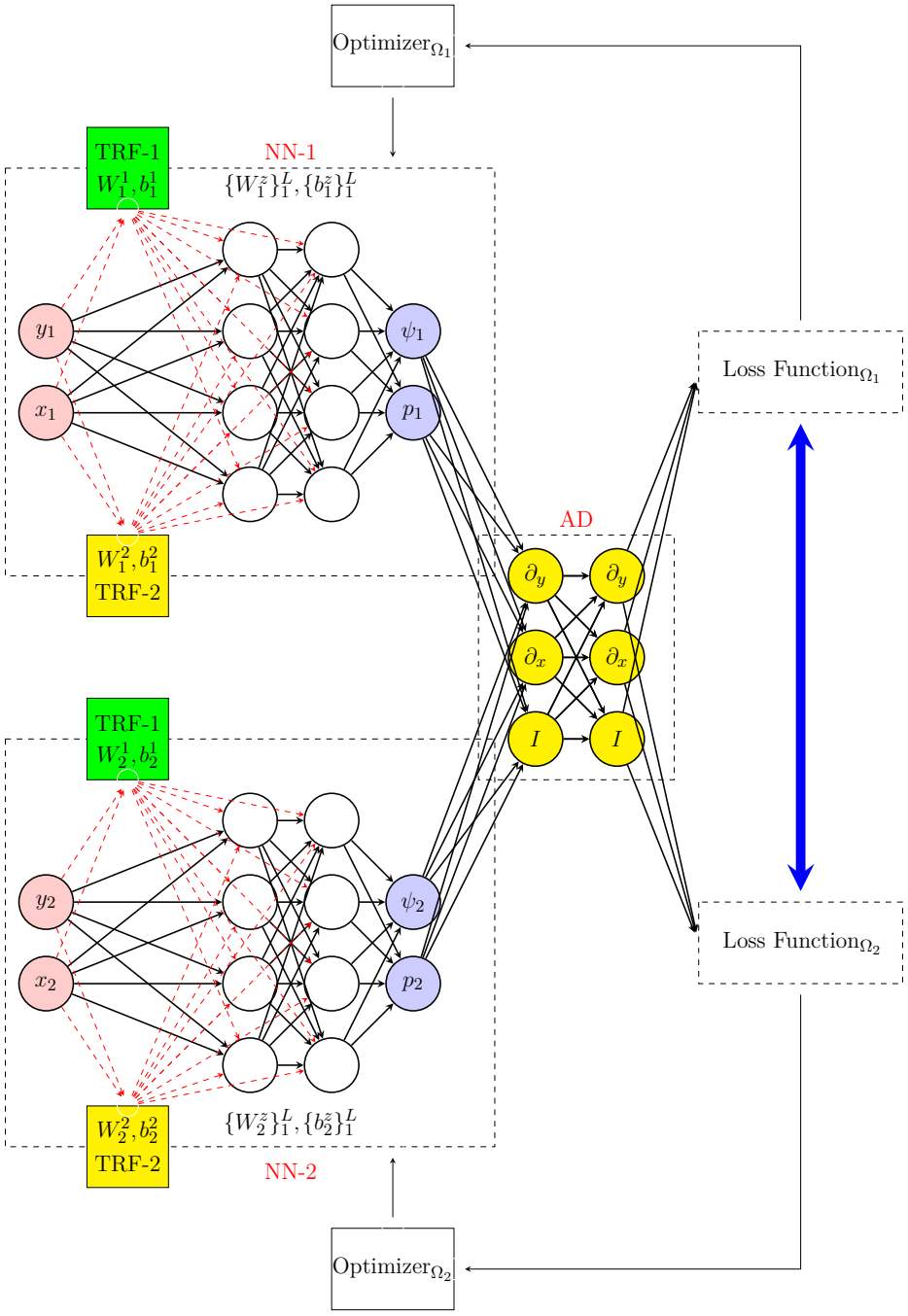


Figure 3.4 Schematic of the DD-PINN for a case with two subdomains. The blue arrow represents a link between subdomains through interface constraints.

multiplication. The forward propagation rule can be written as below:

$$\boldsymbol{U}_\Omega = \sigma(\boldsymbol{X}\boldsymbol{W}^1 + \boldsymbol{b}^1), \quad \boldsymbol{V}_\Omega = \sigma(\boldsymbol{X}\boldsymbol{W}^2 + \boldsymbol{b}^2), \tag{3.31}$$

$$\boldsymbol{H}_\Omega^1 = \sigma(\boldsymbol{X}\boldsymbol{W}^{z,1} + \boldsymbol{b}^{z,1}), \tag{3.32}$$

$$\boldsymbol{Z}_\Omega^k = \sigma(\boldsymbol{H}^k\boldsymbol{W}^{z,k} + \boldsymbol{b}^{z,k}), \quad k = 1, ..., L, \tag{3.33}$$

$$\boldsymbol{H}_\Omega^{k+1} = (1 - \boldsymbol{Z}^k) \odot \boldsymbol{U}_\Omega + \boldsymbol{Z}^k \odot \boldsymbol{V}_\Omega, \quad k = 1, ..., L, \tag{3.34}$$

$$\boldsymbol{f}_{\Omega_\theta}(\boldsymbol{x}) = \boldsymbol{H}_\Omega^{(L+1)}\boldsymbol{W} + \boldsymbol{b}, \tag{3.35}$$

where σ represents a nonlinear activation function, and $\odot$ denotes pointwise multiplication. The variable $\boldsymbol{H}_\Omega^1$ represents the output of the first hidden layer, while $\boldsymbol{Z}_\Omega^k$ denotes the gating mechanism in the transformer network at the k_{th} hidden layer. This mechanism determines the importance or contribution of information from the $\boldsymbol{U}_\Omega$ and $\boldsymbol{V}_\Omega$ pathways in each subdomain network. The variable $\boldsymbol{H}_\Omega^{k+1}$ represents the output of the $(k+1)^{th}$ hidden layer, and f_{Ω_θ} corresponds to the final output of the network. In this setting, the parameters $\boldsymbol{W}^1, \boldsymbol{b}^1, \boldsymbol{W}^2$ and $\boldsymbol{b}^2$ are nontrainable parameters of the TRF. On the other hand, the parameters $\boldsymbol{W}_\Omega^{z,l}$ and $\boldsymbol{b}_\Omega^{z,l}$ are the trainable parameters of the FC-NN networks. Lastly, W_Ω and b_Ω are the trainable parameters of the last layer of the FC-NN network. Leveraging the benefit of decomposed domains, separate domain specific neural networks are used in each subdomain. Thus, the DD-PINN model has several sets of trainable and nontrainable parameters as

$$\boldsymbol{\theta}_\Omega = \{\boldsymbol{W}_\Omega^1, \boldsymbol{b}_\Omega^1, \boldsymbol{W}_\Omega^2, \boldsymbol{b}_\Omega^2, (\boldsymbol{W}_\Omega^{z,l}, \boldsymbol{b}_\Omega^{z,l})_{l=1}^L, W_\Omega, b_\Omega\} \tag{3.36}$$

$$\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_{N_\text{s}}\}, \tag{3.37}$$

where N_s is the number of subdomains. A schematic of the DD-PINN illustrated through Figure 3.4.

Effectiveness and Limitations

The effectiveness of projecting input data to a higher-dimensional feature space, such as through the use of the TRF, depends on various factors related to the nature of the problem and the characteristics of the data. Here are some scenarios where projection to a higher feature space can help, and when it may not provide significant benefits:

When projection to a higher feature space helps:

1. **Complex and Nonlinear Relationships:** In cases where the underlying relation-

ships within the data are complex and nonlinear, projecting the data to a higher-dimensional space can help capture these intricate patterns more effectively. This is because higher-dimensional spaces provide more flexibility for modeling complex relationships.

2. **Intricate Data Structures:** When the data exhibits intricate structures or contains hidden patterns that are not easily discernible in lower-dimensional spaces, projecting it to a higher-dimensional feature space can help reveal and exploit these structures, leading to improved performance.

When projection to a higher feature space may not help as much:

- **Linearly Separable Data:** If the data is already linearly separable in its original feature space, projecting it to a higher-dimensional space may not provide significant benefits, as the additional dimensions may not contribute meaningfully to improving separation.

3.3 Results and Discussion

In this section, the results of numerical studies are presented to validate the proposed methodology of the DD-PINN. The main objective is to approximate the Navier-Stokes solutions in terms of velocity and pressure fields. The contributions of various loss terms in the DD-PINN model are studied and their effectiveness are compared. In the first problem, the performance of a DD-PINN with subdomain TRF is also verified. In all cases, the hyperbolic tangent function serves as the nonlinear activation function, the training employs gradient descent with the Adam optimizer [70], and NN initialization follows the Glorot method [116]. TensorFlow [117] is employed to build and execute all the algorithms. This is worth mentioning that to ensure a fair comparison, the training time and maximum available memory were kept the same for all methods. In this section, the results are presented as below:

- **The 2D Lid-Driven Cavity Problem**
  - **Loss Terms Contributions**: Analyzes the efficiency and error characteristics of loss terms in the basic decomposed-domain strategies in PINN and the DD-PINN. Compares the performance of DD-PINN with basic methods and examines failure modes of basic decomposition strategies in contrast to DD-PINN.
  - **Contribution of Sub-Network TRF**: Investigates the impact of the sub-network TRF within DD-PINN. It compares the performance of DD-PINN with TRF against

a counterpart DD-PINN model without TRF, focusing on error metrics and convergence rates.

- **Combined Contributions (DD-PINN with TRF)**: Examines the impact of integrating TRF into DD-PINN, specifically analyzing convergence rates compared to basic domain decomposition.

- **2D Cylinder Wake**

  - **Dominant Subdomain Approach**: Introduces a novel technique for a priori error analysis, employing a subdomain-wise approach and enforcing selective interface constraints to enhance the accuracy of pressure field approximations.

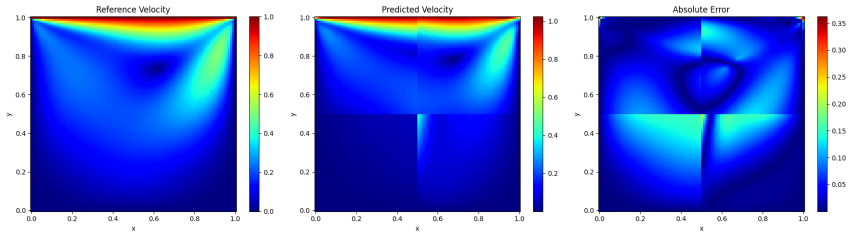
3.3.1 The 2D Lid-Driven Cavity Problem

Following the problem described in section 2, the steady-state flow in a two-dimensional lid-driven cavity problem, governed by the incompressible Navier-Stokes equations (3.16 to 3.19), is considered. The Reynolds number is $Re = 100$. The problem should then converge to a steady-state solution [112]. The network outputs are the stream function $\psi(x, y)$ and the pressure field $p(x, y)$. It is worth mentioning that in this problem, no training data is available inside the domain. The training is solely based on unsupervised learning using $5,000$ collocation points inside the domain, 200 boundary condition points, and 200 interface points on the boundaries of each subdomain.

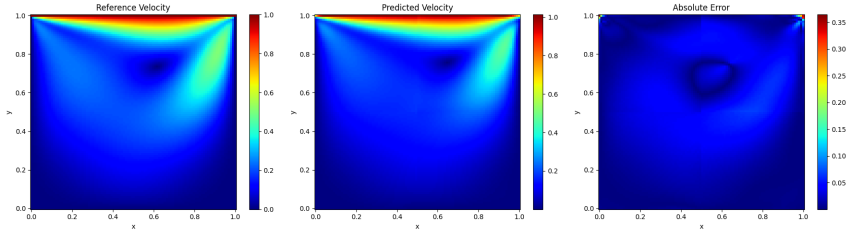
Loss Terms Contributions

As mentioned before, the velocity components can be derived from the stream function $\psi(x, y)$. To achieve this, Automatic Differentiation (AD) [38] is used to compute the derivatives of the stream function with respect to x and y. Based on different combinations of the loss terms explained in Section 2, we have formed six different loss functions, which are listed in Table 3.1. Case 1 represents the loss function that includes all essential terms to satisfy boundary conditions, the governing equations, and to preserve solutions continuity (as in basic decomposed-domain PINN). Additional constraints are introduced in cases 2 to 6, resulting in different loss functions.

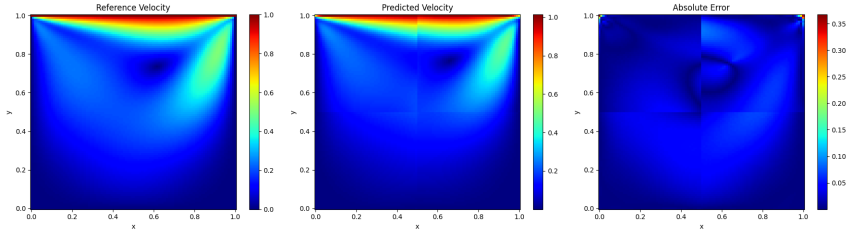
The accuracy associated with each case is given in Table 3.1 and Figure 3.5. In Figure 3.5-a which corresponds to the case with only essential constraints (No.1 in Table 3.1), the solution is weakly continuous at interfaces of subdomains. However, the incorporation of the gradients of the solutions in the loss function improves the continuity and the accuracy of the solution



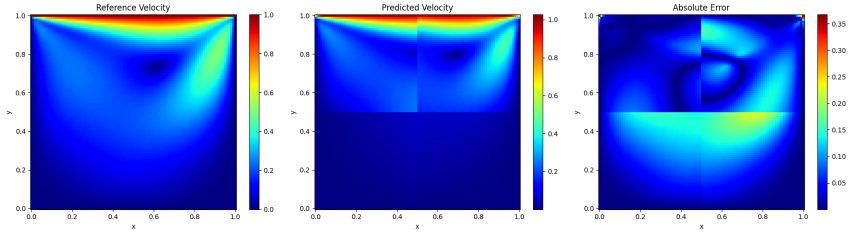
(a)



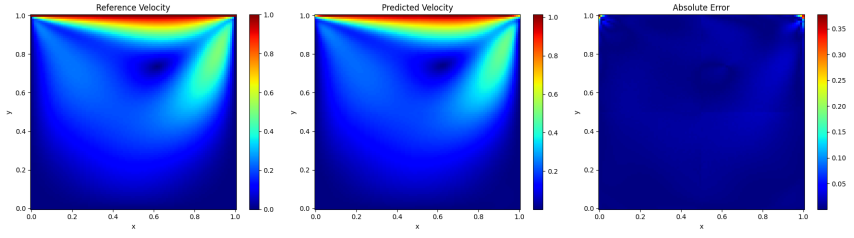
(b)



(c)

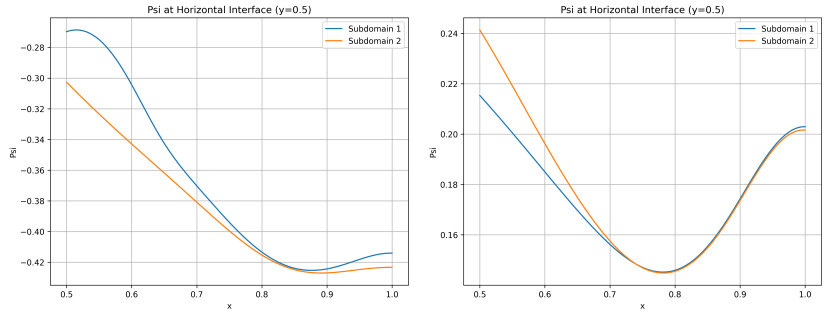


(d)

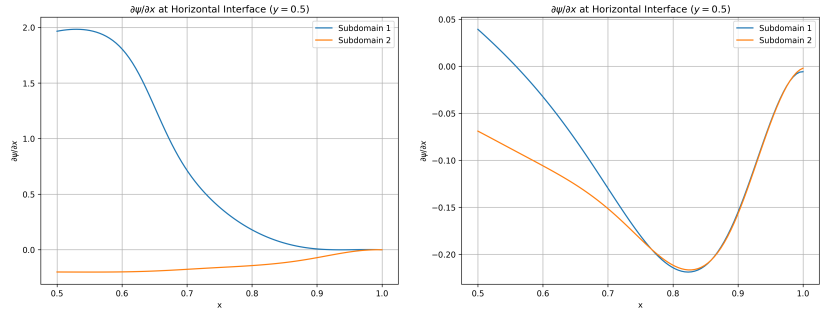


(e)

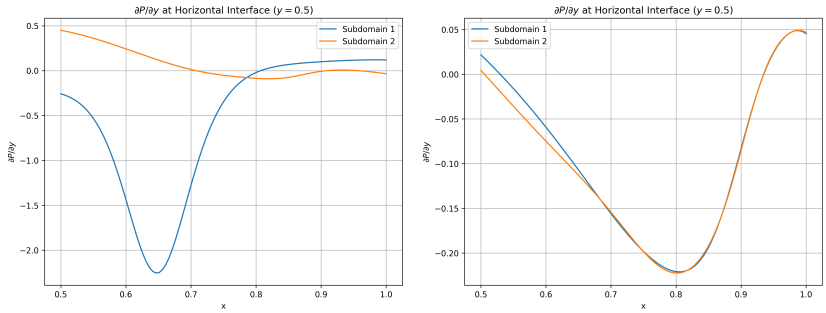
Figure 3.5 LDC problem: velocity fields; subplots (a), (b), (c), (d), and (e)are cases 1 to 5 of Table 3.1.



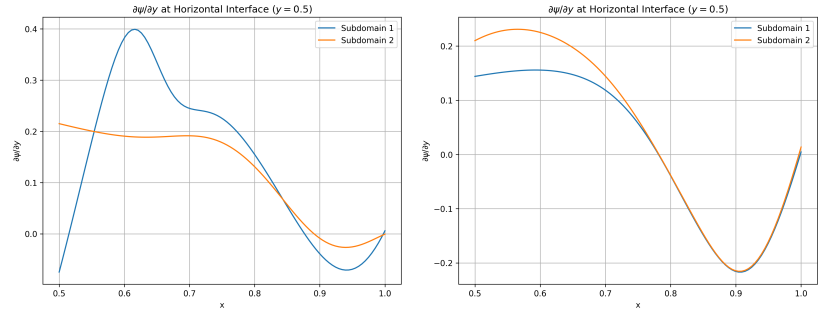
(a)



(b)



(c)



(d)

Figure 3.6 LDC problem; solution regularity analysis. The left column depicts results obtained using the baseline domain decomposition of PINNs, whereas the right column shows those derived from the DD-PINN, employing the H^1 norm of the loss terms. (a): Continuity of ψ at horizontal interface; (b): Differentiability of ψ at horizontal interface (x-direction); (c): Differentiability of P at vertical interface (y-direction); (d): Differentiability of ψ at vertical interface (y-direction).

Table 3.1 Various loss function: The training time remained consistent for all cases.

No.	Loss terms	Rel. L_2 Err.
1	Essentials: $L_{\mathrm{F}} + L_{\mathrm{b}} + L_{\psi_{\mathrm{int}}} + L_{p_{\mathrm{int}}}$	2.13×10^{-1}
2	Essentials + $L_{\nabla(\psi)_{\mathrm{int}}}$	9.90×10^{-2}
3	Essentials + $L_{\nabla(p)_{\mathrm{int}}}$	1.09×10^{-1}
4	Essentials + $L_{\mathrm{F}_{\mathrm{int}}}$	2.61×10^{-1}
5	Essentials + $L_{\nabla(\psi)_{\mathrm{int}}} + L_{\nabla(p)_{\mathrm{int}}}$	3.09×10^{-2}
6	Essentials + $L_{\nabla(\psi)_{\mathrm{int}}} + L_{\nabla(p)_{\mathrm{int}}} + L_{\mathrm{F}_{\mathrm{int}}}$	4.63×10^{-2}

(Figures 3.5-b and 3.5-c). The error is improved accordingly as presented in Table 3.1. The most accurate case includes gradients of both the stream function $\psi(x,y)$ and the pressure field $p(x,y)$ (Figure 3.5-e and case 5 in Table 3.1). This is analogous to using the H^1 norm of the error instead of L^2 in the MSE in the loss function

$$||\psi||^2_{H^1} = ||\psi||^2_{L^2} + ||\nabla\psi||^2_{L^2} \ . \tag{3.38}$$

$$||P||^2_{H^1} = ||P||^2_{L^2} + ||\nabla P||^2_{L^2} \ . \tag{3.39}$$

Consequently, the global solutions obtained using neural networks are not only continuous at interfaces but also continuously differentiable. Continuity of a solution implies that the solution function is well-defined and lacks abrupt changes or discontinuities. A continuous solution typically has a finite Euclidean norm, indicating that it doesn't exhibit wild oscillations or singularities.

$$||\psi||_{L^2}(\Omega) = \Big(\int_\Omega |\psi^2| dx\Big)^{\frac{1}{2}} \tag{3.40}$$

$$||P||_{L^2}(\Omega) = \Big(\int_\Omega |P^2| dx\Big)^{\frac{1}{2}} \tag{3.41}$$

However, differentiability extends the concept of continuity by requiring the existence of well-defined derivatives. Smooth solutions to PDE are often characterized by high degrees of differentiability, allowing for the calculation of gradients. H^1 norms provide a way to measure the differentiability of functions. For example, the Sobolev norm of ψ and P on a subdomain Ω is defined as:

$$||\psi||_{H^1}(\Omega) = \Big(\int_\Omega |\psi|^2 dx + \int_\Omega |\nabla\psi|^2 dx\Big)^{\frac{1}{2}} \tag{3.42}$$

$$||P||_{H^1}(\Omega) = \Big(\int_\Omega |P|^2 dx + \int_\Omega |\nabla P|^2 dx\Big)^{\frac{1}{2}}. \tag{3.43}$$

This H^1 norm quantifies both the continuity (through the integral) and the smoothness (through the gradient) of ψ and P over the domain.

DD-PINN extends the concept of basic domain decomposition techniques, which typically involve constraints on the average solution using the L^2 norm, to include constraints on the H^1 norm within loss terms. This extension significantly enhances the smoothness and regularity of solutions. Figure 3.6 confirms this concept. While using the L^2 norm of MSE in loss functions can preserve solution continuity (Figure 3.6-a), the plots in the left column (corresponding to the basic domain decomposition strategies) not only lack continuous differentiability but also exhibit abrupt changes in the approximations. Only the solutions obtained by DD-PINN using the H^1 norm of MSE are continuously differentiable and consequently smoother and of higher regularity (Figure 3.6-b, c, and d). Smoother and differentiable solutions in NN approximation of PDE can make training using gradient descent approaches more robust and less complicated. This is because differentiability allows for more stable gradient calculations, which facilitates convergence during training. Smoothness often leads to more predictable behavior of the neural network, making it easier to optimize. The higher accuracy of DD-PINN in Table 3.1, in case number 5 where H^1 norms of loss terms are employed, provides further confirmation of this concept.

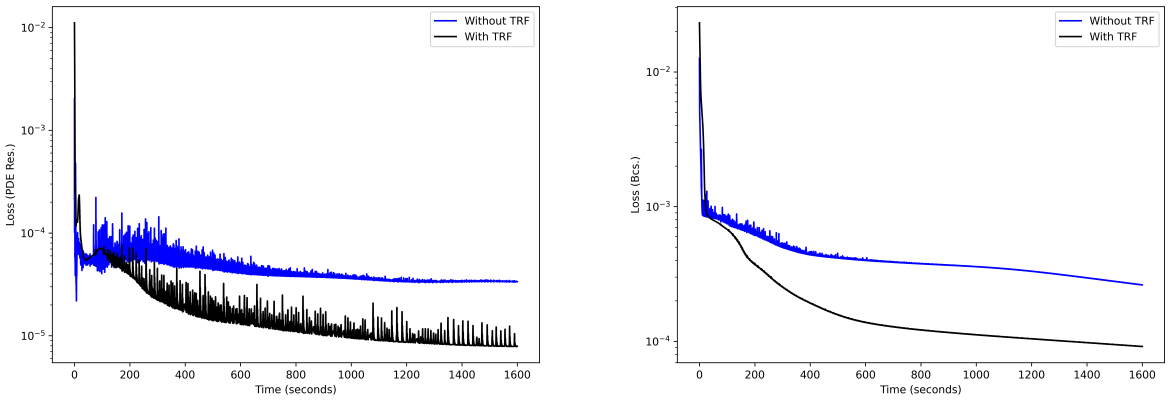
Finally, it is worth mentioning that including the residuals of the momentum equation at the interfaces significantly increases training times (Figure 3.5-d). Each computation of the corresponding loss term requires several extra passes through the computational graph due to the involvement of several first and second-order derivatives in the PDE residual for each interface. This makes optimization more challenging and convergence less likely.

Contribution of Sub-Network TRF

Incorporating TRF within each subdomain of the DD-PINN framework enables the mapping of input variables into a higher-dimensional feature space, thereby increasing the model's capacity to capture intricate patterns and relationships within the data. This mapping enhances the model's performance in several ways, leading to improved accuracy and a faster rate of convergence. Firstly, by mapping inputs into a higher-dimensional space, TRF facili-

Table 3.2 DD-PINN vs. DD-PINN + TRF. The training time remained consistent for both cases.

No.	Architecture	Rel. L_2 Err.
1	DD-PINN	4.60×10^{-2}
2	DD-PINN + TRF	2.34×10^{-2}



(a) (b)

Figure 3.7 Loss decay rate; (a): Residual loss; (b): BCs loss.

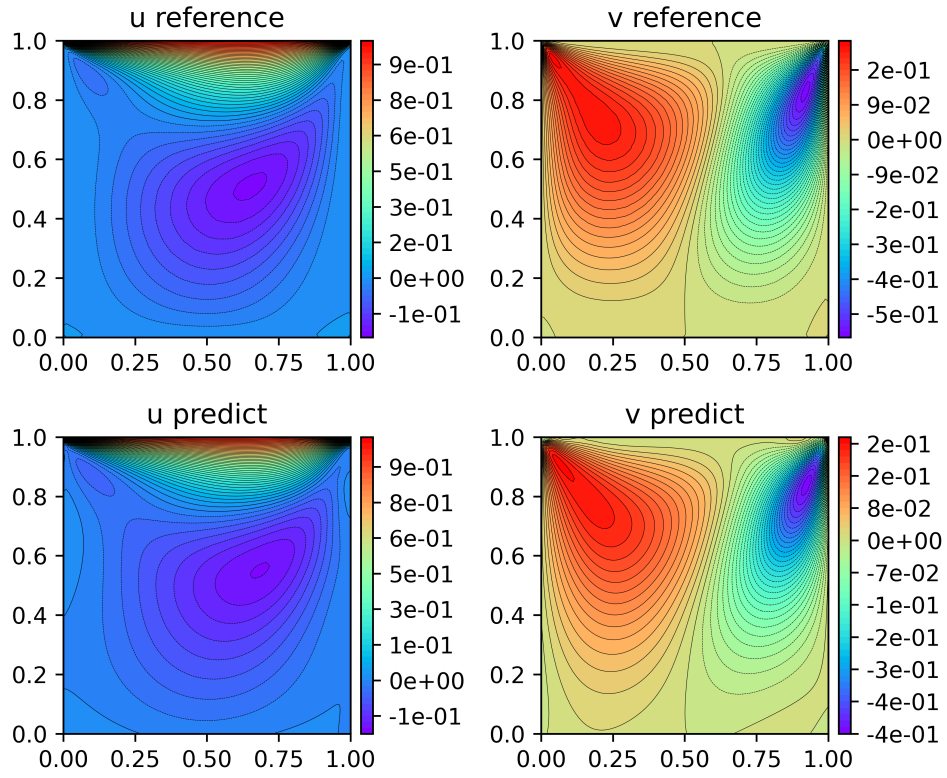


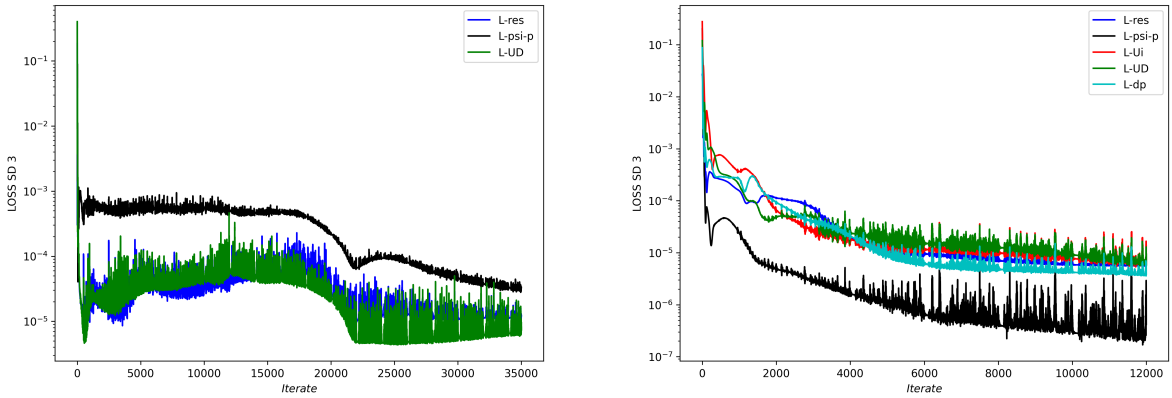
Figure 3.8 Velocity components contours: from top to bottom: reference solution, and DD-PINN with TRF.

tate the extraction of complex and nonlinear features from the data. This allows the model to better represent the underlying characteristics of the problem, leading to more accurate approximations of the solution. Furthermore, operating in a higher-dimensional feature space increases the model's degrees of freedom, providing it with greater flexibility to learn and adapt to the complexity of the problem. This enhanced representational capacity enables the model to capture subtle variations and nuances in the data, leading to improved performance. Additionally, the incorporation of residual connections within the TRF further enhances the hidden state representation, allowing for the efficient propagation of gradients during training. This facilitates faster convergence by alleviating the vanishing gradient problem and enabling smoother optimization trajectories.

The results of the comparison between the DD-PINN with and without inclusion of two transformer networks show that the use of TRF leads to better accuracy, within a specific time as shown in Table 3.2. By conducting this comparison, we aimed to discern the specific contribution of TRF to the model's performance independently from the domain decomposition strategy. Although each iteration of the optimization process takes longer, convergence is achieved faster when compared to the case without TRF. The runtime increase is problem- and transformer-architecture-dependent—42 %, 33 %, and 24 % in our three cases—yet DD-PINN still outperformed the baseline when runtime was matched. Quantitative data showing the loss decay rate is presented in Figure 3.7. As shown, convergence is achieved faster in the case with TRFs. Finally, Figure 3.8 represents a comparison of the velocity components along the x and y directions, demonstrating a good agreement between the approximation and the reference solution. In this comparison, the loss function used is the one specified in Case 5 of Table 3.1. To ensure a fair comparison, the training time is kept the same for both cases. For the DD-PINN model, the setting was 4 subdomains (2 subdomains in the x direction and 2 in the y direction), 5 hidden layers with 40 neurons in each layer, 5,000 collocation points, 200 boundary points, and 200 interface points at each common interface between subdomains. For the baseline PINN model, there was an FC-NN architecture with 5 hidden layers with 80 neurons in each of them, 20,000 collocation points, and 800 boundary points.

Combined Contribution

In Subsections 3.3.1 and 3.3.1, the contributions of DD-PINN and TRF were examined. To ensure accuracy, Subsection 3.3.1 focused on DD-PINN without TRF for a fair comparison. Similarly, in Subsection 3.3.1, DD-PINN with TRF was compared against DD-PINN without TRF to assess the specific contribution of TRF. This part explores the combined improvement



(a) (b)

Figure 3.9 Convergence rate analysis in a subdomain: Contribution of DD-PINN with sub-network TRF compared with basic decomposed-domain PINN (CPINN. (a): Baseline domain decomposition PINNs; (b): DD-PINN with sub-network TRF.

achieved by integrating both components, particularly in terms of convergence rate, against baseline domain-decomposed PINN. As shown in Figure 3.9, the loss decay rate for DD-PINN with TRF is more stable with fewer oscillations. In this figure, "L-res", "L-psi-p", and "L-UD" denote the losses corresponding to residuals of governing equations, solutions differences at the interface between two adjacent subdomains, and the boundary condition loss, respectively. "L-Ui" and "L-dp" represent the losses specifically due to the use of H^1 norm of MSE or the losses corresponding to the first derivatives of the network outputs. The results indicate that DD-PINN + TRF significantly enhances loss minimization across all loss terms, particularly demonstrating a substantial improvement of nearly 10^2 for 'L-psi-p'. Notably, the number of iterations required for DD-PINN + TRF was approximately one-third less compared to the baseline domain decomposition PINN. It's noteworthy that despite the reduction in iterations, the runtime remained nearly same due to the inclusion of TRF, ensuring a fair and accurate comparison.

Lastly, numerical results (Figure 3.10) are provided to verify the solution approximations made by the DD-PINN method against reference solution and to compare the solution accuracy against baseline PINN and the recent related works (namely the CPINN). The numerical results (velocity magnitude) confirm the agreement between the predictions made by the DD-PINN and the reference solution. DD-PINN also outperforms the baseline PINN and previous related works in terms of accuracy. In particular, the difference between the exact solutions and the solutions approximated by previous techniques, as well as the solutions' discontinuities at the interface, are highlighted by the red and the orange arrows in Figure

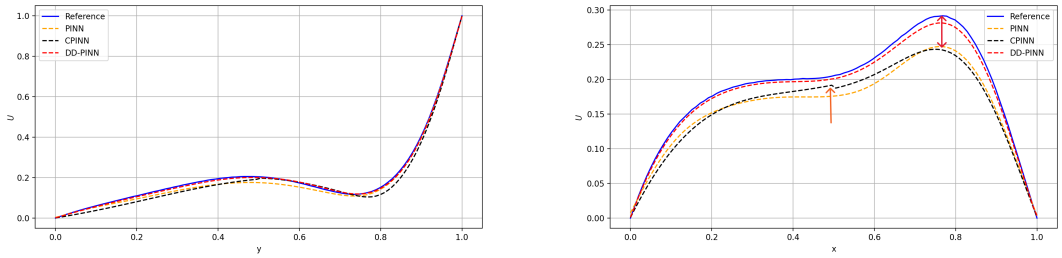


Figure 3.10 (a)

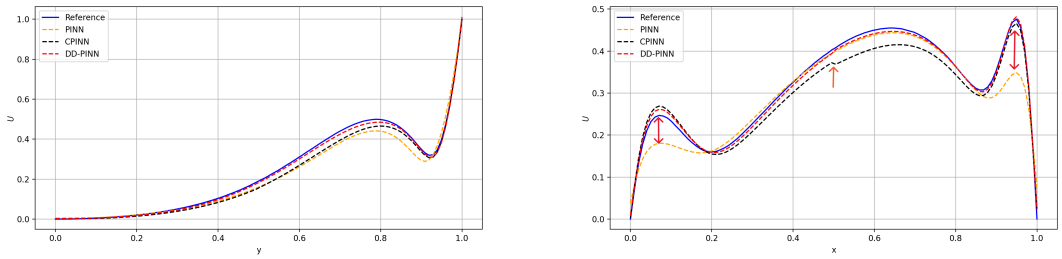


Figure 3.10 (b)

Figure 3.10 Numerical validation for lid-driven cavity flow problem: solution evaluation against reference solution and comparison against the baseline PINN and recent related works (CPINN). (a): solution at the sections corresponding to the $x = 0.5$ and $y = 0.5$ planes; (b): Solution at the sections corresponding to the $x = 0.9$ and $y = 0.9$ planes.

3.10, respectively. This numerical analysis verifies the approximations made by DD-PINN and underscores the significant contribution of utilizing the H^1 norm to enhance the overall solution regularity.

3.3.2 2D Cylinder Wake

The second example is a simulation of 2D vortex shedding behind a circular cylinder at a Reynolds number of $Re = 100$. The phenomenon is described by the 2D incompressible Navier-Stokes equations (3.16 to 3.19). The inlet condition is set as a free stream velocity with non-dimensional $u_\infty = 1$, the kinematic viscosity is $\nu = 0.01$, and the cylinder has a diameter of $D = 1$ located in $(x, y) = (0, 0)$. In this configuration, the system exhibits asymmetrical swirling vortices caused by vortex shedding, famously known as the Karman vortex street. To obtain reference data for training and testing, a high-fidelity DNS approach [27] is used. In this problem, we consider a domain size of the dimension $[1, 8] \times [-2, 2]$, and the time interval is $[0, 20]$ with a time-step of $\Delta t = 0.01$.

The inputs to the network model are spatio-temporal coordinates (x, y, t), and a two-dimensional output, resulting in a stream function $\psi(x, y, t)$ and pressure $p(x, y, t)$. In this problem, labeled training data inside the domain is available. This automatically satisfies the continuity of the global solutions in terms of velocity fields, as available training data are velocity components, but no pressure boundary condition is available. Our objective is to enhance the approximation accuracy of both the velocity and pressure fields. Particularly, one of the shortcomings of both the baseline PINN and the basic domain-decomposed PINN in approximating the Navier-Stokes equations, which has been addressed repeatedly in the literature [27, 80], is the "magnitude difference" between predicted and exact pressure fields in the absence of pressure training data. To improve the accuracy of pressure field approximation we implement a dominant subdomain approach. To be more detailed, the DD-PINN is used to split the domain into two subdomains, as illustrated in Figure 3.11. Initially, the domain decomposition is used without any interface conditions to compute the errors within each subdomain and identify the subdomain with the most significant error. Referring to Figure 3.12 and the color legend in Figure 3.13-b it becomes evident that the left subdomain exhibits a higher approximation error. Consequently, we enforce the dominant subdomain to conform to the solution of its neighboring subdomain at the interface, in terms of pressure. Subsequently, two additional loss terms, as described by Equations 28 and 30, are incorporated into the loss function of this subdomain. These additional terms aim to minimize the pressure mismatch at the interface with the adjacent subdomain, which features a more accurate pressure approximation (Figure 3.12 and Figure 3.13-b, the subdomain at the right), by preserving

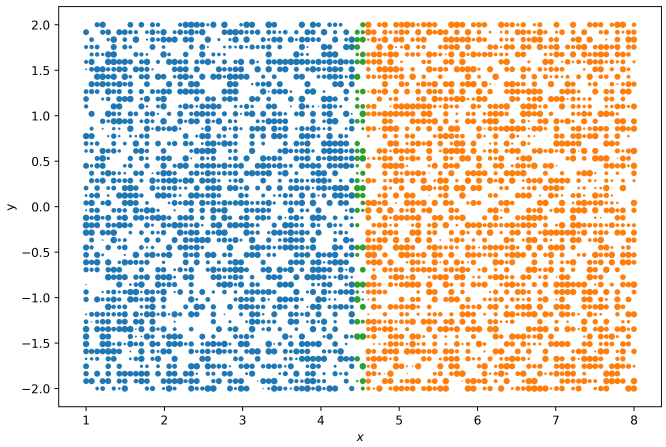


Figure 3.11 2D cylinder wake problem domain: blue and orange points represent the training and collocation points in the left and right subdomains and the green points correspond to the interface points.

pressure fields approximations' continuity and differentiability. As a result, the loss function for the left subdomain incorporates three components: a PDE residual term, a boundary condition mismatch term, and a pressure mismatch term at the interface,

$$L = MSE_u + MSE_{\mathrm{F}} + MSE_{p_{\mathrm{int}}}. \tag{3.44}$$

The gradient of the loss function of each subdomain is minimized separately and in parallel through a gradient descent process to find the optimal values of the network parameters for each subdomain (as illustrated in Figure 3.4). This technique not only enhances the accuracy of velocity approximation but also reduces the magnitude differences in the pressure field between the reference and the approximated values, as given in Table 3.3. Numerical results (pressure magnitude) in Figure 3.14 demonstrate that while both baseline PINN and CPINN struggle to accurately capture correct pressure magnitudes, the approximations made by the DD-PINN improved the accuracy of the predicted pressure magnitudes. This is because the subdomain with more accurate approximations (right) enforces the neighboring subdomain (left) to follow more accurate predictions through interface constraints. Table 3.3

Table 3.3 2D cylinder wake problem: Relative L_2 Error

Method	L_2 Err. u_x	L_2 Err. u_y	L_2 Err. P	Network
Baseline PINN	1.1×10^{-2}	2.6×10^{-2}	4.71	8×30
DD-PINN	6.2×10^{-3}	1.9×10^{-2}	2.8×10^{-1}	$(9) \times 35 + 7 \times 30$

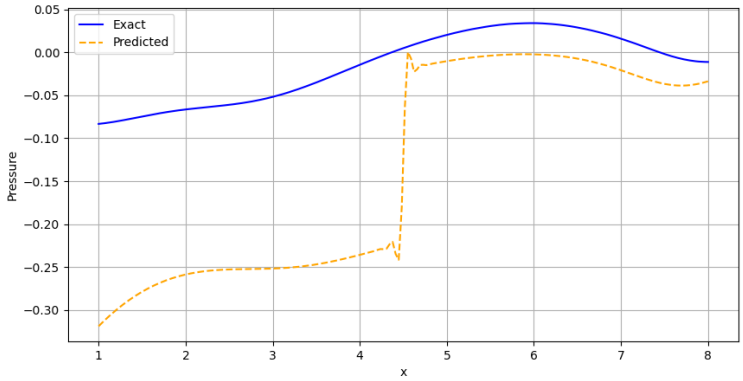
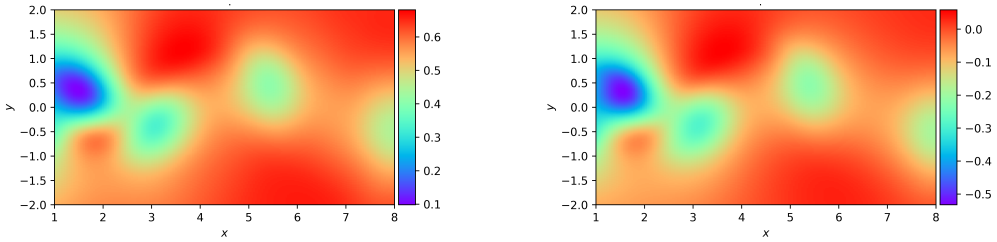
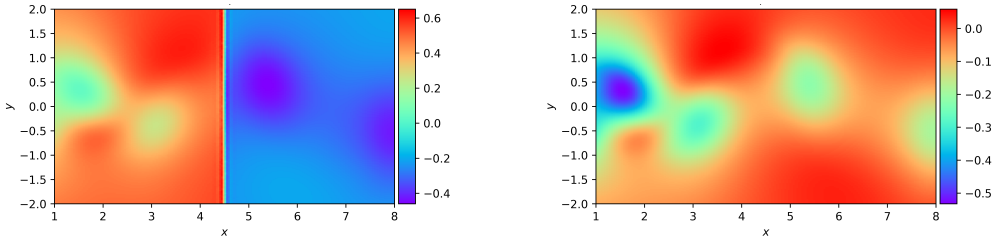


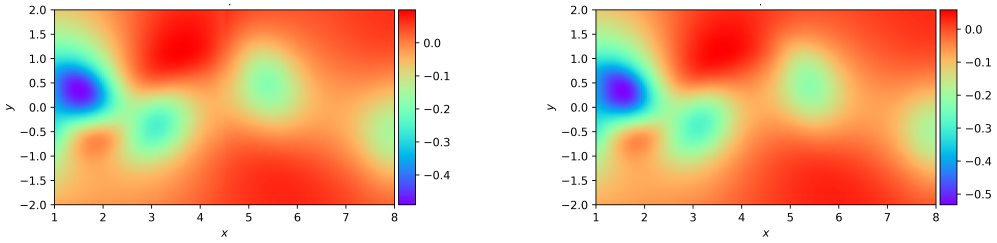
Figure 3.12 Dominant subdomain: analysis of subdomains in terms of solution accuracy



(a)



(b)



(c)

Figure 3.13 2D cylinder wake problem; Pressure fields. Left and right columns show approximated and reference pressure fields, respectively. (a): Baseline PINNs; (b): DD-PINN without interface constraints; (c): DD-PINN with interface constraints.

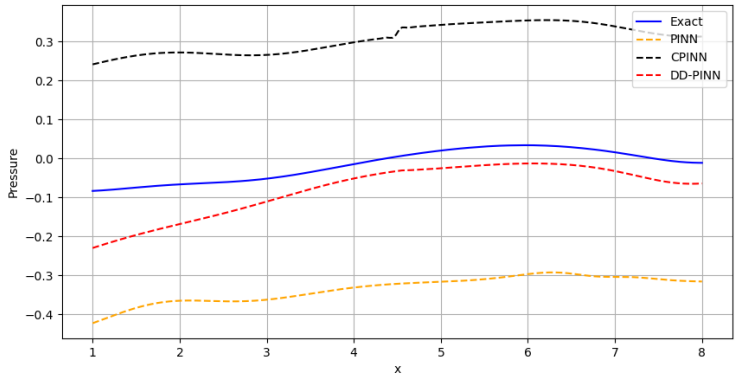


Figure 3.14 Numerical validation for 2D Cylinder Wake: solution evaluation against reference solution and comparison against the baseline PINN and recent related works (CPINN)

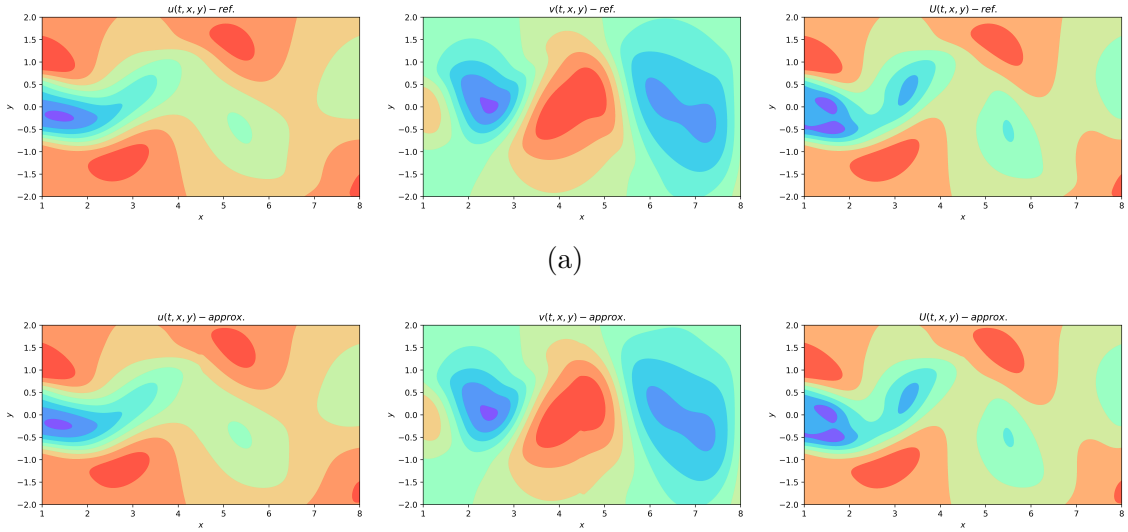


Figure 3.15 2D cylinder wake problem; velocity fields. From the left to the right: velocity components at x direction, y direction and velocity magnitude. (a): Reference solutions; (b): the DD-PINN.

and Figure 3.13 also confirm this conclusion. Additionally, velocity contours at the snapshot $t = 10$s are shown in Figure 3.15, demonstrating a good agreement between the reference and approximated solutions. As represented in Figure 3.15-b, the DD-PINN was able to capture the asymmetrical swirling vortices caused by vortex shedding with slightly better accuracy than the baseline PINN (as shown in Table 3.3).

3.3.3 Viscous Burgers Equation

In previous sections, two numerical examples governed by chaotic Navier-Stokes equations were examined to assess the efficacy of the DD-PINN. These investigations included a comparative analysis of its performance against the baseline PINN concerning accuracy. Additionally, comparisons were made with previous domain-decomposition PINN approach to evaluate both the accuracy and the continuity and differentiability of the global solutions. In this section, the DD-PINN methodology is applied to the one-dimensional viscous Burgers equation, serving as a numerical benchmark to validate the approach's accuracy. The viscous Burgers equation represents a simplified form of the Navier-Stokes equations, achieved by neglecting pressure and viscous terms. Despite its simplicity, this equation holds significant importance as a fundamental model in fluid dynamics. It finds wide-ranging applications in various fields, including computational fluid dynamics, shock wave theory, and turbulence modeling. The equation is expressed as:

$$u_t + uu_x = \nu u_{xx}, \qquad x \in [-1, 1],\ t \in [0, 1] \tag{3.45}$$

where $\nu = 0.01/\pi$. The initial condition is $u(x) = -\sin(\pi x)$, and the boundary conditions are $u(1, t) = u(-1, t) = 0$. The analytical solution, attainable through the Hopf-Cole transformation [118], elucidates the viscous Burgers equation's behavior. The nonlinearity in the convection term engenders highly pronounced solutions owing to the diminutive value of the diffusion coefficient ν. In the application of the DD-PINN, the spatial domain is discretized into three subdomains at interfaces $[-0.25, 0.25]$, accompanied by complementary interface conditions as outlined in Section 3.2. Three independent FC-NN models are deployed across these subdomains. The first and third models feature 6 hidden layers with 30 neurons each, while the intermediate subdomain utilizes 8 hidden layers with 40 neurons each. The choice of nonlinear activation function and neural network initialization method mirrors that of previous examples. In this instance, the network's input comprises two-dimensional data encapsulating x and t, while the output corresponds to the velocity field u. A total of 20,000 iterations are executed, with a stepwise reduction in the learning rate: $[1e-2, 1e-3, 5e-4, 1e-4, 1e-5]$ at iterations $[4000, 8000, 11000, 15000]$. The simulation

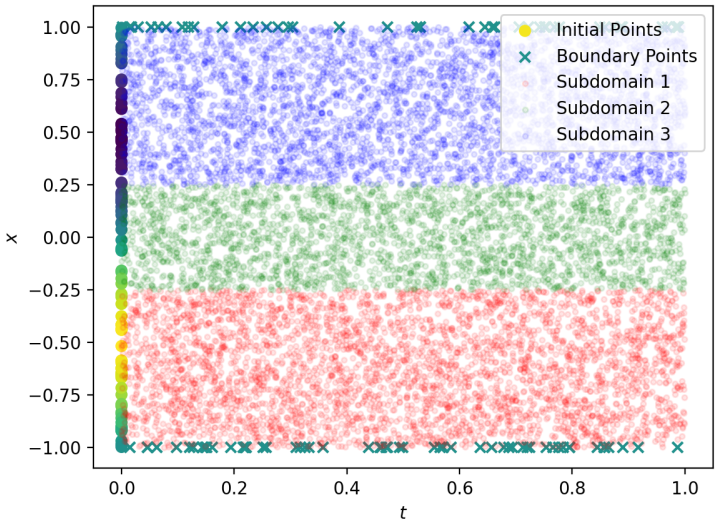


Figure 3.16 The problem domain: one-dimensional viscous Burgers equation.

involves 10,000 collocation points, 100 boundary points, 100 initial points, and 100 interface points, tailored to promote continuity and differentiability constraints. The problem domain is illustrated in Figure 3.16, emphasizing the discretization of the domain into three subdomains. Figure 3.17 displays the comparison between the exact solution and the DD-PINN solution at $t = 0.10$, $t = 0.35$, and $t = 0.75$. The predicted solution demonstrates accurate agreement with the exact solution. Additionally, Table 3.4 presents a numerical comparison of solution accuracy, contrasting the utilization of the average solution at the interface to maintain continuity of global solutions (akin to previous domain decomposition PINN methods) against the incorporation of the H^1 norm of MSE in the loss function (the DD-PINN approach). Through the inclusion of complementary loss terms in the loss function to preserve both continuity and differentiability of the global solution, the overall solution exhibits improvement and aligns closely with the exact solution.

Table 3.4 Viscous Burgers equation: Relative L^2 error of the solution at $t = 0.5$ s, using L^2 norm of MSE against using H^1 norm of MSE.

Method	Time frame (s)	L_2 Err. u
Using average solution	0.5	2.94×10^{-2}
DD-PINN: Using H^1 norm	0.5	1.07×10^{-2}

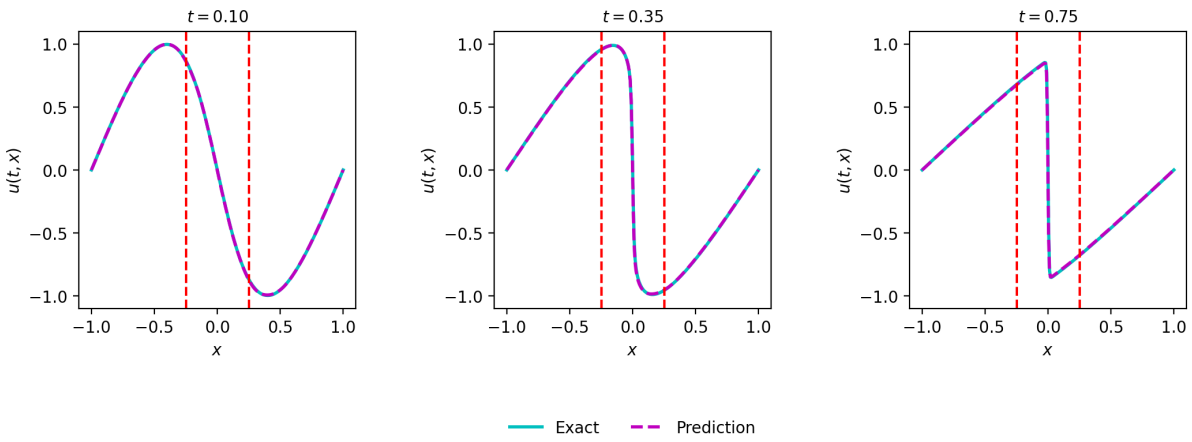


Figure 3.17 Comparison of exact and predicted solutions at $t = 0.10$, $t = 0.35$, and $t = 0.75$ for the viscous Burgers equation. The red dashed lines indicate the locations of the interfaces.

3.4 Conclusion

A new domain decomposition technique in Physics-Informed Neural Networks was introduced to directly approximate solutions of the incompressible Navier-Stokes equations. In this approach, the domain was discretized into subdomains and incorporated subdomain-specific loss terms that enforced effective interface constraints. In the first case study, which involved a steady-state flow in a two-dimensional lid-driven cavity, the networks were trained using unsupervised learning with collocation points and boundary conditions with no labeled data inside the domain. To preserve the continuity of the global solution, continuity constraints were enforced at common interfaces with adjacent subdomains. Furthermore, the effectiveness of using the H^1 norm of the MSE of the loss terms at interfaces was explored. This choice promoted the regularity of the solutions, preserving the differentiability of the solutions of the neural network model and improved the approximation accuracy. Furthermore, transformer networks were incorporated into each subdomain to map inputs into a higher-dimensional feature space, increasing the degrees of freedom and resulting in a faster rate of convergence.

In the second problem, the two-dimensional cylinder wake, by employing the dominant subdomain approach, the approximation accuracy of the pressure field was improved, and the magnitude difference of the approximated pressure field, which has been addressed repeatedly in previous works, was reduced. Incorporating domain-specific neural networks at each subdomain and enforcing continuity and differentiability constraints can significantly enhance the model's approximation capacity, especially for problems with complex physics, where a single network is unable to capture all the underlying physics accurately. Future research

could delve into a more comprehensive exploration of memory usage and the merits and demerits of parallel computation. Furthermore, in future work, we aim to investigate the compatibility of DD-PINN with recent advancements like adaptive refinement and importance sampling techniques. This integration could potentially lead to even more efficient and accurate training for a wider range of problems.

CHAPTER 4 ARTICLE 2: SIMULATION OF 3D TURBULENT FLOWS USING A DISCRETIZED GENERATIVE MODEL PHYSICS-INFORMED NEURAL NETWORKS

Authors: Amirhossein Khademi, Erfan Salari, and Steven Dufour

Journal of Nonlinear Mechanics, Elsevier, submitted on 2024-10-23, accepted on 2024-12-08, published on 2024-12-26

4.1 Introduction

Deep learning and scientific machine learning techniques have significantly advanced the field of numerical methods. In particular, artificial neural networks have played a crucial role in identifying and approximating complex nonlinear mechanics, providing powerful tools for solving challenging problems across various scientific domains [119–123]. These contributions span from using neural networks (NN) to accelerate traditional finite element methods (FEM) [124,125], to more purely NN-based approximation approaches. In particular, PINNs demonstrated significant advancements in both time efficiency and accuracy in various tasks, including advanced physics-informed reduced order modeling (ROM) [126,127] and approximating solutions to PDEs. Introduced by Raissi et al. [15, 27], PINNs offer an innovative approach that bypasses the necessity for mesh generation and numerical discretization by leveraging automatic differentiation [38]. A key advantage of PINNs is their reduced reliance on labeled data and their incorporation of the underlying physics of the model. This is achieved by integrating the residuals of governing equations into the loss function. Despite notable performance improvements, PINNs face challenges with high-dimensional problems like approximating solutions to the three-dimensional Navier-Stokes equations, including convergence issues and prolonged training times. Ongoing research aims to address these limitations.

Building upon the concept of data clustering [39], domain decomposition techniques, namely conservative PINNs (CPINN) [40] and extended PINNs (XPINN) [41], and parallel PINNs [42] were subsequently introduced. They involve the utilization of separate models for distinct subdomains. This approach enables the adjustment of hyperparameters for each model. To enhance the regularity of the global solution, the average solution at the interface with adjacent subdomains was incorporated in the loss function as an extra loss term.

In a similar research line focusing on the concept of domain discretization by Kharzami

et al. [44, 45], a variational approach was incorporated into the PINNs framework to enhance accuracy and reduce training costs. This integration resulted in the development of non-overlapping decomposed domain models along with projections onto a space composed of high-order polynomials. In this context, the trial space corresponds to the NN space, defined globally across the entire computational domain, while the test space consists of piecewise polynomials. Khademi et al. [128] further advanced the PINNs framework by introducing domain decomposition techniques and sub-domain transformer networks. These innovations prevented the error propagation and enable the projection of input spaces into higher-dimensional feature spaces, significantly enhancing the ability of PINNs to handle complex multi-physics problems and systems characterized by highly discontinuous solutions.

Another significant challenge faced by PINNs is numerical stiffness, which leads to unbalanced back-propagation gradients during the training process. Extensive efforts have been made to tackle this issue [50, 61, 63–66, 85, 129], primarily employing regularization, penalization, or learning rate annealing techniques. These approaches aim to balance the coefficients of loss terms. However, manually adjusting the coefficients of loss terms is not only time-consuming but also generally inefficient due to the high sensitivity of divergence to these coefficient values. Additionally, these values are problem-specific. To address the challenge of determining optimal penalty coefficients, Wang et al. [50] proposed an adaptive learning rate, drawing inspiration from another work by Kingma et al. [70], to balance the interplay between the various loss terms. This approach computes the coefficients using a moving average scheme based on the ratio of different loss terms. Nevertheless, the observed enhancement resulting from this method is marginal, particularly in addressing problems with higher dimensions. To expand the concept of such regularization to time-dependent problems, the author introduced the causal PINNs [130] to simulate time-dependent dynamical systems exhibiting multi-scale, chaotic, or turbulent behavior. In the causal PINNs framework, loss terms are multiplied by weighting coefficients, allowing a specific loss term $L_r(t_i, \theta)$ to be minimized only if $L_r(t_k, \theta)_{k=1}^{i}$ before t_i are properly minimized. Song et al. [72] recently introduced LA-PINN, a dynamic and point error-based weighting model. LA-PINN utilizes independent loss-attentional networks for each loss component, dynamically updating weights for individual training points during the training process, to improve convergence in stiff regions.

Inspired by the concepts of the XPINN [41] and the causal PINNs [130], Penwarden et al. [43] proposed a unified framework aimed at describing various causality-enforcing PINNs techniques, specifically applicable to time-dependent PDE. This framework not only encompasses existing methods but also provides a platform for developing new causality-enforcing techniques. By leveraging this unified framework, they successfully reduced training times, enhancing the scalability of PINNs on problems that don't pose significant training chal-

lenges. To resolve the issue of extended training time, Psaros et al. [57] proposed to incorporate a metalearning approach within PINNs. This work focused on enhancing the baseline PINNs' performance for out-of-distribution tasks and offered a more efficient allocation of computational resources during training. The application of this method to real-world problems is yet to be explored. In a similar work, Penwarden et al. [58] conducted a review of model-agnostic metalearning and transfer learning principles, then applied a model-aware adaptation of metalearning to PINNs. The proposed approach is tested on various canonical forward-parameterized PDEs, demonstrating the potential for accelerating optimization in PINNs.

Early deep neural networks (DNN) and PINNs models were mostly limited to generic and canonical problems [15, 27, 40, 41, 44, 50, 131]. Recent progress has enabled the study of more realistic scenarios through the use of increasingly sophisticated models. Biswas et al. [132] applied PINNs to simulate three-dimensional laminar flow governed by the Navier-Stokes equations. Their work demonstrated the capability of PINNs to handle complex three-dimensional fluid dynamics problems without relying on traditional numerical methods. Hu et al. [133] presented an innovative approach that combines a physics-informed neural network with a characteristic-based split algorithm to solve Navier–Stokes equations more efficiently. This method reduces the need for extensive labeled data, cuts down memory usage and computational time, and improves the accuracy and convergence of solutions for 3D incompressible flows. Hijazi et al. [134] introduce a Reduced Order Model (ROM) combining POD-Galerkin methods and PINNs to address inverse problems in the Navier–Stokes equations efficiently. Their approach enhances computational efficiency and accuracy by integrating physical laws directly into the loss functions of deep learning models. Cho et al. [135] developed a separable physics-informed neural network (SPINN) that optimizes the solving of multi-dimensional PDEs by minimizing computational and memory needs, thus facilitating quicker and more precise solutions on regular GPUs. Anagnostopoulos et al. [136] introduce a residual-based attention mechanism for PINN framework, which accelerates convergence by focusing on high-error regions without additional computational costs. Their method significantly improves the accuracy and speed of solving PDEs by dynamically adjusting the training focus, leading to an order of magnitude faster convergence in benchmark problems. Jin et al. [137] develop the Fourier Warm Start (FWS) algorithm to improve PINNs by balancing convergence across frequencies and reducing spectral bias. Their Fourier Analysis Boosted PINN (Fab-PINN) shows substantial performance gains, notably reducing L^2 errors and accelerating convergence in complex PDE scenarios. Wang et al. [138] introduce stacked deep learning models for quickly approximating steady-state Navier-Stokes equations at low Reynolds numbers. Their weakly supervised approach captures boundary and geometric conditions

without requiring labeled data, significantly reducing computational demands and enabling fast, high-quality fluid flows simulations.

The study of real-world high-dimensional systems typically present the challenge of turbulence, necessitating sophisticated models to accurately analyze fluid dynamics and Navier-Stokes equations. Beck et al. [139] proposed a novel data-based technique to derive large eddy simulations (LES) subgrid closure terms using a deep neural networks model. Zhao et al. [140] introduced a novel computational fluid dynamics (CFD)-based machine learning framework tailored for the development of Reynolds-averaged Navier-Stokes (RANS) models. This approach utilizes CFD simulations to inform and refine machine learning algorithms, enhancing the accuracy of RANS modeling. To showcase the efficacy of this method, it was employed in the context of developing models for wake mixing in turbomachines. Notably, the application of the CFD-driven machine learning framework led to significant improvements in the predicted wake mixing profiles. Zhang et al. [121] later proposed a bi-fidelity shape optimization method for turbulent fluid-flow applications, integrating LES and RANS models within a hierarchical-Kriging surrogate framework. The model enhanced RANS models to capture LES behavior, improving correlation across the design space. Demonstrating efficacy on the periodic-hill case, this approach converged to the LES-optimum with just two LES samples, outperforming standard RANS models. Jin et al. [85] used PINNs to simulate 3D time-dependent turbulent channel flow at $Re \approx 1000$ without a turbulence model. They investigated loss function weighting for data and physics balance, achieving good agreement with DNS outcomes but with high training time. However, it was achieved at a notably high training time. Using a similar direct modeling strategy, Hanrahan et al. [141] utilize physics-informed neural networks (PINNs) with sparse data to effectively model turbulent flows. Their approach, which does not use turbulence models, enables the network to infer key dynamics like Reynolds-stress fields. This study shows that PINNs can robustly predict complex turbulent behaviors using limited experimental data.

Given the literature review, using PINNs and their extensions for solving the Navier-Stokes equations has demonstrated prospective results. Specifically, enhancing the performance of the baseline PINNs through decomposition strategies, metalearning, and transfer learning shows promise. However, significant computational cost, prolonged training time, challenging solution regularity, and high approximation error in complex phenomena in high dimensional time-dependent PDEs have constrained the applicability of these methods to generic problems or restricted their use to very small computational domains in realistic high-dimensional fluid flow problems.

In this study, a novel domain decomposition technique for the PINNs is introduced. This

technique involves discretizing the global domain into multiple overlapping subdomains, allowing for parallel solutions of each subdomain. A generative approach in the Sobolev function space is introduced such that subdomains predict and propagate new training data to be used as IC in the subsequent subdomains. A gating mechanism is implemented to control the interconnection between parallel models, permitting optimal global training. To extend the application scope of PINNs beyond generic problems, we have verified this approach using a real-world problem involving turbulent channel flow governed by the 3D time-dependent Navier-Stokes equations. The results demonstrate that approximations by the DG-PINN closely align with the reference solution. Moreover, comparative numerical analyses reveal that DG-PINN improves both training time and convergence rates. Notably, DG-PINN mitigates the likelihood of solution divergence, a challenge commonly encountered in baseline PINNs, especially in scenarios involving 3D turbulent fluid flow problems, where manual adjustment of loss terms coefficients is crucial to prevent divergence. Below are the key contributions of this study:

- Introduction of a novel domain discretization technique for PINNs aimed at reducing computational cost and enhancing scalability for high-dimensional turbulent fluid flow problems.
- Implementation of a gating mechanism to control the interaction between parallel models dynamically, ensuring accuracy while minimizing computational overhead.
- Introduction of a generative modeling approach within PINNs to optimally approximate and transfer initial conditions to subsequent models, thereby reducing approximation error in subsequent time steps where the problem lacks initial conditions, thereby enhancing approximation's accuracy.
- Enhancing the physics-informed part of PINNs by leveraging the H^1 norm of error to effectively promote solutions' regularity.
- Implementation of a distributed training strategy.
- Expanding the applicability of existing domain decomposition PINN methods: The proposed approach is validated through a real-world application involving turbulent channel flow governed by the 3D time-dependent Navier-Stokes equations, demonstrating the potential of PINNs to tackle complex fluid flow problems.

4.2 Methodology

4.2.1 Neural Networks

According to the universal approximation theorem, a multi-layer perceptron having only one hidden layer and a limited number of neurons can accurately approximate any continuous function to an arbitrary degree of precision [21, 113, 114]. In this study, we approximate solutions to the Navier-Stokes equations using multiple deep sequences of interconnected neurons, structured in a fully connected feed-forward (FFNN) architecture, formulated as

$$f_i(\boldsymbol{x}_i;\boldsymbol{\omega}_i,\boldsymbol{b}_i) = \alpha_i(\boldsymbol{\omega}_i \cdot \boldsymbol{x}_i + \boldsymbol{b}_i) \tag{4.1}$$

where $\boldsymbol{x}$ is the input, $\boldsymbol{\omega}$ denotes the weight vector, $\boldsymbol{b}$ represents the bias vector and α is a non-linear activation function. An FFNN is a type of architecture of artificial neural network where the connections between its neurons are structured so that each neuron in one layer is connected to every neuron in the subsequent layer. "Fully connected" implies that all neurons from one layer are connected to every neuron in the following layer without skipping any connections. Information flows from the input layer, through one or more hidden layers, and finally to the output layer without any feedback loops or cycles, hence the term "feed-forward". In this network architecture, each neuron's output is determined by applying a weighted sum of the inputs from the preceding layer, followed by an activation function that introduces non-linearities to the network. These activation functions enable the neural network to learn and model complex relationships within the data. Feed forward neural networks are foundational in machine learning and are used in various applications, including pattern recognition, regression, classification, and more complex tasks such as natural language processing, image recognition, and approximation and identification of differential equations.

In our designated NN model to approximate solutions to the Navier-Stokes equations, spatiotemporal coordinates, $\boldsymbol{X} = (t, x, y, z)$, are given to the neural network while instantaneous velocity and pressure fields, $\boldsymbol{U}(\boldsymbol{X}) = (u, v, w)$ and $P(\boldsymbol{X})$, are directly approximated as the model's output, such that

$$\boldsymbol{U}_\theta(\boldsymbol{X}), P_{\boldsymbol{\theta}}(\boldsymbol{X}) = f_{NN}(\boldsymbol{X}, \theta). \tag{4.2}$$

The neural network function is denoted by $f_{NN}(\boldsymbol{X}, \theta)$, where θ represents the parameters (weights and biases) of the neural network. The training involves adjusting the parameters θ of the neural network to minimize the difference between the predicted solutions $\boldsymbol{U}_\theta(\boldsymbol{X})$ and the reference solutions $\boldsymbol{U}(\boldsymbol{X}) = (u, v, w)$.

4.2.2 PINN

Physics-Informed Neural Networks are a class of machine learning models that leverage both data and known physical laws to make predictions. They have gained popularity in scientific computing and engineering for their ability to incorporate prior knowledge of underlying physics into the learning process. PINNs offer a promising approach for solving inverse problems, system identification, and modeling complex physical systems. Let $u(x,t)$ represent the solution to a physical system, where $x \in \Omega \subset \mathbb{R}^d$ denotes the spatial coordinates and $t \in [0,T]$ denotes time. The governing equation describing the system can be written as:

$$\frac{\partial u}{\partial t} + \mathcal{N}(u;x,t) = 0, \tag{4.3}$$

where $\mathcal{N}(u;x,t)$ represents the nonlinear operator governing the dynamics of the system. PINNs aim to learn the solution $u(x,t)$ directly from data while satisfying the physics described by Equation 4.3. This is achieved by minimizing a loss function consisting of two components: a data fidelity term and a physics-informed regularization term. The data fidelity term penalizes the discrepancy between the predicted solution and observed data, typically defined as:

$$L_{\text{data}} = \sum_{i=1}^{N_{\text{data}}} \|u(x_i,t_i) - u_{\text{obs}}(x_i,t_i)\|^2, \tag{4.4}$$

where N_{data} is the number of data points, (x_i,t_i) are the spatial-temporal coordinates of the data points, and $u_{\text{obs}}(x_i,t_i)$ are the corresponding observed values.

The physics-informed regularization term enforces the satisfaction of the governing PDE in Equation 4.3. This is typically formulated as

$$L_{\text{physics}} = \sum_{j=1}^{N_{\text{phys}}} \|\mathcal{N}(u(x_j,t_j);x_j,t_j)\|^2, \tag{4.5}$$

where N_{phys} is the number of physics-informed points sampled from the domain. The overall loss function for training the PINNs model is given by

$$L_{\text{PINN}} = L_{\text{data}} + L_{\text{physics}}. \tag{4.6}$$

4.2.3 Related Works

XPINN: Jagtap et al. [41] proposed an extension to the CPINN [40] to decompose spatio-temporal domain into subdomains and using independent models in each subdomain that

are linked to adjacent subdomain through average solution at the interface, such that:

$$u_{\text{average}} = \frac{u_1 + u_2}{2}. \tag{4.7}$$

In this settings, u_1 and u_2 are the solutions at the interface by two independent adjacent models. The continuity of the global solution is then preserved by enforcing $MSE(u_{average} - u_1) + MSE(u_{average} - u_2) = ||u_{average} - u_1||_2^2 + ||u_{average} - u_2||_2^2$. This approach lacks causality and encounters training challenges similar to those seen in standard PINNs. These issues can be exacerbated when dealing with interfaces and separate networks, as they pose a more complex optimization problem by disregarding the causal structure during the training process for time-dependent PDEs [130]. To elaborate, the effectiveness of transfer learning for subsequent models relies on the prior convergence of solutions from previous time steps.

Stacked Decomposition: Penwarden et al. [43] expanded the concept of the XPINN to make it more applicable on time-dependent PDE. Using the causality in PINNs [130], causal weights were applied to the loss function of subdomains to take the significance of causality into account. In this settings, the loss function of the PINNs is re-written as below:

$$L_r(\theta) = \frac{1}{N_t} \sum_{i=1}^{N_t} \exp\Big(-\epsilon \sum_{k=1}^{i-1} L_r(t_k, \theta) \Big) L_r(t_i, \theta) \tag{4.8}$$

where ϵ is the causality hyperparameter. Consequently, the $L_r(t_i, \theta)$ will be minimized only if the previous residuals $L_r(t_k, \theta)_{k=1}^{i-1}$ are minimized to a predefined value. Additionally, the stacked-decomposition method is characterized by two parameters, namely n and dS. The duration covered by each subdomain over time is determined based on the total time domain of the problem and the number of partitions n. When $dS = 1$, the stacked-decomposition process occurs sequentially rather than concurrently. Conversely, when $dS = n$ and using XPINN interface conditions with all domains activated at the onset of training, the stacked-decomposition method is equivalent to the conventional XPINN approach. The illustrations depicting the decomposition of the XPINN, the Stacked-Decomposition, and the DG-PINN, as described in the following section, are presented in Figure 4.1. Also, Table 4.1 represents a brief specification of these methods and highlights their differences and contributions.

4.2.4 Discretized Generative Model Physics-Informed Neural Network

The domain discretization technique involves decomposing the domain into multiple overlapping subdomains and using separate models with subdomain-specific hyperparameters and parallel training. This approach enables the representation of the entire domain by combining

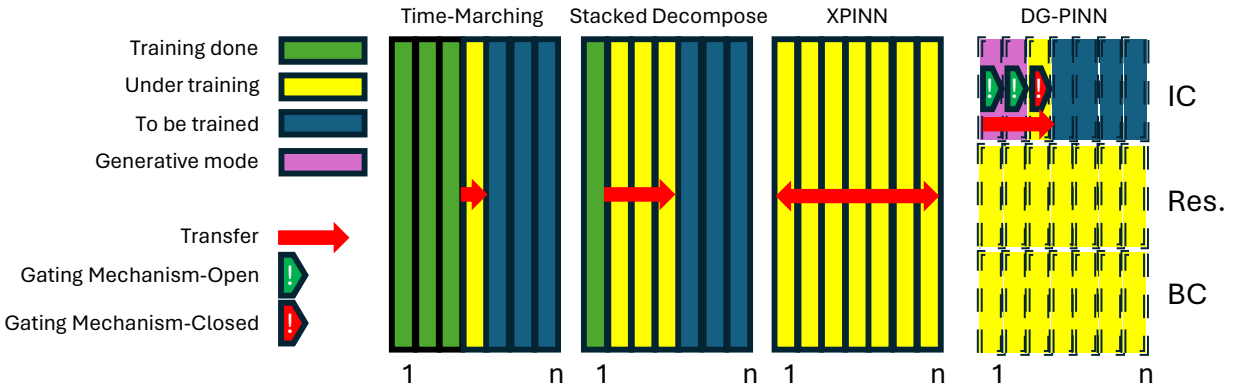


Figure 4.1 Illustration of existing decomposition methods compared with the DG-PINN

Table 4.1 Classification and comparison of related methods with the DG-PINN.

Method	Soft Causality	Hard Causality	Function Space	Parallel Computing	Reported Application
XPINN	—	—	L^2	Parallel	1D Burgers eq.
Stacked-Decomposition	Weights	Time marching	L^2	Partially parallel	Convection eq. Allen-Cahn eq. Korteweg–de Vries eq.
DG-PINN	Gating	Generative modeling	H^1	Parallel	3D time-dependent Navier-Stokes (Turbulent)

these subdivided regions as

$$\Omega = \cup_{i=1}^{N_t} \Omega_i \ , \tag{4.9}$$

where

$$\Omega_i \cap \Omega_j = \Omega_{ij} \ , \tag{4.10}$$

where N_t shows the total number of partitioned subdomains and Ω_{ij} is the overlapping zone between any two adjacent subdomains. Within this context, the approximation of the solution within the i_{th} subdomain by the i_{th} model is

$$\boldsymbol{u}_{\theta_i} = f_{NN}^{i}(\boldsymbol{x}, \boldsymbol{\theta}_i), \tag{4.11}$$

where $\boldsymbol{u}_{\theta_i}$ is the latent solution and f_{NN}^{i} is the neural network model in the i_{th} subdomain, with parameters $\boldsymbol{\theta}_i$. A problem, involving 3D time-dependant turbulent channel flow is considered. This flow is governed by the incompressible Navier-Stokes equations which are written in the VP form as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{1}{Re} \Delta \boldsymbol{u} = 0 \qquad \text{in } \Omega \times (0, T], \tag{4.12}$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \text{in } \Omega \times (0, T], \tag{4.13}$$

$$\boldsymbol{u} = u_\Gamma \qquad \text{on } \Gamma_D \times (0, T], \tag{4.14}$$

$$\frac{\partial \boldsymbol{u}}{\partial n} = 0 \qquad \text{on } \Gamma_N \times (0, T]. \tag{4.15}$$

where the non-dimensional time is denoted by t. $\boldsymbol{u}(x, y, z, t) = [u, v, w]^T$ refers to the non-dimensional velocity vector, while P represents the non-dimensional pressure. The term Re stands for the Reynolds number, which characterizes the flow by comparing inertial forces with viscous forces. Reynolds number is defined as $Re = \rho \ u \ L/\mu$, where ρ signifies fluid density, u is the characteristic velocity, L denotes the characteristic length scale, and μ represents dynamic viscosity. The Dirichlet and Neumann boundaries are provided by 4.14 and 4.15, respectively.

In this study, we are interested in approximating solutions to the Navier-Stokes equations, using multiple neural networks, novel physics-informed parts, and a distributed optimization strategy based on a mini-batch gradient descent approach.

For the VP form of the time-dependent Navier-Stokes equations 4.12 to 4.15, the residuals of the PDEs, including residuals of the momentum equation and the divergence-free constraint, could be written as

$$R_x = \partial_t u + u\partial_x u + v\partial_y u + w\partial_z u + \partial_x p - \frac{1}{Re}(\partial^2_{xx} u + \partial^2_{yy} u + \partial^2_{zz} u) \tag{4.16}$$

$$R_y = \partial_t v + u\partial_x v + v\partial_y v + w\partial_z v + \partial_y p - \frac{1}{Re}(\partial^2_{xx} v + \partial^2_{yy} v + \partial^2_{zz} v) \tag{4.17}$$

$$R_z = \partial_t w + u\partial_x w + v\partial_y w + w\partial_z w + \partial_z p - \frac{1}{Re}(\partial^2_{xx} w + \partial^2_{yy} w + \partial^2_{zz} w) \tag{4.18}$$

$$R_c = \partial_x u + \partial_y v + \partial_z w \tag{4.19}$$

where R_x, R_y, R_z, and R_c refers to the residuals of the momentum equations in x, y, and z directions and divergence free constraint, respectively. For computations of partial differential operators automatic differentiation [38] is used. To approximate the derivatives in the governing equations, the derivatives of the outputs of the computational graph are calculated with respect to x, y, z, and t.

In PINNs settings, the approximation problem is transformed into an optimization problem of the network parameters $\boldsymbol{\theta}$, where minimization of the function associated with the approximation yields optimal solutions. The associated function is called the loss function which is written as

$$L = L_{IC} + L_{BC} + L_R \tag{4.20}$$

and the loss terms are written as

$$L_{\mathrm{IC}} = \frac{1}{N_{\mathrm{I}}} \sum_{n=1}^{N_{\mathrm{I}}} \left| \boldsymbol{u}_\theta^n - \boldsymbol{u}_{\mathrm{I}}^n \right|^2 \tag{4.21}$$

$$L_{\mathrm{BC}} = \frac{1}{N_{\mathrm{B}}} \sum_{n=1}^{N_{\mathrm{B}}} \left| \boldsymbol{u}_\theta^n - \boldsymbol{u}_{\mathrm{B}}^n \right|^2 \tag{4.22}$$

$$L_{\mathrm{R}} = \frac{1}{N_{\mathrm{R}}} \left(\sum_{n=1}^{N_{\mathrm{R}}} \left| R_x^n \right|^2 + \sum_{n=1}^{N_{\mathrm{R}}} \left| R_y^n \right|^2 + \sum_{n=1}^{N_{\mathrm{R}}} \left| R_z^n \right|^2 + \sum_{n=1}^{N_{\mathrm{R}}} \left| R_c^n \right|^2 \right), \tag{4.23}$$

where L_{IC}, L_{BC}, and L_R correspond to the error associated with the approximations of the IC, BC, and residuals of the governing PDEs, respectively.

When applying the baseline PINNs approach to time-dependent PDEs, optimizing the network becomes challenging. Although capturing all the chaotic behavior of the PDE in different time steps with a single model is not impossible based on the universal approximation theorem [21, 113, 114], this becomes highly time-consuming and prone to divergence. Furthermore, the absence of IC compared to the evolving constraints across the time domain, such as boundary conditions (BC) and PDE residuals, results in noticeable approximation errors in subsequent time steps.

Here we put a novel decomposition technique into practice. The temporal domain is discretized such that there is an overlapping zone between any two adjacent subdomains (the zones highlighted in yellow in figure 4.2). Since the IC is provided solely within the first subdomain, the subdomain's model approximates the latent solution at the spatial coordinates corresponding to the IC, but at the time step corresponding to the overlapping zone. With this generative model, the neural network outputs from one model are used as initial conditions in the subsequent model for training (Represented by the red arrow in figure 4.3). In this setting, the loss term associated with the approximation of IC in the subsequent model

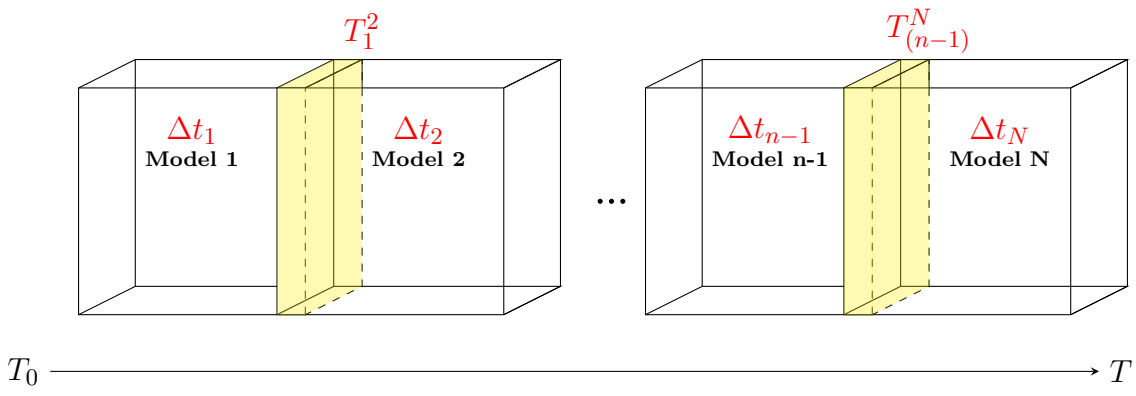


Figure 4.2 Schematics of the model discretization

is written as

$$L_{\mathrm{IC}}(\boldsymbol{X}) = \frac{1}{N_{\mathrm{I}}} \sum_{n=1}^{N_{\mathrm{I}}} \left(\|\boldsymbol{u}_{\theta_i}^n - \boldsymbol{u}_{\theta_{i-1}}^n\|_{L^2}^2 + \|\nabla(\boldsymbol{u}_{\theta_i}^n - \boldsymbol{u}_{\theta_{i-1}}^n)\|_{L^2}^2 \right) \tag{4.24}$$

which is equivalent to:

$$L_{\mathrm{IC}}(\boldsymbol{X}) = \frac{1}{N_{\mathrm{I}}} \sum_{n=1}^{N_{\mathrm{I}}} \|\boldsymbol{u}_{\theta_i}^n - \boldsymbol{u}_{\theta_{i-1}}^n\|_{H^1}^2 \tag{4.25}$$

$$\boldsymbol{X} = (x_0, y_0, z_0, t_{int}) \tag{4.26}$$

where the subscript 0 denotes the correspondence to the IC, t_{int} represents the time step corresponding to the overlapping zone, $\boldsymbol{u}_{\theta_i}^n$ is the approximation to the IC in the receiving subdomain, and $\boldsymbol{u}_{\theta_{i-1}}^n$ is the predicted solution at the overlapping zone (to serve as training data), generated by the transmitter subdomain

$$\boldsymbol{u}_{\theta_{i-1}} = f_{NN}^{i-1}(\boldsymbol{X}, \boldsymbol{\theta}_{i-1}). \tag{4.27}$$

The process involves sequentially transferring approximations to be used as IC from the first to the last subdomains. This transfer mechanism ensures a systematic propagation of IC throughout all the subdomains. The schematics of the process is illustrated in Figure 4.1. In this figure, the generative mode indicates that the subdomain has reached the accuracy

threshold, signaling the opening of the gating mechanism. This allows for the transfer of training data to the subsequent subdomain while also continuing simultaneous training.

The term "Generative" is used in this work to describe the iterative process by which the model generates new approximations at temporal points or coordinates where no training data were originally provided. Unlike traditional supervised learning, which relies solely on labeled data, this approach enables the model to autonomously create approximations in previously unexplored regions of the temporal domain. These newly generated approximations, produced in an unsupervised manner, are then utilized to inform the training process in subsequent networks and subdomains. The gating mechanism plays a critical role in this process by determining the accuracy of the generated information. It does so by evaluating the error of the approximation within the transmitter model. Based on this error, the gating mechanism decides whether the generated data is sufficiently accurate to be transferred to subsequent subdomains. This iterative generation and validation of new approximations allow the model to continuously refine its solution over time, even in areas where initial data are sparse or unavailable. It is important to note that the newly generated and transferred information is treated as additional training data that was not originally available. At each time step, except for the very first one, the training data originally consists only of boundary conditions (supervised learning) and the residuals of the governing equations (unsupervised learning). The newly generated data, which the model creates as part of the iterative process, is incorporated as supplementary information to refine and enhance the training process at subsequent time steps.

The availability of sufficient boundary conditions for each subdomain is essential to avoid an under-determined system. When initial conditions are only provided to the first subdomain, a lack of adequate BC in subsequent subdomains can leave the system under-determined, thereby hindering the training process. In an extreme case, one might encounter a system where no BC or IC are available, relying solely on the minimization of the residuals of the governing equations. To mitigate this issue, a gating mechanism is typically employed to transfer additional information between subdomains. If adequate BC are unavailable, it may be necessary to relax the gating mechanism's transfer threshold, even if the transferred information is not perfectly accurate. This strategy ensures that subsequent subdomains do not remain under-determined, preserving the effectiveness of the training process across the domain. The optimal value of the transfer threshold, however, remains a hyperparameter that requires manual adjustment.

Clarification on the Proposed Method vs. Standard Transfer Learning:

While our method shares some conceptual similarities with transfer learning, it fundamentally differs in both methodology and application:

Decomposition Approach: We decompose the entire temporal domain into multiple overlapping subdomains, each handled by a separate neural network model with its own model parameters. This allows each model to specialize in a specific region, capturing local dynamics more effectively.

Parallel Training: These subdomain models are trained in parallel, not sequentially.

Overlapping Zones: The overlapping regions between subdomains facilitate the transfer of information and ensure continuity and consistency across the entire domain. These overlaps allow models to share approximations in the overlapping zones, stitching together a coherent global solution.

Generative Mode: Our models generate new approximations at temporal points where no training data were originally provided. This is crucial for time-dependent problems with sparse initial data, as it allows the models to autonomously create necessary solution approximations based on the governing equations.

Iterative Refinement: The process allows for continuous refinement of the solution over time. By generating and validating new approximations, the models iteratively improve the solution, even in regions lacking initial data.

Use of the H^1 Norm of Error: We utilize the H^1 norm of the error when transferring approximations between subdomains. By incorporating both the function values and their gradients, the H^1 norm ensures that the transferred approximations maintain consistency with the governing PDEs and adhere to the underlying physics.

These features distinguish our approach from standard transfer learning, which typically involves sequentially training a single model on a source task and then fine-tuning it on a target task.

Terminology Clarification:

While we refer to this mechanism as "generative model", we acknowledge that it differs from the concept of "generative learning". Traditional generative learning models aim to model data distributions and generate new samples from these distributions. In our approach, the term "generative" is used to describe the process by which the model creates new approximations at temporal points or regions where no original training data is available. These

approximations are deterministic outputs derived from solving the governing equations and are not probabilistically generated. The mechanism is "generative" in the sense that it autonomously propagates solutions to unexplored regions of the domain, enabling systematic information transfer and continuity across subdomains.

Analysis of the Solution Regularity

Using the H^1 function space instead of the L^2 in the loss function promotes solutions with higher regularity. This is because the H^1 norm penalizes not only the differences in function values but also the differences in their gradients, which encourages smoother solutions. It is important to note that this regularity assumption holds only when the approximations are sufficiently smooth, i.e., at least C^1 continuous. In cases where the functions are only C^0 continuous, the regularity argument breaks down since gradients may not be well-defined. Additionally, the regularity of the approximation is influenced by the activation function used in the network, as certain activation functions may limit the smoothness of the solutions. Let us formalize this using mathematical formulations and proofs:

H^1 **Norm**: The H^1 norm of a function $f(x)$ over a domain Ω is defined as:

$$\|f\|^2_{H^1(\Omega)} = \int_\Omega \left(|f(x)|^2 + |\nabla f(x)|^2\right) dx$$

where $|\nabla f(x)|$ denotes the gradient of $f(x)$ and $|\cdot|$ denotes the L^2 norm.

Regularizing Effect: Adding the term $\|\nabla f(x)\|^2$ in the H^1 norm penalizes large gradients in the function $f(x)$. Minimizing this term encourages smoother solutions because abrupt changes or sharp variations in the function are penalized. However, the regularizing effect also depends on the smoothness of the approximation, which is influenced by the choice of activation function. For instance, ReLU activations only guarantee C^0 continuity, limiting the smoothness of the approximation. On the other hand, the tanh activation function, which is a smooth, C^∞ continuous function, naturally provides a higher level of smoothness. By using tanh, the network approximations will exhibit smoother transitions, which aligns better with the assumptions of the H^1 norm. The proof of the regularizing effect in the model approximation is given in the appendix 4.5.

Enhancement of the PINNs solution process using the H^1 norm of errors: Using the H^1 norm in the mean squared error of the loss function introduces additional constraints that further limit the space of admissible solutions. By considering both the function values and their gradients, the H^1 norm effectively narrows the solution space to functions that satisfy not only the governing equations but also possess the desired smoothness properties. This

added constraint enhances the approximation by steering the optimization toward solutions that are physically and mathematically more consistent with the problem's nature.

- Alignment with Variational Principles: Many PDEs arise from variational formulations where the solution minimizes an energy functional in an H^1 space. By incorporating the H^1 norm into the loss function, we ensure that the PINN model adheres to the same minimization principles as the true solution. This alignment can potentially improve convergence to the true solution and enhance the accuracy of the approximation.

- Improved Optimization Landscape: Including derivative terms in the loss function can smooth the optimization landscape, making it easier for gradient-based optimization algorithms to find the global minimum. The presence of gradient information reduces the likelihood of encountering sharp minima or saddle points, leading to more stable convergence and reduced sensitivity to initialization and hyperparameters.

- Complementary Role to Tolerance and Method Order: While tolerance and the order of the numerical method are crucial factors dictating accuracy, the norm used in the loss function directly impacts how errors are measured and minimized during training. The H^1 norm provides a more comprehensive measure of error by accounting for both the solution and its derivatives. This can lead to more accurate and stable solutions, complementing the effects of tolerance and method order. By influencing the optimization process at each iteration, the H^1 norm guides the network toward better approximations that satisfy both the PDE and the desired regularity.

Gating Mechanism

Since subdomain models are trained in parallel and separately, dynamic information propagation may sometimes slow convergence. Transferred data in the receiver model only helps when the transmitter's approximation loss is below a certain level. To enforce this, we introduce a gating mechanism in the DG-PINN: data transmission between subdomain models is allowed only when the transmitter subdomain's loss falls beneath a predefined threshold $L_{Omega(\tau)}$ (see the decision point in Figure 4.3). Let

$$L_\Omega(\mathcal{T})$$

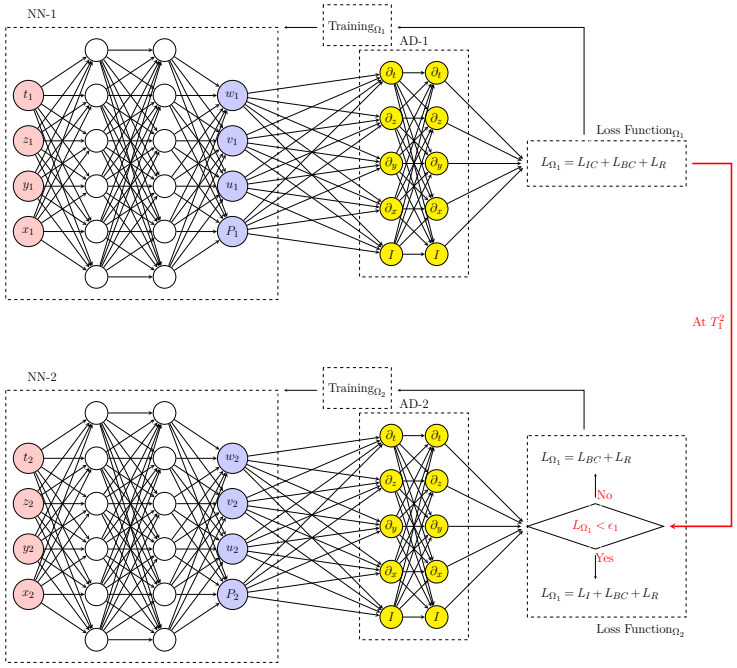


Figure 4.3 Schematics of the DG-PINN

denote the approximation loss in the transmitter model $\mathcal{T}$. We define the binary gate

$$\gamma = \begin{cases} 1, & \text{if } L_\Omega(\mathcal{T}) \leq \epsilon, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, to limit memory overhead, this check can be performed only at fixed intervals (e.g., every 50 iterations).

Training

For the training of models in the DG-PINN, a distributed mini-batch gradient descent approach was employed. In this setup, both the training data and the model are discretized and processed separately and in parallel (refer to Figure 4.3). Consequently, the training occurs independently for each subdomain, executed in parallel. The parameters of each model are updated utilizing the mini-batch gradient descent method. Considering different learning rates $\alpha^{(i)}$ for each parallel model $i = 1, 2, \ldots, N$, and using the loss function denoted by $L^{(i)}(\theta^{(i)})$, the update rule for each model's parameters $\theta^{(i)}$ in mini-batch gradient descent can be expressed as follows, such that the gradient calculation for i^{th} model using a mini-batch

of size m is

$$\nabla L^{(i)}(\theta^{(i)}) = \frac{1}{m}\sum_{j=1}^{m} \nabla L_j^{(i)}(\theta^{(i)}) \tag{4.28}$$

where $\nabla L_j^{(i)}(\theta^{(i)})$ is the gradient contribution from the j^{th} data point in the mini-batch for the i^{th} model. The update rule for i^{th} model's parameter $\theta^{(i)}$ using the computed gradient and the corresponding learning rate is written as

$$\theta^{(i+1)} := \theta^{(i)} - \alpha^{(i)} \nabla L^{(i)}(\theta^{(i)}) \tag{4.29}$$

$$\theta^{(i+1)} := \theta^{(i)} - \frac{\alpha^{(i)}}{m}\sum_{j=1}^{m} \nabla L_j^{(i)}(\theta^{(i)}). \tag{4.30}$$

Accordingly, each parallel model i utilizes distinct learning rate $\alpha^{(i)}$ and updates parameters $\theta^{(i)}$ independently based on the gradients computed from its mini-batches.

The inputs of the NN are normalized to fall within the range of $[-1, 1]$ for each dimension. Similarly, the governing equations linked to these inputs are normalized using the same scaling factor. Consider three sets of data, including IC_i, BC_i, and R_i, where the subscript i denotes the i^{th} data set related to the IC, BC, and the residuals of governing equations in the i^{th} model

$$X_{\text{IC}_i} = \{(x_{0i}, y_{0i}, z_{0i}, t_{0i}, u_{0i}, v_{0i}, w_{0i})\} \tag{4.31}$$

$$X_{\text{BC}_i} = \{(x_{bi}, y_{bi}, z_{bi}, t_{bi}, u_{bi}, v_{bi}, w_{bi})\} \tag{4.32}$$

$$X_{\text{R}_i} = \{(x_i, y_i, z_i, t_i)\}. \tag{4.33}$$

Splitting each data set i into mini-batches, N/m min-batches are created, each containing m samples. The k^{th} mini-batch, denoted as B_k, consists of data indexed from $(k-1)\times m+1$ to $k \times m$. This can be mathematically represented as

$$B_{\text{IC}_i}^{k} = \Big\{(x_{0i}, y_{0i}, z_{0i}, t_{0i}, u_{0i}, v_{0i}, w_{0i})\Big\}_{(k-1)\times m+1}^{k\times m} \tag{4.34}$$

$$B^k_{\mathrm{BC}_i} = \left\{ (x_{bi}, y_{bi}, z_{bi}, t_{bi}, u_{bi}, v_{bi}, w_{bi}) \right\}^{k\times m}_{(k-1)\times m+1} \tag{4.35}$$

$$B^k_{\mathrm{R}_i} = \left\{ (x_i, y_i, z_i, t_i) \right\}^{k\times m}_{(k-1)\times m+1} \tag{4.36}$$

Permuting the data within mini-batches involves shuffling the indices within each mini-batch to randomize the order in which the data samples are fed into the learning algorithm. Given a mini-batch k with indices initially ordered as $[1, 2, ..., m]$, here is a mathematical representation for applying permutations within a mini-batch; let permutation (k) denote the permutation applied to the indices of mini-batch k. After applying a permutation to the k^{th} mini-batch, the new order of indices becomes:

$$\text{Permutation}(k) = [\text{perm}(1), \text{perm}(2), ..., \text{perm}(m)] \tag{4.37}$$

where $perm(i)$ represents the i^{th} element after permutation within mini-batch k. This operation randomizes the order of data samples within each mini-batch, contributing to a more effective training process in machine learning algorithms, especially during mini-batch gradient descent.

The DG-PINN is illustrated through Algorithm 1. N and e denote the number of subdomains and iterations, respectively. Additionally, α_i is the learning rate for each model and ϵ represents the predefined accuracy threshold for the gating mechanism. The training is based on the distributed mini-batch gradient descent approach, where various network parameters and learning rates are assigned to subdomains and models are trained in parallel. Notably, to mitigate excessive memory usage, the gating mechanism is activated at intervals of every 50 iteration, a configuration fine-tuned based on empirical observations gleaned from experimental analysis. The algorithm above illustrates only the general workflow of the proposed DG-PINN method. For clarity, it uses a basic gradient descent update rule (while in the code implementation we have used Adam) and deliberately omits the permutation and batching steps; these are handled in the full implementation.

Algorithm 1: Algorithm of the DG-PINN

Given discretization of the temporal domain into N subdomains, where each subdomain has a separate model with trainable parameter θ_i, loss function $L(\theta_i)$, parameters e, s and N, and collocation sets

$$X_{\mathrm{IC}_i} = \{(x_{0i}, y_{0i}, z_{0i}, t_{0i}, u_{0i}, v_{0i}, w_{0i})\}, \quad X_{\mathrm{BC}_i} = \{(x_{bi}, y_{bi}, z_{bi}, t_{bi}, u_{bi}, v_{bi}, w_{bi})\}, \quad X_{\mathrm{R}_i} = \{(x_i, y_i, z_i, t_i)\}.$$

Initialization: Set each θ_i using [116];
for $n = 1$ **to** e **do**
  for $i = 1$ **to** N **do**

$$L(\theta_i) = L_{\mathrm{IC}_i}\left(X_{\mathrm{IC}_i}\right) + L_{\mathrm{BC}_i}\left(X_{\mathrm{BC}_i}\right) + L_{\mathrm{R}_i}\left(X_{\mathrm{R}_i}\right).$$

$$\theta_i \leftarrow \theta_i - \alpha_i \nabla_{\theta_i} L(\theta_i).$$

  end
end

4.3 Results and Discussion

In this section, the objective is to evaluate how our proposed method, the DG-PINN, performs against the baseline PINNs and against traditional domain decomposition PINNs. The tangent hyperbolic (Tanh) function serves as the activation function, and the Glorot normal initialization method [116] is utilized for initializing all networks. As outlined in Algorithm 1, all networks are trained using a mini-batch gradient descent Adam optimizer [70], using piece-wise decay of learning rates for each model. The temporal domain $[0, T]$ is discretized into subdomains $[0 : t_1], [t_1 : t_2], ...[t_{n-2} : t_{n-1}], [t_{n-1} : T]$. Controlling over the proper transfer of generated data from the previous subdomain to serve as the initial condition in the next subdomain, during the overlapping time instances $t_1, t_2, ..., t_{n-1}$, is managed by a gating mechanism. Implementation of automatic differentiation and construction of computational graphs are facilitated using TensorFlow [117].

The validation is performed using multiple cross-validations utilizing randomly generated and entirely unseen evaluation points distinct from those employed during training. This process is followed by averaging the results. The number of test points corresponds to 25% of the total number of training points. In this section, the term PINNs refers to the baseline PINNs model, while DD-PINN represents a domain decomposition PINNs without overlapping constraints, and DG-PINN denotes our proposed method. Additionally, comparisons will be made between the DG-PINN and a counterpart of the DG-PINN model but using the L^2 norm of errors, to highlight the contribution os using H^1 norm of errors when propagating information between models. It is important to note that, for a fair comparison, the training time was kept the same across all competing methods in every comparison conducted in this study.

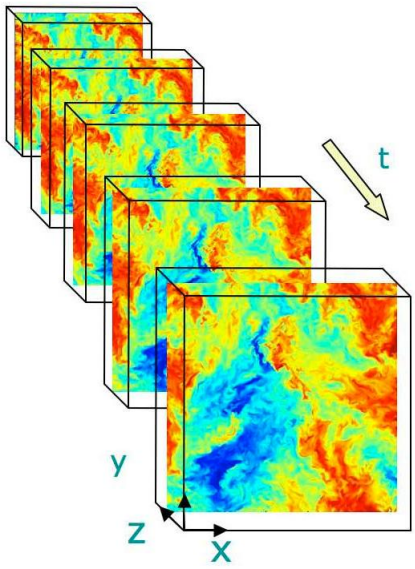


Figure 4.4 Schematics of the problem domain [1–3]

4.3.1 Problem Setup

The test problem involves a channel flow with $Re = 9.994 \times 10^2$. This Reynolds number is almost the critical Reynolds number for the transition from laminar to turbulent flow in channel flows. According to several authoritative sources, turbulence can occur at this Reynolds number in channel flows [142, 143]. We acknowledge that $Re \approx 1000$ is at the lower limit for turbulence in channel flows, but it is within the range where turbulence is expected to develop. Training and testing data for this study originate from the turbulent channel flow database from the Johns Hopkins Turbulence Database (JHTDB) available at https://turbulence.pha.jhu.edu/. These datasets are widely used in turbulence research and demonstrate turbulent characteristics at this Reynolds number. The spatial domain is $[0, 8\pi] \times [-1, 1] \times [0, 3\pi]$ and the time step in the online database is 0.0065. The viscosity is $\nu = 5 \times 10^{-5}$. The initial condition of the problem signifies a state of fully developed turbulence, as supported by the turbulence statistics provided in the reference datasets [3, 144]. We note that detailed turbulence statistics from the reference data demonstrate the turbulent nature of the flow at this Reynolds number. These include:

- Mean Velocity Profiles: Exhibiting a clear logarithmic layer near the wall, which is characteristic of turbulent boundary layers in wall-bounded flows. The velocity profiles align with the classical law of the wall [144]:

$$U^+ = \frac{1}{k} ln y^+ + B, \tag{4.38}$$

where U^+ is the dimensionless velocity, y^+ is the dimensionless wall distance, k is the von Kármán constant, and B is an empirical constant.

- Reynolds Stress Profiles: Showing significant turbulent shear stresses and normal stresses, indicating turbulent momentum transfer by turbulent eddies [144]. The Reynolds shear stress $-\overline{u'v'}$ has substantial values across the channel height, with peaks away from the wall.

- Turbulent Kinetic Energy (TKE) Profiles: Displaying high levels of TKE near the wall, characteristic of wall-bounded turbulent flows [145]. The TKE decreases towards the channel centerline, reflecting the distribution of energetic turbulent motions.

- Energy Spectra: Demonstrating an inertial subrange with a $-5/3$ slope in the spectral energy density when plotted on log-log scales, consistent with Kolmogorov's theory of turbulence [146]. The spectra show energy distributed across a wide range of scales, from large to small eddies.

While these key turbulence statistics illustrate the turbulent nature of the flow at this Reynolds number, a detailed analysis is beyond the scope of this work. For in-depth discussions on turbulence characteristics, we refer interested readers to the cited references.

The architecture of all the networks in this study is based on the FFNN architecture. Network input and output are four-dimensional, where (x, y, z, t) serve as inputs and (u, v, w, P) are outputs. In case studies I and II, we have considered two different domains with different spatial and temporal sizes at various locations within the channel. It is important to note that the quantitative results presented in this study are the average outcome obtained from 10 consecutive tests. Schematics of the problem domain are shown in Figure 4.4. The data and the codes utilized in this manuscript are publicly accessible on GitHub at https://github.com/AmirhosseinnnKhademi/DG-PINN.git.

4.3.2 Case study I

To study the performance of the DG-PINN, a domain is considered with dimensions $[12.47, 12.66] \times [-0.90, -0.70] \times [4.61, 4.82]$ for x, y, and z respectively. The temporal domain consists of 110 time steps. According to the description of the database [147], there is no viscous sublayer within the boundaries of this simulation. There are 10,000 collocation points and 6,644 BC points within each time step. Within the first time step, there are 33,524 initial condition (IC) points. The adjustments for the piece-wise decay of learning rates, number of epochs,

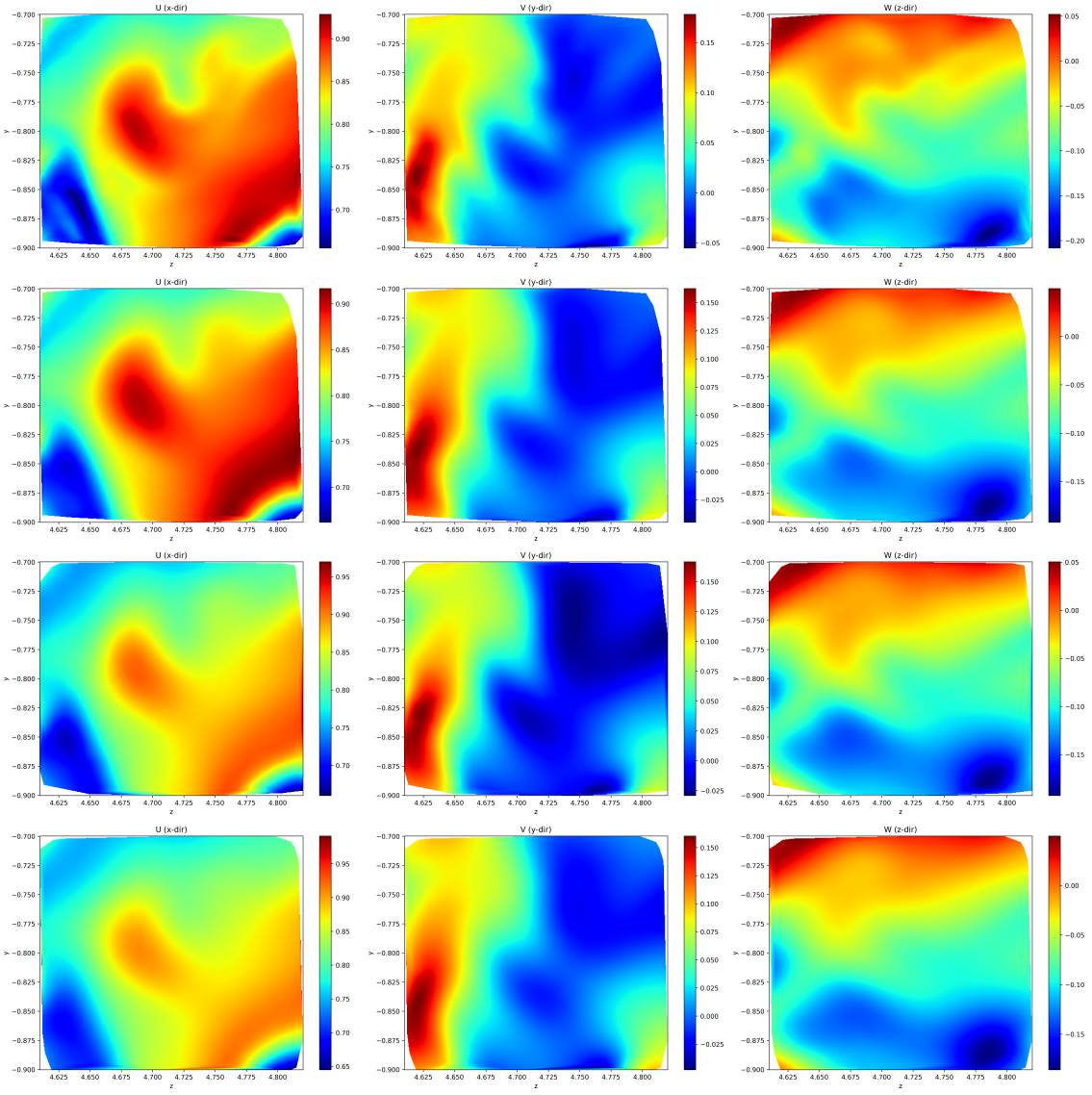


Figure 4.5 Instantaneous velocity fields at $t_8 = 8 \times 0.0065$: From top to bottom: DNS solution, DG-PINN, DD-PINN, and Baseline PINNs

Table 4.2 Settings of the training in case study I

# Epochs	# Iterations	Learning rate
300	150	1×10^{-3}
200	150	2×10^{-4}
200	150	4×10^{-5}
100	150	9×10^{-6}
50	150	9×10^{-7}

Table 4.3 Networks settings and memory usage in case study I

Method	Architecture	GPU use (Gb)
Baseline PINNs	$(1) \times 10 \times 300$	8.6
DD-PINN	$(10) \times 10 \times 150$	2.5
DG-PINN	$(10) \times 10 \times 150$	2.5

and number of iterations are specified in Table 4.2. The architectural configuration for networks in case study I is outlined in Table 4.3. The second column shows the architecture of the networks, where the number in parentheses represents the number of subdomains, followed by the number of hidden layers and the third is the number of neurons in each hidden layer. Additionally, memory usage is given for each model. The comparison between the DD-PINN and the DG-PINN is intended to highlight the significance of the generative model. This table highlights the efficacy of domain discretization, enabling the implementation of a distributed training strategy that results in more efficient memory usage. The instantaneous velocity fields in the $y-z$ plane at $x = 12.66$ approximated by the DG-PINN are compared with those predicted by the baseline PINNs and the reference DNS solution in Figures 4.5, 4.6, and 4.7 at time instances t_8, t_{65}, t_{95}, respectively. In this simulation, both the DG-PINN and the DD-PINN decompose the temporal domain into 10. As it is observable, the DG-PINN can simulate the 3D turbulent fluid flow with high accuracy and successfully capture the details. On the other hand, capturing general patterns while struggling with finer details was observed in the DD-PINN. Finally, the baseline PINNs could not converge to an accurate approximation within this limited number of epochs. Previous research by Jin et al. [85] indicated that significantly more epochs and iterations were required for the PINNs to converge in a similar problem.

To be more detailed, quantitative results comparing the errors among PINNs, DD-PINNs, and the DG-PINN are presented in Figure 4.8. While this figure confirms the findings of Figures 4.5 to 4.7, they also reveal an additional point; as time progressed away from the initial time step with well-defined IC, the accuracy of the prediction made by the baseline

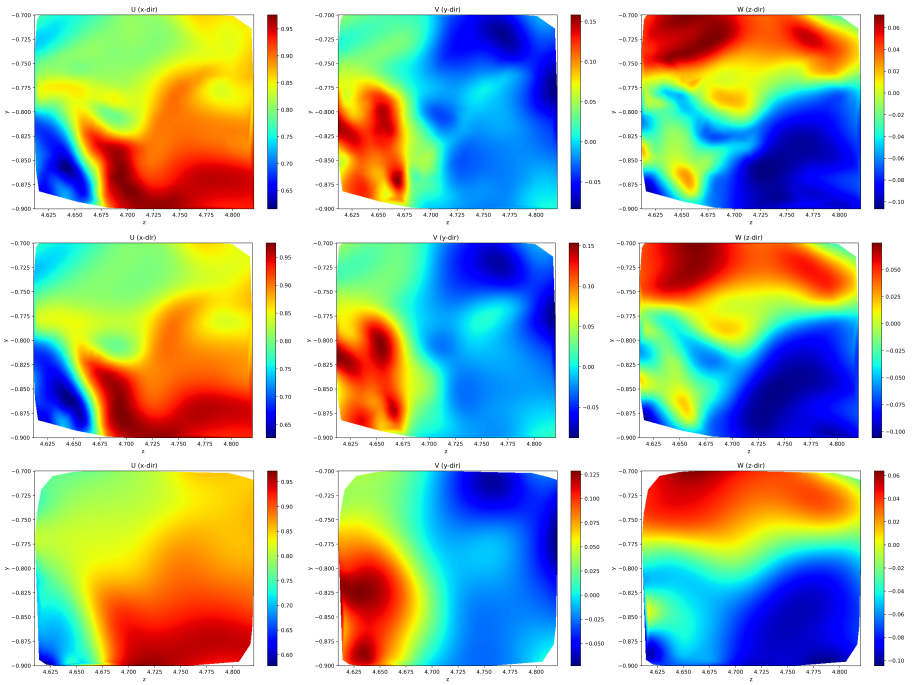


Figure 4.6 Instantaneous velocity fields at $t_{65} = 65 \times 0.0065$: From top to bottom: DNS solution, DG-PINN, and Baseline PINNs

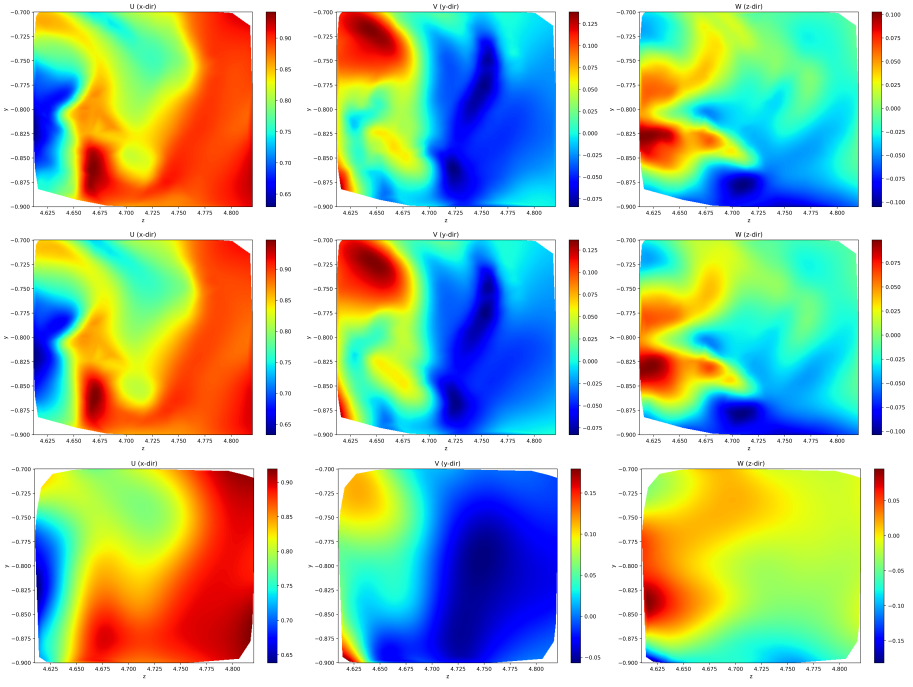


Figure 4.7 Instantaneous velocity fields at $t_{95} = 95 \times 0.0065$: From top to bottom: DNS solution, DG-PINN, and Baseline PINNs

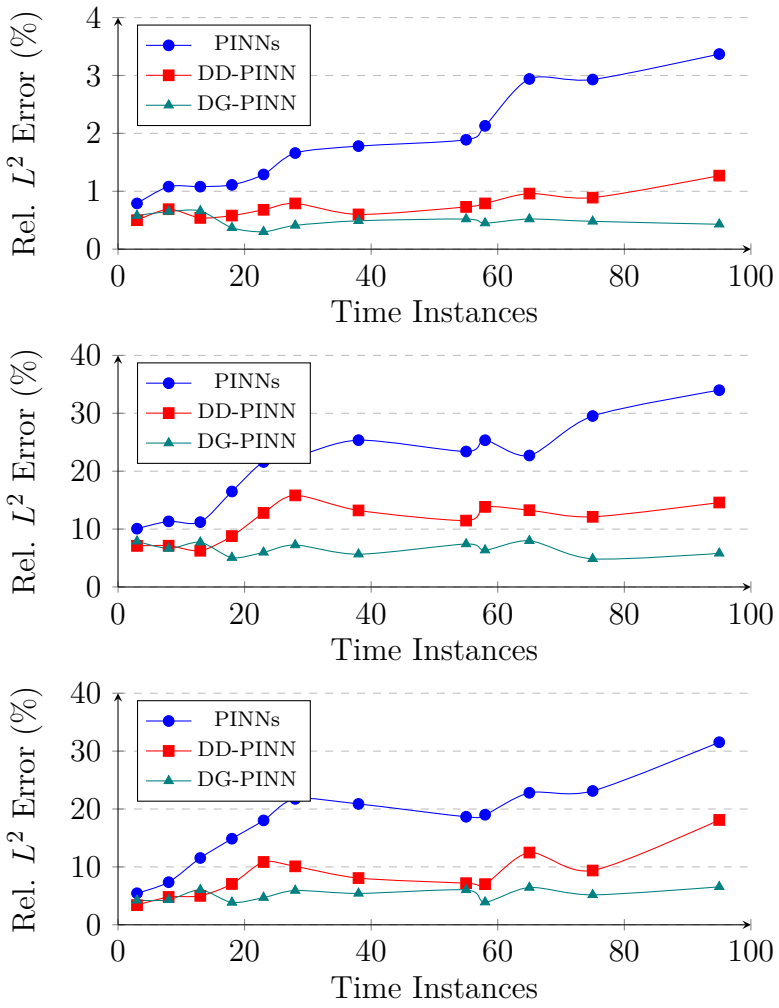


Figure 4.8 Comparison of the error: Top: x direction, middle: y direction, and bottom: z direction.

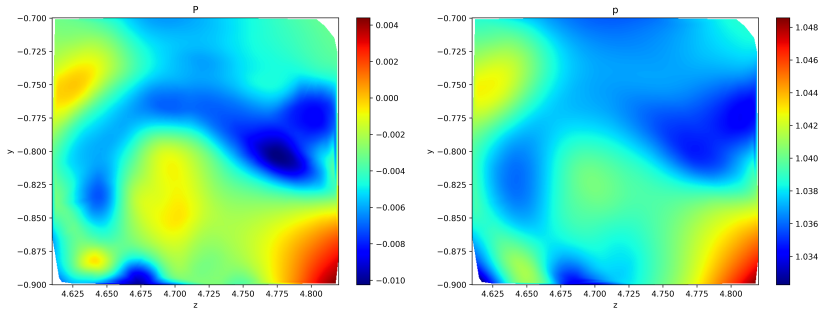


Figure 4.9 Pressure fields at $t_{15} = 15 \times 0.0065$. left: DNS solution; right: DG-PINN approximation

PINNs drastically decreases. Although the DD-PINN is able to predict more accurate approximations in the subsequent time steps, the accuracy is not comparable with the initial time steps due to the lack of IC. However, the DG-PINN is able to maintain the approximation accuracy at time instances where the model lacks IC, thanks to the discretized generative model. In particular, in the y and z directions where the PINNs' prediction error increases by approximately 25 percent, the DG-PINN significantly improves the accuracy at distant time steps.

The findings suggest that the wall-normal and spanwise velocities exhibit significantly greater relative L^2 errors compared to the streamwise velocity. This discrepancy arises due to the streamwise velocity's substantially larger amplitude in comparison to the other two velocity components. Lastly, in Figure 4.9, the approximation of instantaneous pressure fields by the DG-PINN is compared against the DNS solution. The results refer to the $y-z$ plane at $x = 12.55$ at time step $t_{15} = 15 \times 0.0065$. The DG-PINN demonstrates an acceptable agreement with the reference solution. It is intriguing to observe that, despite no explicit training data for pressure being provided to the model, it successfully learns and approximates the pressure field solely through unsupervised physics-informed learning. As illustrated by Raissi et al. [27], the observed difference in magnitude between the exact and predicted pressure is justified by the inherent characteristics of the incompressible Navier–Stokes system. In this system, the pressure field is identifiable only up to a constant. To underscore the significance of employing the H^1 function space, over the L^2 as used in previous discretized PINNs models, a comparison between the DG-PINN and its counterpart model utilizing the L^2 norm is shown in Figure 4.10. To simplify visualization and maintain clarity despite the inherent complexity of the 3D time-dependent problem, the plots in the figure depict solution approximations and their derivatives along the y direction. Six time steps are assigned to each subdomain and there is one overlapping time step between each pair of subdomains. Other settings and adjustments remain same as before. The subplots in the left column depict results for the model using the L^2 norm, while those in the right column represent the results for the DG-PINN utilizing the H^1 norm. In the left column of the figure, it is observed that in a limited number of training epochs, the discretized model utilizing L^2 norm exhibits differing gradient values between adjacent subdomains, suggesting the absence of a well-defined gradient for the global solution. Moreover, the model faces challenges in fully preserving continuity. Conversely, in the right column, representing the DG-PINN model employing the H^1 function space, despite the presence of sharp edges, the H^1 function space ensures consistent first derivatives. Solutions obtained through the DG-PINN model demonstrate continuity, as the approximations by different subdomains at the boundary remain consistent. This implies the existence of continuity and first derivatives in the global

solutions, a feature lacking in previous domain decomposition techniques in PINNs. For further insights into subsequent time steps and subdomains, additional plots, all confirming the conclusion here, are provided in Figure **??** and **??** in the appendix.

4.3.3 Case study II

To reflect the ability of the DG-PINN to simulate fluid flows with more complicated phenomena, a problem involving a larger domain, $[12.25, 12.75] \times [-1.0, -0.5] \times [4.5, 5.0]$, is considered. Within this larger domain, various interactions between different scales of eddies happen, which covers the range involving the viscous sub-layer, the log-law region, the law of the wall, the buffer layer, and the outer layer [3, 147]. The non-dimensional time domain is limited to 16 time steps, where each time interval is 0.0065. There are 50,000 collocation points inside the domain and 26,048 boundary points at each time step, with 147,968 points corresponding to the IC at the initial time step. Finally, there have been 1400 epochs in the training process, with the details presented in Table 4.4. In this simulation, the temporal domain is discretized into 4 subdomains. Each subdomain covers 4 consecutive time steps and shares one overlapping time step with the adjacent subdomain, resulting in a total of 16 non-repeated time steps. The computational graph adopts an FFNN architecture with 9 hidden layers, each containing 300 neurons. The input and output layers have dimensions of 4.

Table 4.4 Settings of the training in case study II

# Epochs	# Iterations	Learning rate
50	150	1×10^{-3}
500	150	5×10^{-4}
400	150	7×10^{-5}
400	150	1×10^{-5}
50	150	5×10^{-6}

Figure 4.11 compares the approximations of the instantaneous velocity and pressure fields by the DNS, and the DG-PINN at $t_{13} = 13 \times 0.0065$ in the $y-z$ plane at $x = 12.66$. As proposed by Figure 4.11, the DG-PINN demonstrates the capability to accurately approximate solutions for the turbulent 3D Navier-Stokes equations in a large domain, without employing any turbulence filter, directly approximating the Navier-Stokes equations. Table 4.5 shows the relative L^2 error in approximations by the DG-PINN in various sample time steps. Sustained accuracy of approximations over time, particularly in the absence of IC, even within an expanded domain, highlights the effectiveness of this novel technique. This is worth mentioning

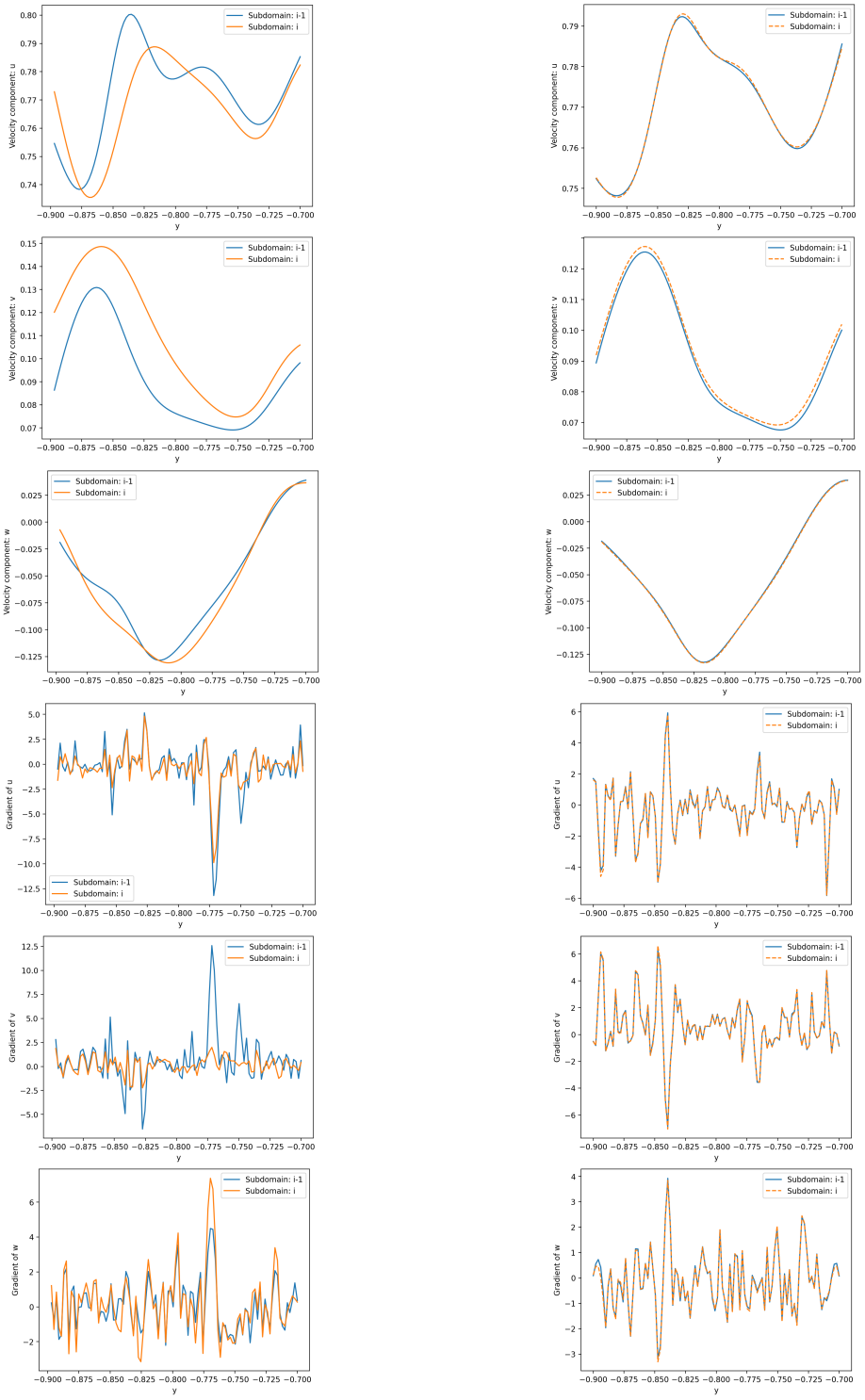


Figure 4.10 Solution regularity analysis at 5^{th} time step in an overlapping zone in y direction. Right column: DG-PINN approximations using H^1 norm; left column: a counterpart model using L^2 norm resembling traditional decomposition methods. From top to bottom: continuity of u; continuity of v; continuity of w; first derivative of u; first derivative of v; first derivative of w.

that the baseline PINNs fail to converge to an accurate approximation within 1400 training epochs for such a large domain with noticeably more training and collocation points.

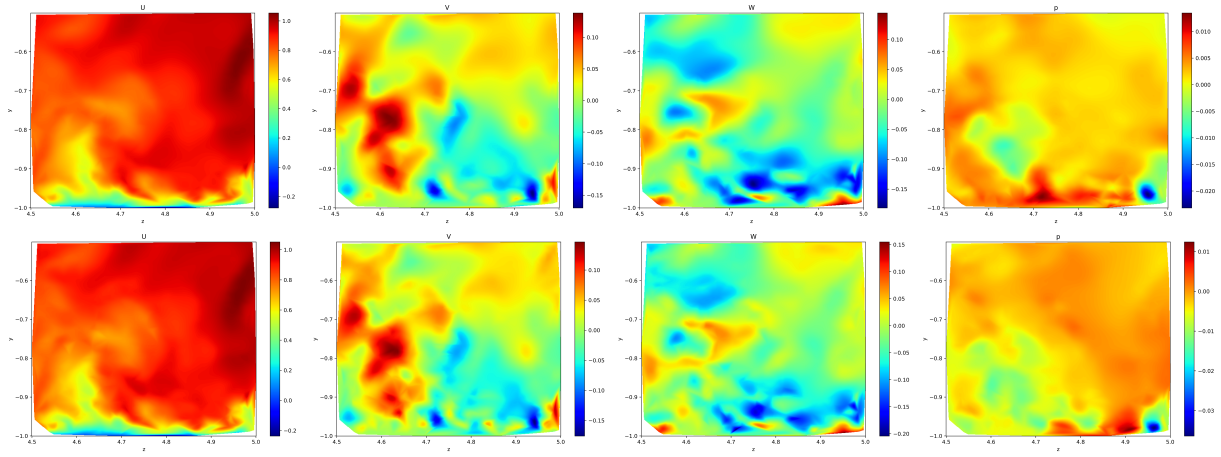


Figure 4.11 Velocity and pressure fields at $t_{13} = 13 \times 0.0065$. Top: DNS solutions; bottom: DG-PINN approximations

Table 4.5 DG-PINN velocity approximation accuracy in case study II

Time step	L^2 error x direction	L^2 error y direction	L^2 error z direction
0	1.4%	12.2%	10.1%
4	2.2%	16.3%	13.7%
8	1.7%	15.1%	12.4%
12	2.4%	17.9%	14.1%

4.4 Conclusion

This research introduced a novel methodology within Physics-Informed Neural Networks known as the discretized generative model PINNs. The DG-PINN is developed through discretization of the temporal domain, employing multiple models and generation of new training data, incorporating a gating mechanism for optimal regulation of data transfer among models, formulation of extra loss terms within H^1 function space, and using a distributed training strategy. This methodology was applied to a time-dependent 3D turbulent channel flow at $Re \approx 1000$, governed by the incompressible Navier-Stokes equations, and achieved good agreements with the reference solution. In the first case study, a comparison was made between the instantaneous velocity fields generated by the baseline PINNs and those of the DG-PINN over an extended period. The DG-PINN exhibits superior accuracy compared to the baseline PINNs while utilizing less memory thanks to a distributed training strategy

made possible by model discretization. The approximation error of the baseline PINNs increases significantly over time, particularly in subsequent time steps where initial conditions are not available. In contrast, the DG-PINN successfully maintains accuracy throughout. Furthermore, quantitative results demonstrated that employing H^1 function space enhanced the regularity of the global solutions in comparison to previous domain decomposition PINNs techniques utilizing L^2 function space. Interestingly, the DG-PINN accurately approximated pressure field patterns without utilizing any pressure training data within the complex 3D time-dependent turbulent flow regime. In the second case study, covering a more extensive spatial domain, which includes the viscous sub-layer, log-law region, law of the wall, buffer layer, and outer layer, the DG-PINN converges to a relatively accurate approximation, whereas the baseline PINNs fails to converge within the specified number of training epochs. The DG-PINN allows for the simultaneous use of multiple models, paving the way for future research to deploy our method with different turbulent models, thereby enhancing accuracy in broader domains characterized by diverse flow regimes.

4.5 Appendix

4.5.1 Methodology: Solution Regularity Analysis

Proof: Consider a sufficiently smooth function $f(x)$ and its approximate solution $g(x)$ obtained through optimization. We aim to show that minimizing the H^1 norm of $g(x)$ encourages smoothness. Specifically, this proof assumes that $f(x)$ and $g(x)$ are at least C^1 continuous.

Let $E(f,g) = \frac{1}{2}\|f-g\|^2_{H^1(\Omega)}$ denote the error between the true function $f(x)$ and the approximate solution $g(x)$ in the H^1 norm.

We can expand $E(f,g)$ as follows:

$$E(f,g) = \frac{1}{2}\left(\|f-g\|^2_{L^2} + \|\nabla(f-g)\|^2_{L^2}\right)$$

Now, let's analyze the second term $\|\nabla(f-g)\|^2_{L^2}$. By the triangle inequality, we have:

$$\|\nabla(f-g)\|^2_{L^2} \leq \|\nabla f\|^2_{L^2} + \|\nabla g\|^2_{L^2}$$

Since the true function $f(x)$ is assumed to be smooth (i.e., at least C^1 continuous), $\|\nabla f\|^2_{L^2}$ is bounded.

Minimizing $E(f,g)$ with respect to $g(x)$ entails minimizing both $\|f-g\|^2_{L^2}$ and $\|\nabla(f-g)\|^2_{L^2}$.

Minimizing the latter term $\|\nabla(f-g)\|_{L^2}^2$ effectively encourages $g(x)$ to be smooth by penalizing sharp variations. However, the regularity of the approximation $g(x)$ is affected by the activation function used in the neural network. For instance, ReLU-based networks, which produce piecewise linear approximations, may not fully benefit from the H^1 norm's regularizing effect due to their limited smoothness. In contrast, networks using smoother activation functions, such as tanh or softplus, produce much smoother approximations, better aligning with the assumptions of the H^1 norm. Therefore, using the H^1 norm instead of the L^2 in the loss function promotes solutions with higher regularity, provided that the approximations are sufficiently smooth and the choice of activation function supports this regularity.

CHAPTER 5 ARTICLE 3: PHYSICS-INFORMED NEURAL NETWORKS WITH TRAINABLE SINUSOIDAL ACTIVATION FUNCTIONS FOR APPROXIMATING THE SOLUTIONS OF THE NAVIER-STOKES EQUATIONS

Authors: Amirhossein Khademi and Steven Dufour

Journal of Computer Physics Communications (CPC), Elsevier, submitted on 2024-11-28, accepted on 2025-05-09, published on: 2025-05-15.

5.1 INTRODUCTION

Machine learning (ML) has shown promise for performing nonlinear mapping tasks through the use of surrogate models. But, ML-based 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. [15] introduced Physics-Informed Neural Networks (PINN), with subsequent variants further developed to applied to fluid mechanics problem [27, 28]. 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. This innovative approach eliminates the need for mesh generation and numerical discretization by using automatic differentiation [38]. 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. [39], Jagtap et al. [40] 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 [41], and finally led to the parallel PINN [42]. A time-sweeping and information propagation scheme was later incorporated in the scientific ML framework, to optimize training schedules on subdomains [43]. This approach led to computational speed-ups. Recently, Khademi et al. [128] 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. [50]. Lorin et al. [46] explored the capabilities of PINN framework to solve the time-dependent Dirac equation, showcasing its efficiency in quantum relativistic physics without 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 [47]. Zhang et al. [48] 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. [49] 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. [148] 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. [149] 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. [56] 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. [57] 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. [58] 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 [59]. Spline-PINNs [60]

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. Self-adaptive PINNs (SA-PINNs), as proposed by Mcclenny et al. [61], enhance training dynamics using the neural tangent kernel (NTK). This method adjusts network weights in response to increases in loss, effectively training the model to minimize losses, while maximizing weight 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 [63–66], 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. [50] introduced a method that adjusts learning rates dynamically, inspired by the work of Kingma et al. [70]. 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 [71] 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. [72] 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. [78] 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. [79] utilized parameterized Navier-Stokes equations as constraints.

Using RANS model with eddy viscousity, showed that missing data in randomly specified regions can be inferred. Similar approaches were used in other studies to approximate turbulent flows [80, 81]. Patel et al. [82] 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. [83] 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 et al. [84] 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. Finally, Jin et al. [85] used PINNs to directly numerically 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. 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.

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. [86]'s adaptive sigmoidal activation function for multilayer networks, Qian et al. [87]'s data-driven method of combining activation functions for convolutional neural networks, and Dushkoff et al. [88]'s technique allowing neurons to choose from multiple activation functions. Abbasi et al. [89] 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. [90] 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 [91] 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.

5.2 METHODOLOGY

5.2.1 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:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \mathcal{N}(\boldsymbol{u};\boldsymbol{x},t) = 0. \tag{5.1}$$

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

$$L_{\text{data}} = \sum_{i=1}^{N_{\text{data}}} \left\| \hat{\boldsymbol{u}}(\boldsymbol{x}_i,t_i) - \boldsymbol{u}(\boldsymbol{x}_i,t_i) \right\|^2, \tag{5.2}$$

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),

$$L_{\text{phys}} = \sum_{j=1}^{N_{\text{phys}}} \left\| \mathcal{N}(\hat{\boldsymbol{u}}(\boldsymbol{x}_j,t_j);\boldsymbol{x}_j,t_j) \right\|^2, \tag{5.3}$$

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

$$L_{\text{PINN}} = L_{\text{data}} + L_{\text{phys}}, \tag{5.4}$$

where we want the approximate solution $\hat{\boldsymbol{u}}(\boldsymbol{x},t)$ converge 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.

5.2.2 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 [21, 113, 114]. 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

$$f_i(\boldsymbol{x};\boldsymbol{w}_i,\boldsymbol{b}_i)=\alpha(\boldsymbol{w}_i\cdot\boldsymbol{x}+\boldsymbol{b}_i),\quad \text{for } i=1,2,\ldots,n, \tag{5.5}$$

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

$$\boldsymbol{U}\theta(\boldsymbol{X}),P\theta(\boldsymbol{X})=f_{\mathrm{NN}}(\boldsymbol{X},\theta), \tag{5.6}$$

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 5.1, 5.2, 5.3, and 5.4). The final FFNN representation is written as:

$$f_{\mathrm{NN}}(\boldsymbol{x},\theta)=(a_L\circ\alpha\circ a_{L-1}\circ\cdots\circ\alpha\circ a_1)(x), \tag{5.7}$$

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

5.2.3 Trainable Sinusoidal Activation of Physics-Informed Neural Networks

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

$$a_i=z_i=w_ia_{i-1}+b_i, \tag{5.8}$$

starting with the input layer

$$a_1=w_1x+b_1, \tag{5.9}$$

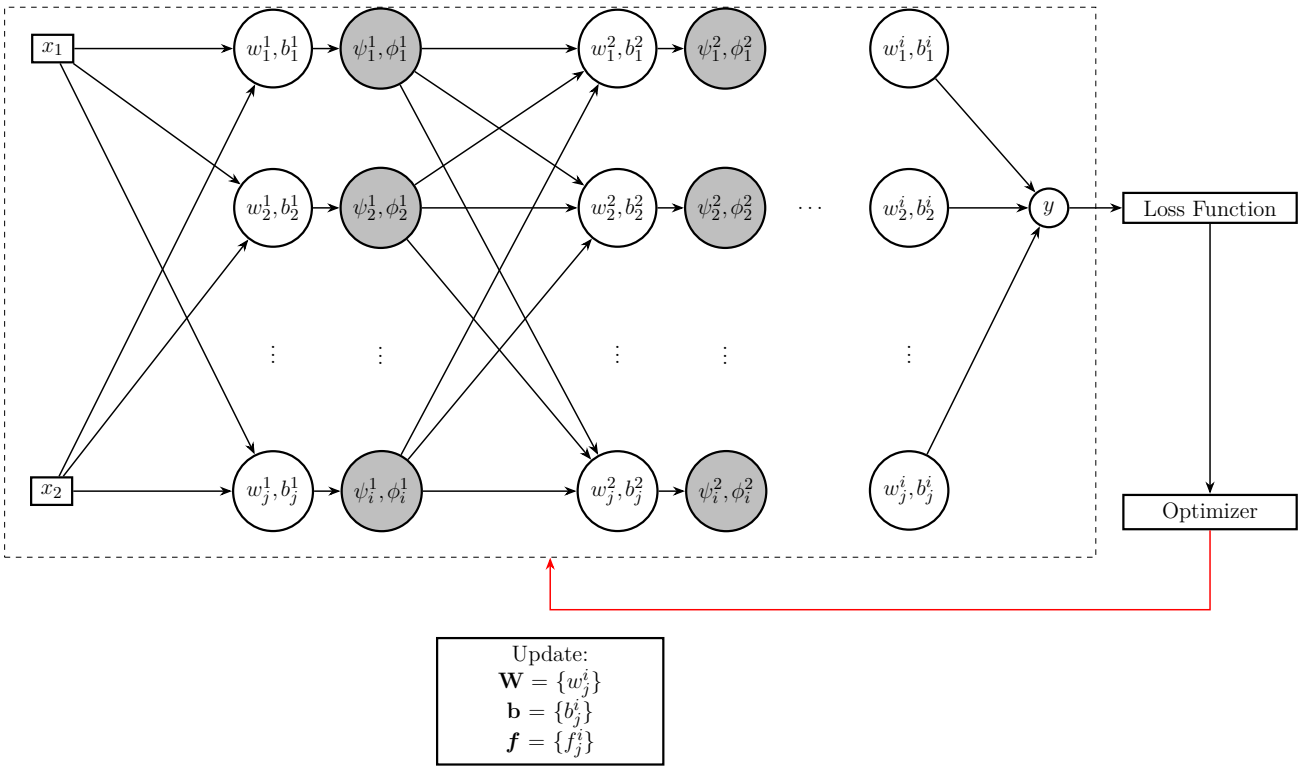


Figure 5.1 Schematics of the TSA-PINN. The white circles represent the multiplication of weights and the addition of biases. The gray circles indicate the application of activation functions.

the second layer

$$\begin{aligned} a_2 = w_2 a_2 + b_2 = w_2(w_1 x + b_1) + b_2 \\ = w_1 w_2 x + w_2 b_1 + b_2 \end{aligned} \tag{5.10}$$

and the third layer

$$\begin{aligned} a_3 = w_3 a_2 + b_3 = w_3(w_2 w_1 x + w_2 b_1 + b_2) + b_3 \\ = w_3 w_2 w_1 x + w_3 w_2 b_1 + w_3 b_2 + b_3. \end{aligned} \tag{5.11}$$

This process continues until the final layer L is reached

$$\begin{aligned} a_L = w_L w_{L-1} \dots w_2 w_1 x + (w_L w_{L-1} \dots w_2 b_1 + w_L w_{L-1} \\ \dots w_3 b_2 + \dots + b_L). \end{aligned} \tag{5.12}$$

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

$$\boldsymbol{y} = \boldsymbol{w}_{\text{total}} x + \boldsymbol{b}_{\text{total}} \tag{5.13}$$

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 (5.13). For a standard fully connected neural network (FFNN), the pre-activation output of neuron i in layer k is given by:

$$z_i^{(k)} = \mathbf{w}_i^{(k)} \cdot \mathbf{a}^{(k-1)} + b_i^{(k)}, \tag{5.14}$$

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:

$$\psi_i^{(k)} = \sin(f_i^{(k)} z_i^{(k)}), \tag{5.15}$$

$$\phi_i^{(k)} = \cos(f_i^{(k)} z_i^{(k)}), \tag{5.16}$$

$$a_i^{(k)} = \zeta_1 \psi_i^{(k)} + \zeta_2 \phi_i^{(k)} = \zeta_1 \sin(f_i^{(k)} z_i^{(k)}) + \zeta_2 \cos(f_i^{(k)} z_i^{(k)}), \tag{5.17}$$

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:

$$\mathbf{z}^{(k)} = \mathbf{W}^{(k)} \mathbf{a}^{(k-1)} + \mathbf{b}^{(k)}, \tag{5.18}$$

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 $\mathbf{f}^{(k)} \in \mathbb{R}^{n_k}$:

$$\boldsymbol{\psi}^{(k)} = \sin(\mathbf{f}^{(k)} \odot \mathbf{z}^{(k)}), \tag{5.19}$$

$$\boldsymbol{\phi}^{(k)} = \cos(\mathbf{f}^{(k)} \odot \mathbf{z}^{(k)}), \tag{5.20}$$

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

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

$$\mathbf{a}^{(k)} = \begin{bmatrix} a_1^{(k)} \\ a_2^{(k)} \\ \vdots \\ a_N^{(k)} \end{bmatrix} = \begin{bmatrix} \zeta_1 \sin(f_1^{(k)} z_1^{(k)}) + \zeta_2 \cos(f_1^{(k)} z_1^{(k)}) \\ \zeta_1 \sin(f_2^{(k)} z_2^{(k)}) + \zeta_2 \cos(f_2^{(k)} z_2^{(k)}) \\ \vdots \\ \zeta_1 \sin(f_N^{(k)} z_N^{(k)}) + \zeta_2 \cos(f_N^{(k)} z_N^{(k)}) \end{bmatrix}. \tag{5.21}$$

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.

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. [91], the network is forced to rapidly enhance the activation slope, consequently accelerating the training process. We have

$$S(a) = \frac{1}{\frac{1}{L-1}\sum_{k=1}^{L-1} \exp\left(\frac{1}{N_k}\sum_{i=1}^{N_k} f_i^k\right)}, \tag{5.22}$$

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

$$L_{\text{PINN}} = L_{\text{data}} + L_{\text{phys}} + \lambda S(a), \tag{5.23}$$

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.

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:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p - \frac{1}{Re}\Delta\boldsymbol{u} = 0 \qquad \text{in } \Omega\times(0,T], \tag{5.24}$$

$$\nabla\cdot\boldsymbol{u} = 0 \qquad \text{in } \Omega\times(0,T], \tag{5.25}$$

$$\boldsymbol{u} = u_\Gamma \qquad \text{on } \Gamma_D\times(0,T], \tag{5.26}$$

$$\frac{\partial \boldsymbol{u}}{\partial n} = 0 \qquad \text{on } \Gamma_N\times(0,T]. \tag{5.27}$$

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 (5.26) and (5.27), respectively. The residuals associated with the conservation of the momentum and continuity equations, (5.24) and (5.25) respectively, can be expressed as:

$$\begin{aligned} R_x = \partial_t u + u\partial_x u + v\partial_y u + w\partial_z u + \partial_x p \\ -\frac{1}{Re}(\partial^2_{xx} u + \partial^2_{yy} u + \partial^2_{zz} u); \end{aligned} \tag{5.28}$$

$$\begin{aligned} R_y = \partial_t v + u\partial_x v + v\partial_y v + w\partial_z v + \partial_y p \\ -\frac{1}{Re}(\partial^2_{xx} v + \partial^2_{yy} v + \partial^2_{zz} v); \end{aligned} \tag{5.29}$$

$$\begin{aligned} R_z = \partial_t w + u\partial_x w + v\partial_y w + w\partial_z w + \partial_z p \\ -\frac{1}{Re}(\partial^2_{xx} w + \partial^2_{yy} w + \partial^2_{zz} w); \end{aligned} \tag{5.30}$$

$$R_c = \partial_x u + \partial_y v + \partial_z w. \tag{5.31}$$

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 [38]. 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:

$$L = L_{\mathrm{IC}} + L_{\mathrm{BC}} + L_{\mathrm{R}} + \lambda L_{\mathrm{S}}, \tag{5.32}$$

and the loss terms are written as:

$$L_{\mathrm{IC}} = \frac{1}{N_{\mathrm{I}}} \sum_{n=1}^{N_{\mathrm{I}}} \left| \boldsymbol{u}^n_\theta - \boldsymbol{u}^n_{\mathrm{IC}} \right|^2; \tag{5.33}$$

$$L_{\mathrm{BC}} = \frac{1}{N_{\mathrm{B}}} \sum_{n=1}^{N_{\mathrm{B}}} \left| \boldsymbol{u}_{\theta}^{n} - \boldsymbol{u}_{\mathrm{BC}}^{n} \right|^{2}; \tag{5.34}$$

$$L_{\mathrm{R}} = \frac{1}{N_{\mathrm{R}}} \left(\sum_{n=1}^{N_{\mathrm{R}}} \left| R_{x}^{n} \right|^{2} + \sum_{n=1}^{N_{\mathrm{R}}} \left| R_{y}^{n} \right|^{2} + \sum_{n=1}^{N_{\mathrm{R}}} \left| R_{z}^{n} \right|^{2} + \sum_{n=1}^{N_{\mathrm{R}}} \left| R_{c}^{n} \right|^{2} \right), \tag{5.35}$$

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 (5.32) represents the slope recovery term, as introduced in equation (5.22). The optimization problem discovers the optimal values of network parameters such that to minimize the loss associated with the approximation:

$$\boldsymbol{W}^{*} = \arg\,\min_{w}(L(w)); \tag{5.36}$$

$$\boldsymbol{b}^{*} = \arg\,\min_{b}(L(b)); \tag{5.37}$$

$$\boldsymbol{f}^{*} = \arg\,\min_{f}(L(f)). \tag{5.38}$$

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

$$w^{m+1} = w^{m} - \eta \nabla_{w^{m}} L^{m}(w); \tag{5.39}$$

$$b^{m+1} = b^{m} - \eta \nabla_{b^{m}} L^{m}(b); \tag{5.40}$$

$$f^{m+1} = f^{m} - \eta \nabla_{f^{m}} L^{m}(f), \tag{5.41}$$

where, at the m-th iteration, η denotes the learning rate and L^{m} is the loss function. Please note that equations 5.39 to 5.41 illustrate a generic gradient-descent update; the Adam optimizer used in our experiments employs a different, moment-based update scheme.

5.3 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

$$\text{Error}_i = \frac{\|\hat{\boldsymbol{U}}_i - \boldsymbol{U}_i\|_2}{\|\boldsymbol{U}_i\|_2} \times 100, \tag{5.42}$$

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 [116]. Trainable frequencies are initialized using $\sigma = 1.0$ unless specified otherwise. A gradient descent method is used in all cases, with the ADAM optimizer [70]. TensorFlow is used for automatic differentiation and computational graph construction [117].

5.3.1 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 (5.24) and (5.25). For this problem, we set the Reynolds number to $Re = 100$, and the system is expected to converge to a steady-state solution [112]. 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

$$u_x + v_y = 0. \tag{5.43}$$

In this setting, the velocity components are

$$u = \partial_y \psi, \tag{5.44}$$

and

$$v = -\partial_x \psi. \tag{5.45}$$

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

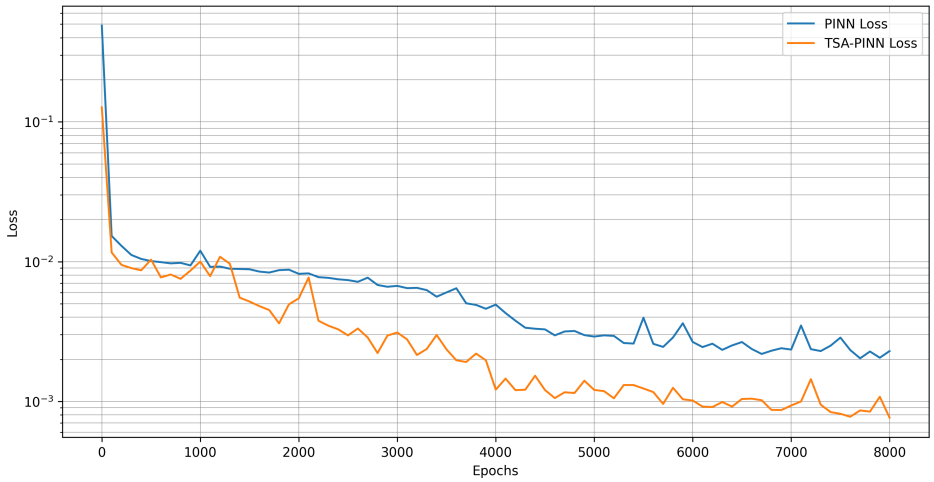
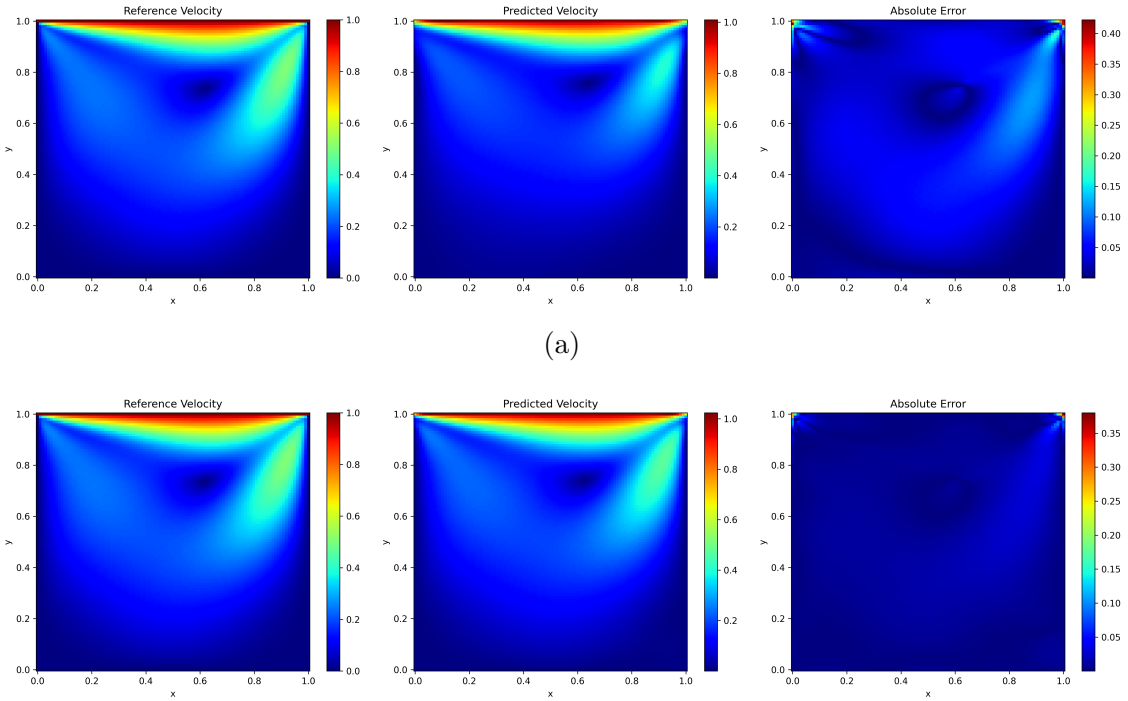


Figure 5.2 Lid-driven cavity problem at $Re = 100$: Comparison of loss decays.



(a)

(b)

Figure 5.3 Lid-driven cavity problem at $Re = 100$: Velocity contours and approximation error. (a) Standard PINN; (b) TSA-PINN.

governing equations are:

$$R_x = u\partial_x u + v\partial_y u + \partial_x p - \frac{1}{Re}(\partial^2_{xx} u + \partial^2_{yy} u); \tag{5.46}$$

$$R_y = u\partial_x v + v\partial_y v + \partial_y p - \frac{1}{Re}(\partial^2_{xx} v + \partial^2_{yy} v). \tag{5.47}$$

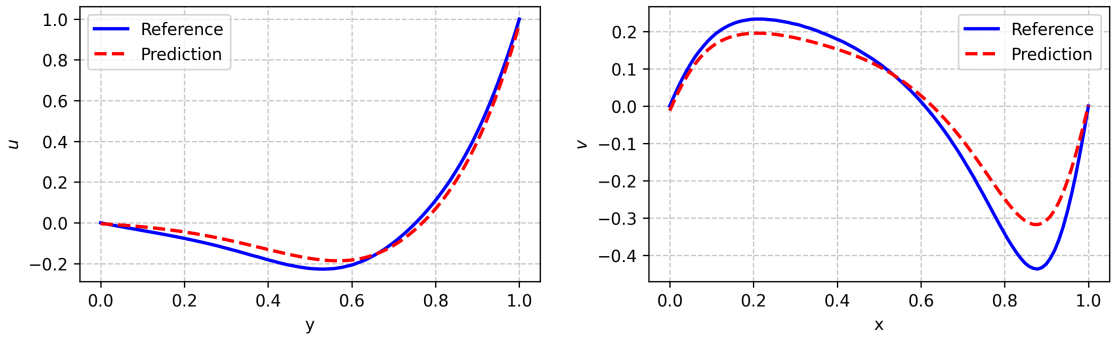
These terms are included in the loss function to satisfy the physics-informed part. The learning rate is set to exponential decay, starting from an initial value of $\eta = 10^{-3}$ with a decay rate of 0.9 every 1000 steps. For this case, training consists in 8,000 iterations. To estimate the error, 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. Figure 5.2 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 Figure 5.3. While the simulation using the standard PINN struggles to accurately capture the velocity fields (Figure 5.3-a), the TSA-PINN exhibits minimal error and accurately captures all phenomena (Figure 5.3-b).

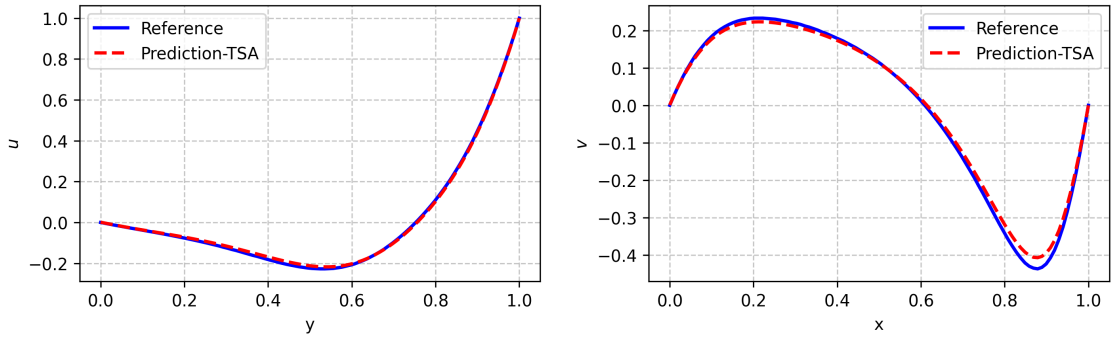
Figure 5.4 shows a quantitative analysis of the approximations made by both the TSA-PINN (Figure 5.4-b) and the standard PINN (Figure 5.4-a). This analysis quantitatively reaffirms the earlier conclusion, highlighting the superior accuracy of the TSA-PINN. The approximations made by the TSA-PINN aligns almost exactly with the reference solution. The standard PINN approximations exhibit noticeable errors. Table 5.1 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.

Table 5.1 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}



(a)



(b)

Figure 5.4 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.

5.3.2 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. [150], and to mitigate discontinuities at the two corners of the moving lid boundary, we reformulated the top boundary condition as:

$$u(x,y) = 1 - \frac{cosh(C_0(x-0.5))}{cosh(0.5C_0)}, \tag{5.48}$$

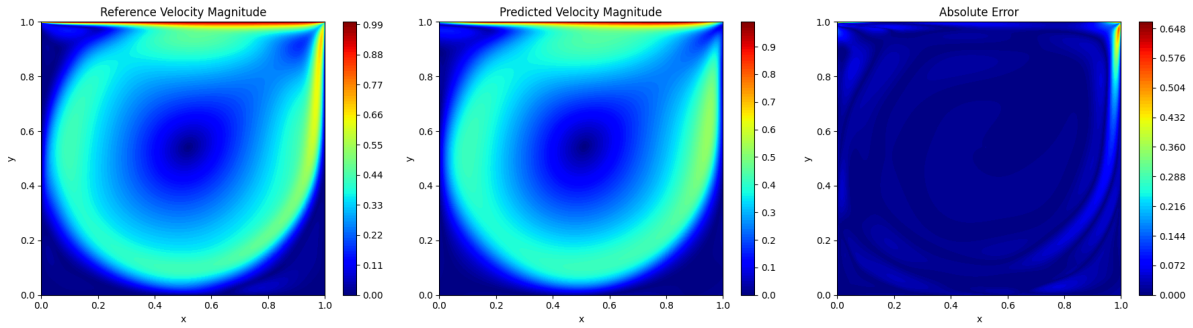
$$v(x,y) = 0, \tag{5.49}$$

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.

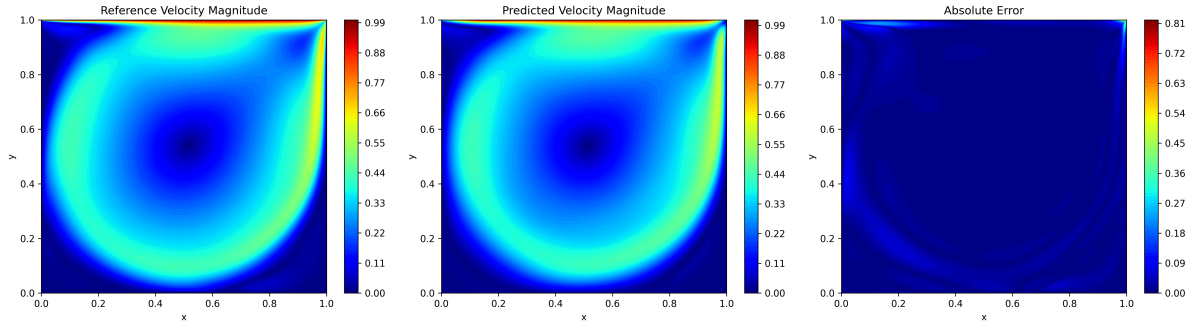
For the lid-driven cavity problem at $Re = 3200$, our results highlight the superior performance of TSA-PINN compared to the standard PINN framework. In Figure 5.5, the velocity contours and approximation errors are presented for both methods. In panel (a), the standard PINN results reveal less accurate 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 Figure 5.6, 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. Figure 5.7 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 5.2 summarizes the error metrics and runtime comparisons between 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

Table 5.2 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



(a)



(b)

Figure 5.5 Lid-driven cavity problem at $Re = 3200$: Velocity contours and approximation error. (a) Standard PINN; (b) TSA-PINN.

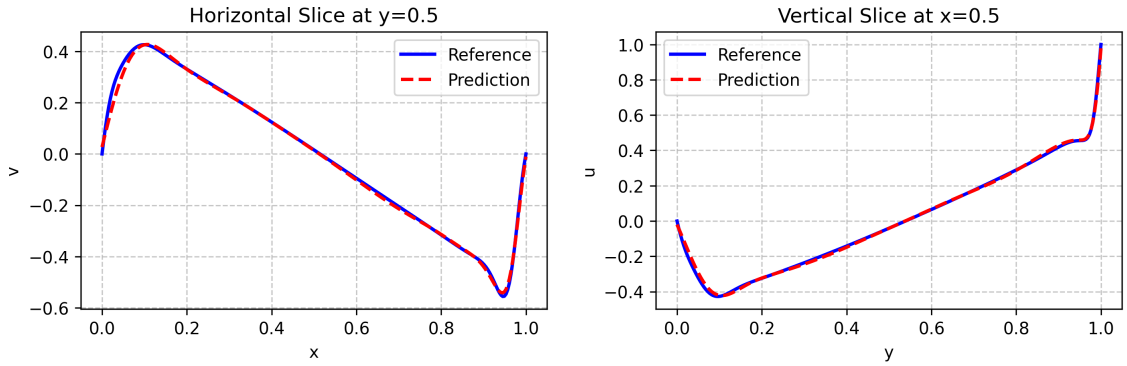


Figure 5.6 Lid-driven cavity problem at $Re = 3200$: Quantitative validation of the solution approximations at $x_{\text{constant}} = 0.5$ and $y_{\text{constant}} = 0.5$.

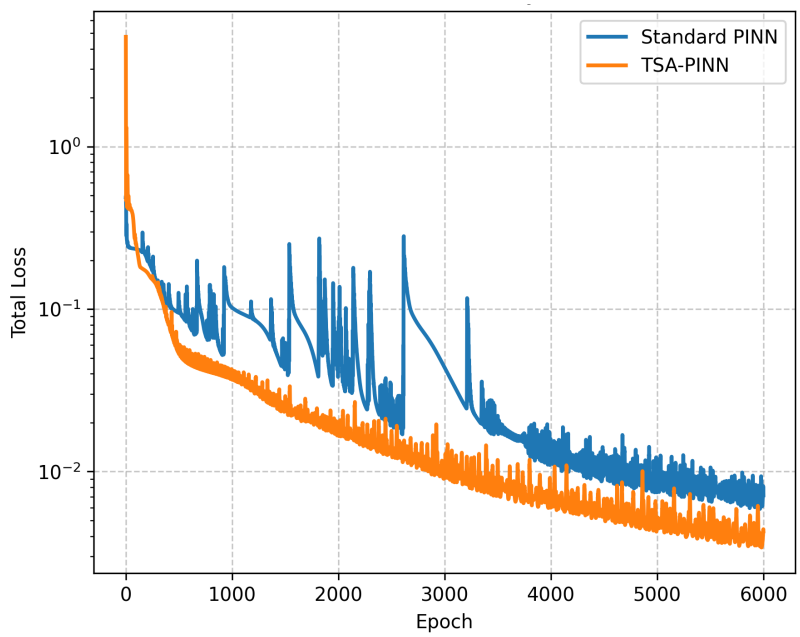


Figure 5.7 Lid-driven cavity problem at $Re = 3200$: Comparison of loss histories.

for simulating the complex flow phenomena encountered at $Re = 3200$.

5.3.3 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 (5.24) and (5.25). 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 [27]. 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 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

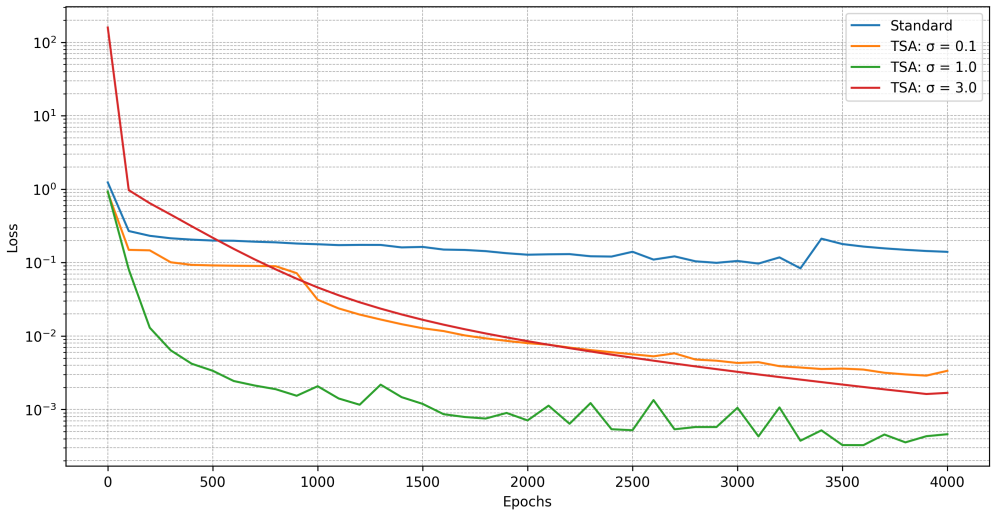


Figure 5.8 Cylinder wake problem: Comparison of the loss history.

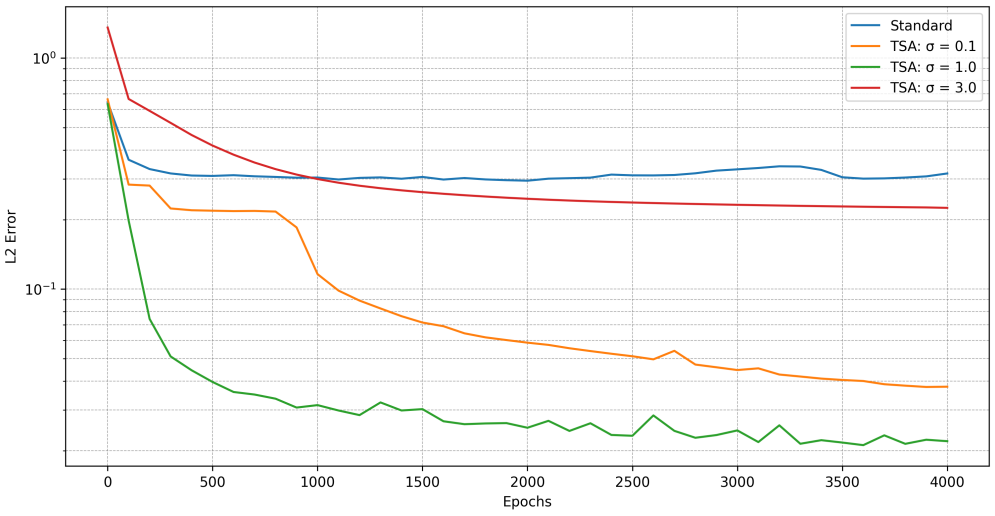


Figure 5.9 Cylinder wake problem: Comparison of the error history.

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:

$$R_x = \partial_t u + u\partial_x u + v\partial_y u + \partial_x p - \frac{1}{Re}(\partial^2_{xx} u + \partial^2_{yy} u); \tag{5.50}$$

$$R_y = \partial_t v + u\partial_x v + v\partial_y v + \partial_y p - \frac{1}{Re}(\partial^2_{xx} v + \partial^2_{yy} v). \tag{5.51}$$

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 more efficiently towards optimal parameter values. Figure 5.8 illustrates the loss decay comparison between the standard PINN and the TSA-PINN initialized with various frequency values σ. The loss history, illustrated in Figure 5.8, 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 sub-optimal 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 Figure 5.9 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-PINsN across a range of σ configurations shows 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 hyper-parameters.

Figure 5.10 shows the velocity contours at $t = 10.0$ s, comparing the predictions from a stan-

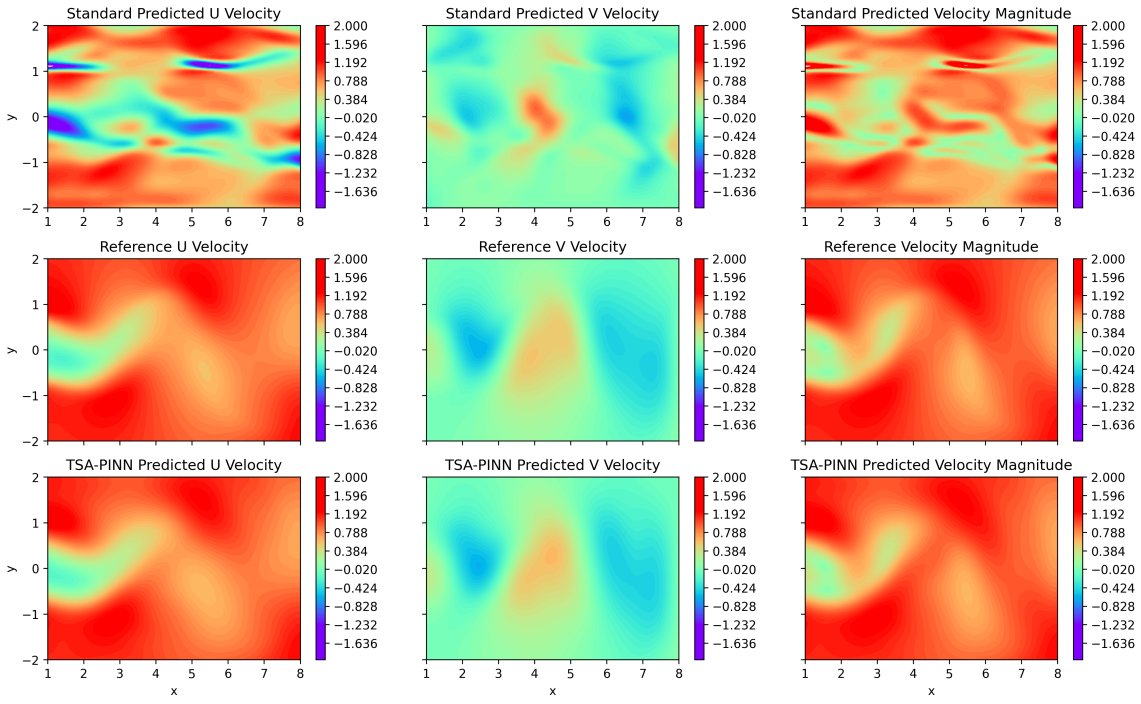
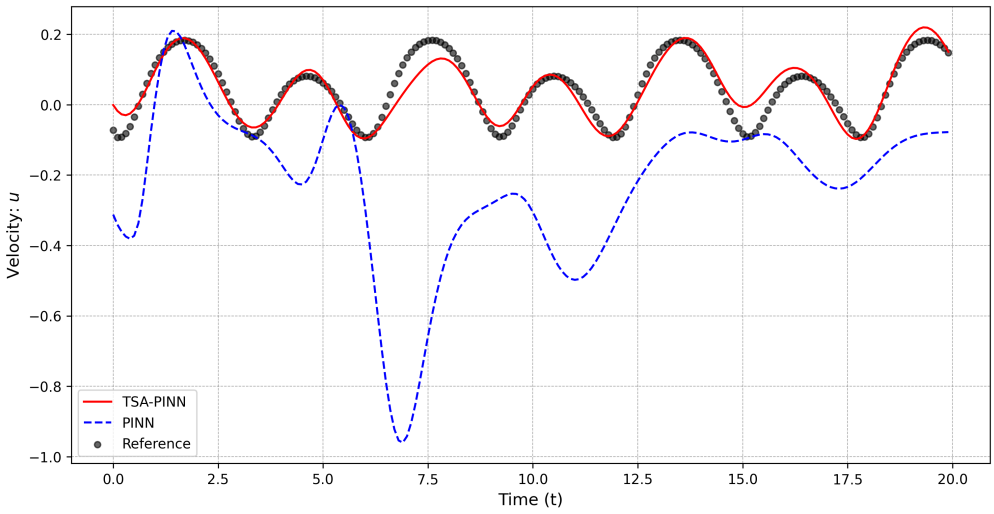


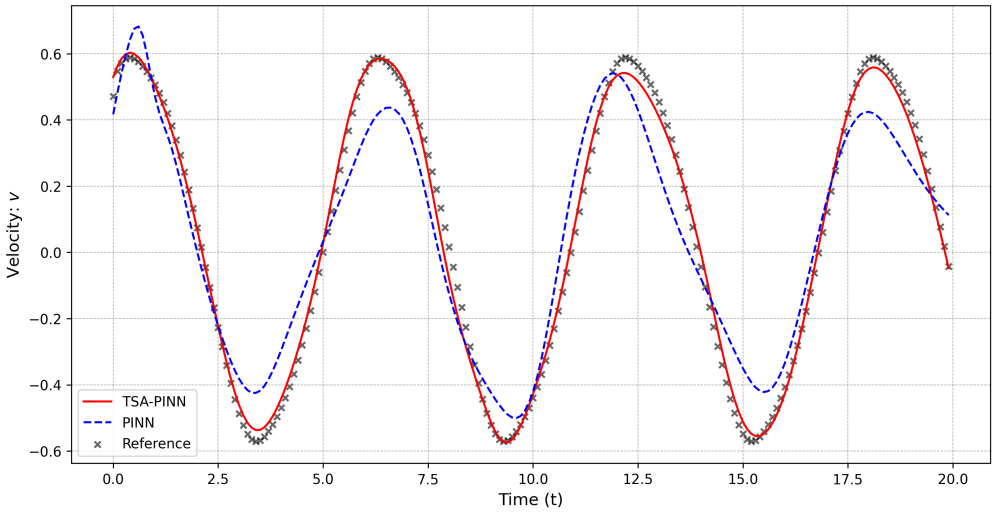
Figure 5.10 Cylinder wake problem: Comparison of velocity contours for Standard PINN (top) and TSA-PINN (bottom) against the reference solution (middle).

dard 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 formation 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.

Figure 5.11 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 5.3). 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



(a)



(b)

Figure 5.11 Cylinder wake problem: Quantitative comparison of the solutions at $x = 2.0$, $y = 0.04$. (a) u vs. t; (b) v vs. t.

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 5.3 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.

5.3.4 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 (5.24) and (5.25). 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 databases [1–3], 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,

Table 5.3 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}

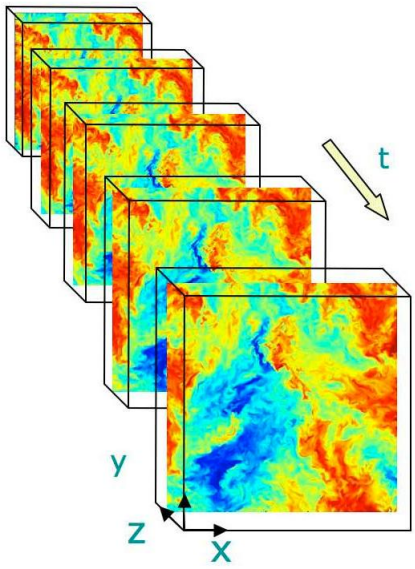
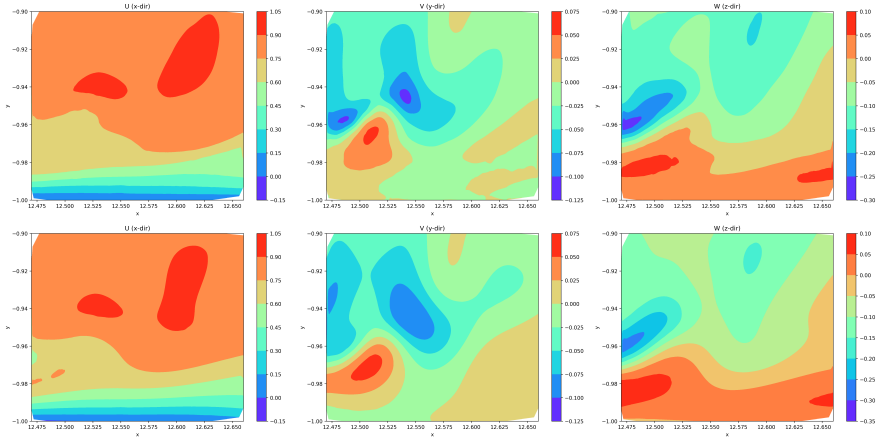


Figure 5.12 Schematics of the 3D turbulent channel flow problem [1–3]

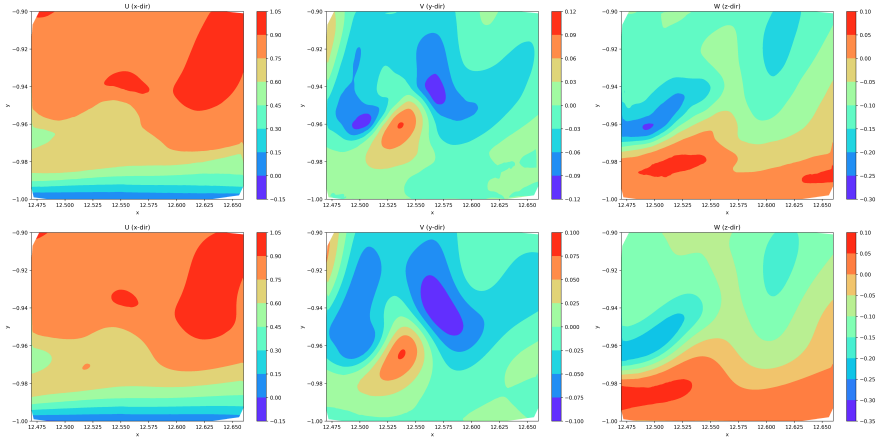
with a time-step size of 0.0065 as provided by the online database. The viscosity is $\nu = 5 \times 10^{-5}$, and the initial conditions represent a fully developed turbulence. The network's input and output dimensions consist in four variables each: (x, y, z, t) serve as inputs, while the corresponding outputs are the velocity components and pressure, (u, v, w, P). The schematics of the problem domain are depicted in Figure 5.12.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 [3], 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.

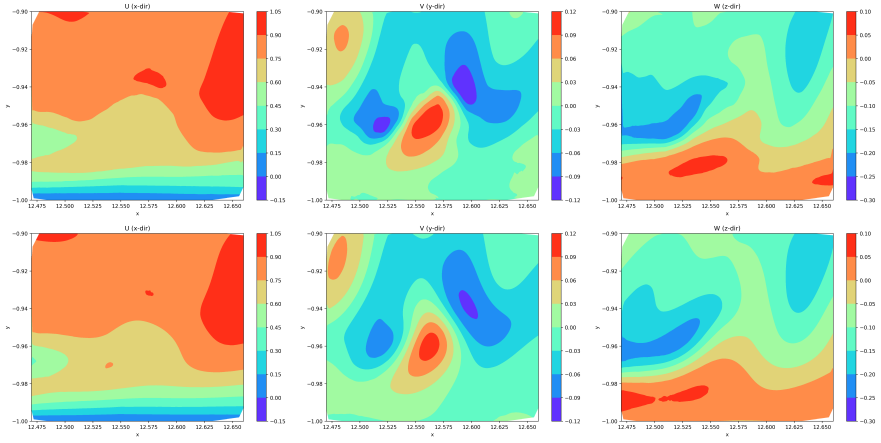
- Located above the viscous sub-layer and below the fully turbulent zone, the *buffer layer* serves as a transition zone where the effects of viscosity and turbulence intermingle. This layer is significant for both viscous and turbulent shears, and it is characterized by a complex velocity profile that neither follows a simple linear nor logarithmic law. It marks the region where turbulence production and dissipation are approximately



(a)



(b)



(c)

Figure 5.13 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 $t = 10 \times 0.0065$.

balanced.

- **L**ocated 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,

$$u^+ = \frac{1}{k}\log(y^+) + B, \tag{5.52}$$

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 (5.28), (5.29), (5.30), and (5.31).

Table 5.4 Error comparison for the 3D turbulent channel (Problem 4): Standard PINN vs. TSA-PINN.

Method	Time-step	L_2-err.: u	L_2-err.: v	L_2-err.: w
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}

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 Figure 5.13 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 5.4, 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. Figure 5.14 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 [3].

5.3.5 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

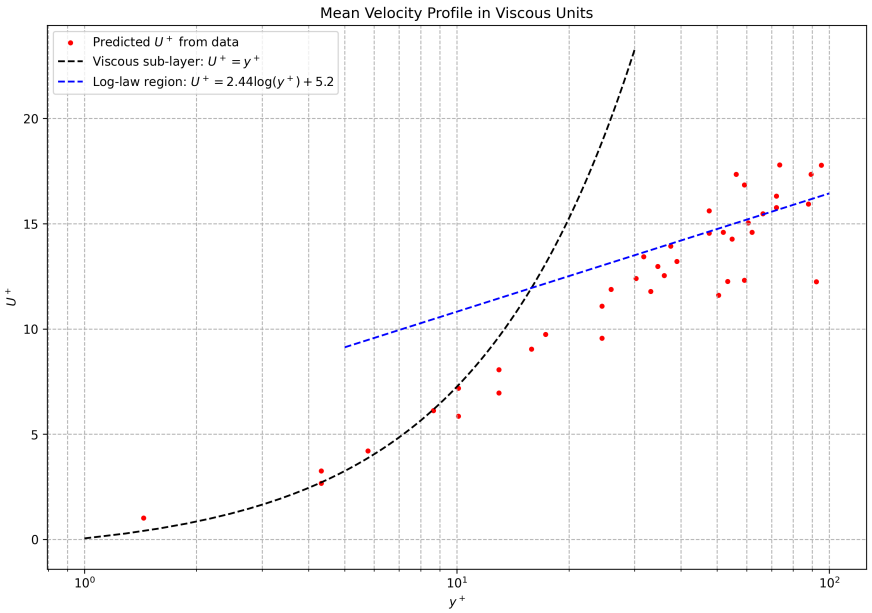
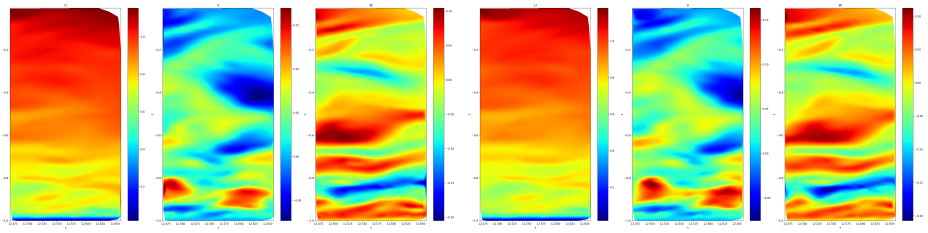


Figure 5.14 3D turbulent channel (Problem 4): Mean velocity profile in viscous units. Standard values of $k = 0.41$ and $B = 5.2$ are used in the log-law for reference.

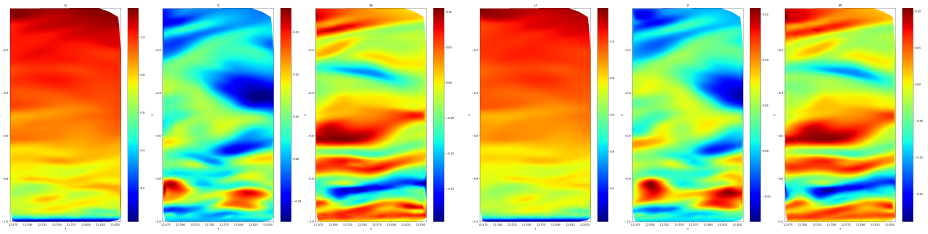
additional 250 epochs, and ultimately with to 10^{-6} for the final 150 epochs, maintaining 150 iterations per epoch.

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. Figure 5.15 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.

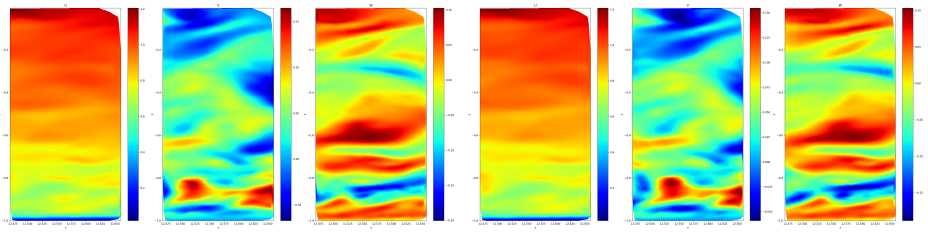
Figure 5.16 and Table 5.5 quantitatively confirm this observation where the L_2-norm of the errors associated with the models' numerical approximations are compared to the reference data. Within the interpolation domain, TSA-PINN's approximations are closely aligned with the reference solution, maintaining 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 Figure 5.16, 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



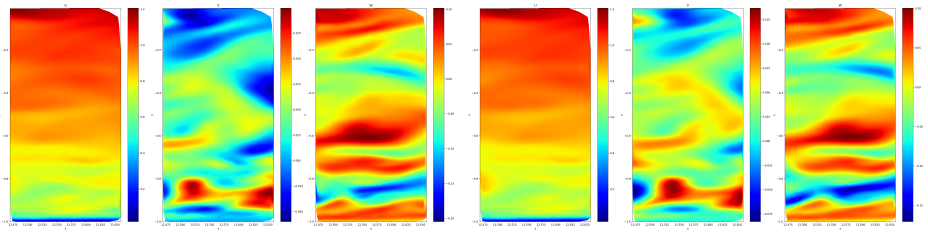
(a)



(b)



(c)



(d)

Figure 5.15 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).

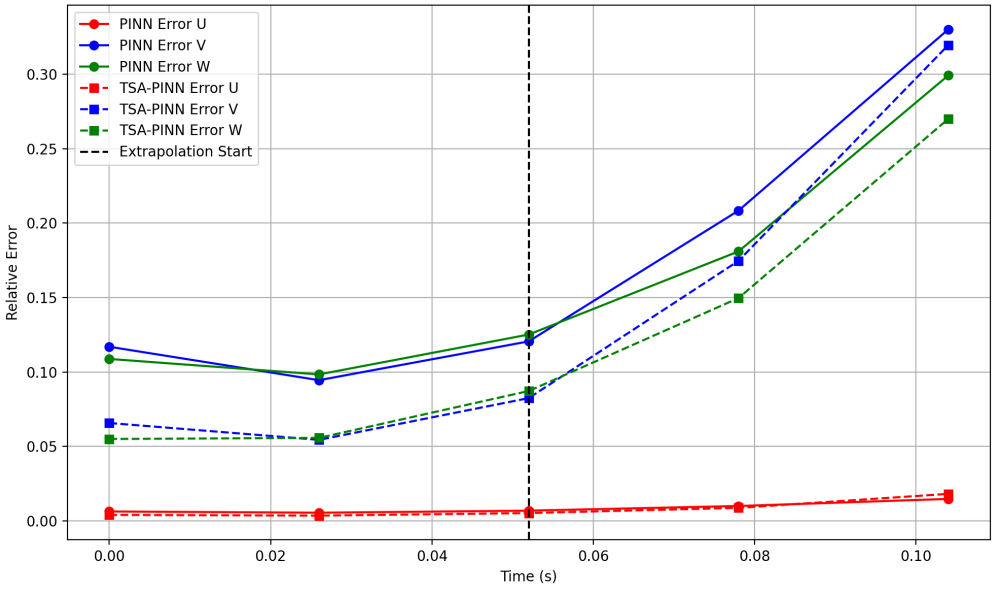


Figure 5.16 3D turbulent channel (Problem 5): Comparison of the error history.

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.

5.4 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

Table 5.5 Error comparison for the 3D turbulent channel (Problem 5): Standard PINN vs. TSA-PINN.

Method	Time-step	L_2-err.: u	L_2-err.: v	L_2-err.: w
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}

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.

CHAPTER 6 CONCLUSION

This thesis has contributed to the development of DL-based approaches, specifically Physics-Informed Neural Networks, for approximating the Navier-Stokes equations, a cornerstone of fluid dynamics. We have enhanced the PINN framework's ability to capture complex fluid behaviors under diverse conditions. The research focused on exploring the efficacy of domain decomposition, enhancing training efficiency through transformer networks, and innovating with temporal discretization augmented by generative modeling. Additionally, the implementation of an improved network architecture using a trainable sinusoidal activation mechanism marked a substantial progression in the field. Each chapter of the dissertation detailed methods that extend the boundaries of traditional PINN framework, thus enabling superior adaptability, accuracy, and expedited convergence for complex fluid dynamics simulations. The first article addressed the first specific objective by using a domain decomposition strategy, that allows for a more adaptable modeling framework, paired with subdomain-specific transformer networks. These networks project the input space into a higher dimensional feature space, which accelerates model convergence. The proposed models were validated against standard benchmark problems, including a lid-driven cavity flow, a 2D cylinder wake problem, and the viscous Burgers equation. While standard PINN models streamline the process of achieving continuous approximations, the enhanced approaches mitigate error propagation, boost the model's capability to capture local phenomena effectively, advance model's performance in stiff PDEs with highly discontinuous solutions, and set the stage for tackling multi-physics problems.

In the second article, a time dicretization technique, coupled with generative modeling to propagate information among discretized subdomains, enabled PINNs to be more effectively applied to high-dimensional problems. This improvement effectively addressed the second specific objective. The application of standard PINNs was mostly limited to canonical problems. With this improvement, we extended the application of PINN-based techniques to 3D turbulent problems, where inferring an approximate solution requires a large number of collocation points. While standard PINNs struggle to handle such high-dimensional problems, the proposed methodology effectively reached an accurate solution with fewer training epochs, particularly thanks to the integration of generative modeling with the proposed PINN-based method, along with a gating mechanism, allowing the accurate approximations from previous time steps to be shared with subsequent time steps as additional training information. This prevented error propagation in time.

In the final article, the third specific objective was tackled by implementing a novel network architecture that incorporates a neuron-wise trainable sinusoidal activation mechanism with adaptive frequencies and a frequency-based slope-recovery method. This configuration allows the learning system to adjust effectively to the frequency of the target function to swiftly capture highly oscillating and chaotic behaviors, commonly seen in nonlinear problems governed by the Navier-Stokes equations at high Reynolds numbers. The method underwent testing across a broad spectrum of scenarios—from well-explored standard problems like a 2D lid-driven cavity flow and a time-dependent cylinder wake problem to less-explored and more challenging scenarios like the 3D time-dependent turbulent channel flow at high Reynolds numbers. Specifically, the method was applied to two test scenarios in the latter problem: a smaller near-wall region to specifically study near-wall phenomena, such as the viscous sublayer, a log-law region, and a larger domain to simulate a myriad of phenomena driven by multi-scale eddies. The results show that the proposed technique not only accurately captures all the target dynamics and rapidly changing phenomena, but also showcased significantly higher accuracy in fewer training epochs compared to previous PINN-based techniques. The proposed approach also yielded relatively acceptable results in extrapolation tasks beyond the training domain, although it did not surpass the standard PINN in this respect.

6.1 Summary of Works

The following summarizes the key contributions made in this research, each of which marks a step forward in the use of deep learning techniques for approximating partial differential equations in fluid mechanics.

1. Enhanced domain decomposition: Developed a domain decomposition strategy enabling Physics-Informed Neural Networks to handle complex geometries and boundary conditions by dividing the computational domain into interconnected subdomains. This method improved the model's ability to simulate fluid dynamics with high accuracy and continuity across the interfaces of subdomains;

2. Integration of transformer networks: Implemented transformer networks for feature transformation within each subdomain, which projected the input space into a higher dimensional feature space conducive to enhanced model expressivity, enabling the capture of complex patterns and interactions within the data that are less discernible in lower-dimensional spaces.;

3. Advanced temporal discretization: Introduced a novel discretization method in time, combined with generative modeling mechanisms, to facilitate efficient information trans-

fer across time steps. This approach minimized error propagation and enhanced the capability of the PINN to predict future states in dynamics of fluid systems, especially in turbulent flow conditions;

4. Trainable sinusoidal activation mechanism: Proposed adaptable sinusoidal activation functions within the network architecture, which dynamically tuned the frequency response of neurons to match the oscillatory patterns observed in turbulent flows. This innovation led to greater model adaptability and improved prediction accuracy;

5. Expansion of applicability to complex fluid flows: Broadened the range of applicability of deep learning-based techniques to encompass a wide array of fluid flow problems, particularly involving high-dimensional 3D turbulent flows at high Reynolds numbers. This expansion facilitated the modeling of oscillating and nonlinear behaviors, as well as a diverse range of rapidly changing behaviors driven by sharp gradients and multi-scale eddies, showcasing the versatility and depth of our approach in addressing highly complex fluid dynamics scenarios.

6.2 Limitations

Despite the enhanced capabilities of the PINN methods in approximating solutions to the Navier-Stokes equations, several limitations remain:

- Dependance on training data: While PINN models integrate physical laws, their reliance on boundary data becomes crucial, especially in highly complex, high-dimensional problems with nonlinear behaviors. This reliance often surpasses the need for collocation points where residuals of the governing equations are known.

- Scalability issues: The complexity of the models and the substantial computational resources required can restrict their application at a larger scale, or in more complex geometrical settings.

- Sensitivity to hyperparameters: The effectiveness of PINN models is sometimes contingent on the selection of hyperparameters, such as network width, depth, and the weights of different loss terms, which typically involve a process of trial and error.

6.3 Future Research

Future research could focus on various areas to further enhance the capability of PINNs for PDE approximations:

- Theoretical works on hyperparameters: Developing rigorous theoretical frameworks for key architectural hyperparameters—particularly network width and depth—can reduce dependence on empirical tuning and enhance the robustness and adaptability of PINNs.;
- Real-time simulation capabilities: Improving the computational efficiency of PINNs to support real-time simulations of dynamical fluid systems, which would be particularly beneficial in applications requiring immediate data processing and decision-making;
- Broadening application domains: This research has begun to push the boundaries of where deep learning techniques can be applied in fluid dynamics, a field not yet widely explored in current literature. Future efforts could further extend the use of PINNs to tackle complex multiscale and multiphysics phenomena such as magnetorheological flows; as well as to address challenges in systems requiring significant spatio-temporal scaling, such as global and regional climate modeling or geosystem reservoir modeling;
- Integration with conventional computational methods: Investigating the synergistic integration of DL-based methods, specifically PINN-based methods, with established computational approaches, like the FEM, which could lead to more robust and versatile modeling frameworks.

In conclusion, this dissertation has set a foundational framework for advancing the application of deep learning within PDEs, particularly using PINNs to address complex problems governed by the Navier-Stokes equations. The methodologies developed herein pave the way for more accurate, efficient, and reliable simulations, potentially revolutionizing the application of deep learning in fluid dynamics.

REFERENCES

[1] E. Perlman, R. Burns, Y. Li, and 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.

[2] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink, "A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence," *Journal of Turbulence*, no. 9, p. N31, 2008.

[3] 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," *Journal of Turbulence*, vol. 17, no. 2, pp. 181–215, 2016.

[4] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[5] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, R. Tibshirani, and J. Friedman, "Overview of supervised learning," *The elements of statistical learning: Data mining, inference, and prediction*, pp. 9–41, 2009.

[6] G. I. Webb, E. Keogh, and R. Miikkulainen, "Naïve bayes." *Encyclopedia of machine learning*, vol. 15, no. 1, pp. 713–714, 2010.

[7] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[8] C. Kingsford and S. L. Salzberg, "What are decision trees?" *Nature biotechnology*, vol. 26, no. 9, pp. 1011–1013, 2008.

[9] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[10] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[11] D. F. Specht *et al.*, "A general regression neural network," *IEEE transactions on neural networks*, vol. 2, no. 6, pp. 568–576, 1991.

[12] M. Awad, R. Khanna, M. Awad, and R. Khanna, "Support vector regression," *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*, pp. 67–80, 2015.

[13] H. B. Barlow, "Unsupervised learning," *Neural computation*, vol. 1, no. 3, pp. 295–311, 1989.

[14] X. J. Zhu, "Semi-supervised learning literature survey," 2005.

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

[16] K. Gurney, *An introduction to neural networks*. CRC press, 2018.

[17] X. Guo, W. Li, and F. Iorio, "Convolutional neural networks for steady flow approximation," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 481–490.

[18] L. Prantl, B. Ummenhofer, V. Koltun, and N. Thuerey, "Guaranteed conservation of momentum for learning particle-based fluid dynamics," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6901–6913, 2022.

[19] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, "Deep fluids: A generative network for parameterized fluid simulations," in *Computer graphics forum*, vol. 38, no. 2. Wiley Online Library, 2019, pp. 59–70.

[20] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning mesh-based simulation with graph networks," *arXiv preprint arXiv:2010.03409*, 2020.

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

[22] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[23] M. G. Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," *communications in Numerical Methods in Engineering*, vol. 10, no. 3, pp. 195–201, 1994.

[24] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE transactions on neural networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[25] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries," *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1041–1049, 2000.

[26] M. Kumar and N. Yadav, "Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey," *Computers & Mathematics with Applications*, vol. 62, no. 10, pp. 3796–3811, 2011.

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

[28] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[29] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris, "Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 358, p. 112623, 2020.

[30] M. De Florio, E. Schiassi, B. D. Ganapol, and R. Furfaro, "Physics-informed neural networks for rarefied-gas dynamics: Poiseuille flow in the bgk approximation," *Zeitschrift für angewandte Mathematik und Physik*, vol. 73, no. 3, p. 126, 2022.

[31] M. Aliakbari, M. Mahmoudi, P. Vadasz, and A. Arzani, "Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport solvers using physics-informed neural networks," *International Journal of Heat and Fluid Flow*, vol. 96, p. 109002, 2022.

[32] D. W. Abueidda, S. Koric, E. Guleryuz, and N. A. Sobh, "Enhanced physics-informed neural networks for hyperelasticity," *International Journal for Numerical Methods in Engineering*, vol. 124, no. 7, pp. 1585–1601, 2023.

[33] C. Xu, B. T. Cao, Y. Yuan, and G. Meschke, "Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios," *Computer Methods in Applied Mechanics and Engineering*, vol. 405, p. 115852, 2023.

[34] B. Zapf, J. Haubner, M. Kuchta, G. Ringstad, P. K. Eide, and K.-A. Mardal, "Investigating molecular transport in the human brain from mri with physics-informed neural networks," *Scientific Reports*, vol. 12, no. 1, p. 15475, 2022.

[35] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM Journal on Scientific Computing*, vol. 43, no. 6, pp. B1105–B1132, 2021.

[36] L. Sun, H. Gao, S. Pan, and J.-X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," *Computer Methods in Applied Mechanics and Engineering*, vol. 361, p. 112732, 2020.

[37] M. Baymani, S. Effati, H. Niazmand, and A. Kerayechian, "Artificial neural network method for solving the navier–stokes equations," *Neural Computing and Applications*, vol. 26, pp. 765–773, 2015.

[38] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *Journal of Marchine Learning Research*, vol. 18, pp. 1–43, 2018.

[39] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020.

[40] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113028, 2020.

[41] A. D. Jagtap and 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.

[42] K. Shukla, A. D. Jagtap, and G. E. Karniadakis, "Parallel physics-informed neural networks via domain decomposition," *Journal of Computational Physics*, vol. 447, p. 110683, 2021.

[43] M. Penwarden, A. D. Jagtap, S. Zhe, G. E. Karniadakis, and R. M. Kirby, "A unified scalable framework for causal sweeping strategies for physics-informed neural networks

(pinns) and their temporal decompositions," *Journal of Computational Physics*, vol. 493, p. 112464, 2023.

[44] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, "Variational physics-informed neural networks for solving partial differential equations," *arXiv preprint arXiv:1912.00873*, 2019.

[45] ——, "hp-vpinns: Variational physics-informed neural networks with domain decomposition," *Computer Methods in Applied Mechanics and Engineering*, vol. 374, p. 113547, 2021.

[46] E. Lorin and X. Yang, "Time-dependent dirac equation with physics-informed neural networks: computation and properties," *Computer Physics Communications*, vol. 280, p. 108474, 2022.

[47] ——, "Quasi-optimal domain decomposition method for neural network-based computation of the time-dependent schrödinger equation," *Computer Physics Communications*, vol. 299, p. 109129, 2024.

[48] D. Zhang, Y. Li, and S. Ying, "Trans-net: A transferable pretrained neural networks based on temporal domain decomposition for solving partial differential equations," *Computer Physics Communications*, vol. 299, p. 109130, 2024.

[49] P. Ren, C. Rao, S. Chen, J.-X. Wang, H. Sun, and Y. Liu, "Seismicnet: Physics-informed neural networks for seismic wave modeling in semi-infinite domain," *Computer Physics Communications*, vol. 295, p. 109010, 2024.

[50] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[51] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[52] S. Cao, "Choose a transformer: Fourier or galerkin," *Advances in neural information processing systems*, vol. 34, pp. 24 924–24 940, 2021.

[53] H. Gao, M. J. Zahr, and J.-X. Wang, "Physics-informed graph neural galerkin networks: A unified framework for solving pde-governed forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 390, p. 114502, 2022.

[54] J. Sirignano and K. Spiliopoulos, "Dgm: A deep learning algorithm for solving partial differential equations," *Journal of computational physics*, vol. 375, pp. 1339–1364, 2018.

[55] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[56] P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y.-S. Ong, "Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method," *Computer Methods in Applied Mechanics and Engineering*, vol. 395, p. 114909, 2022.

[57] A. F. Psaros, K. Kawaguchi, and G. E. Karniadakis, "Meta-learning pinn loss functions," *Journal of computational physics*, vol. 458, p. 111121, 2022.

[58] M. Penwarden, S. Zhe, A. Narayan, and R. M. Kirby, "A metalearning approach for physics-informed neural networks (pinns): Application to parameterized pdes," *Journal of Computational Physics*, vol. 477, p. 111912, 2023.

[59] J. Oldenburg, F. Borowski, A. Öner, K.-P. Schmitz, and M. Stiehm, "Geometry aware physics informed neural network surrogate for solving navier–stokes equation (gapinn)," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 9, no. 1, p. 8, 2022.

[60] N. Wandel, M. Weinmann, M. Neidlin, and R. Klein, "Spline-pinn: Approaching pdes without data using fast, physics-informed hermite-spline cnns," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 8, 2022, pp. 8529–8538.

[61] L. D. McClenny and U. M. Braga-Neto, "Self-adaptive physics-informed neural networks," *Journal of Computational Physics*, vol. 474, p. 111722, 2023.

[62] G. K. Yadav, S. Natarajan, and B. Srinivasan, "Distributed pinn for linear elasticity—a unified approach for smooth, singular, compressible and incompressible media," *International Journal of Computational Methods*, vol. 19, no. 08, p. 2142008, 2022.

[63] S. Wang, H. Wang, and P. Perdikaris, "Improved architectures and training algorithms for deep operator networks," *Journal of Scientific Computing*, vol. 92, no. 2, p. 35, 2022.

[64] O. Hennigh, S. Narasimhan, M. A. Nabian, A. Subramaniam, K. Tangsali, Z. Fang, M. Rietmann, W. Byeon, and S. Choudhry, "Nvidia simnet™: An ai-accelerated multi-physics simulation framework," in *International conference on computational science*. Springer, 2021, pp. 447–461.

[65] A. Kendall, Y. Gal, and 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.

[66] Y. Liu, W. Liu, X. Yan, S. Guo, and C.-a. Zhang, "Adaptive transfer learning for pinn," *Journal of Computational Physics*, vol. 490, p. 112291, 2023.

[67] C. L. Wight and J. Zhao, "Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks," *arXiv preprint arXiv:2007.04542*, 2020.

[68] S. Shi, D. Liu, and Z. Zhao, "Non-fourier heat conduction based on self-adaptive weight physics-informed neural networks," in *2021 40th Chinese control conference (CCC)*. IEEE, 2021, pp. 8451–8456.

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

[70] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[71] S. Wang, S. Sankaran, and P. Perdikaris, "Respecting causality for training physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 421, p. 116813, 2024.

[72] Y. Song, H. Wang, H. Yang, M. L. Taccari, and X. Chen, "Loss-attentional physics-informed neural networks," *Journal of Computational Physics*, vol. 501, p. 112781, 2024.

[73] Z. Deng, C. He, Y. Liu, and K. C. Kim, "Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework," *Physics of Fluids*, vol. 31, no. 12, 2019.

[74] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, vol. 807, pp. 155–166, 2016.

[75] J.-M. Lévy-Leblond, "Galilei group and galilean invariance," in *Group theory and its applications*. Elsevier, 1971, pp. 221–299.

[76] R. Maulik, O. San, A. Rasheed, and P. Vedula, "Subgrid modelling for two-dimensional turbulence using neural networks," *Journal of Fluid Mechanics*, vol. 858, pp. 122–144, 2019.

[77] R. H. Kraichnan, "Inertial ranges in two-dimensional turbulence," *The Physics of Fluids*, vol. 10, no. 7, pp. 1417–1423, 1967.

[78] H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, "Physics-informed neural networks for solving reynolds-averaged navier–stokes equations," *Physics of Fluids*, vol. 34, no. 7, 2022.

[79] H. Xu, W. Zhang, and Y. Wang, "Explore missing flow dynamics by physics-informed deep learning: The parameterized governing systems," *Physics of Fluids*, vol. 33, no. 9, 2021.

[80] C. Cheng, H. Meng, Y.-Z. Li, and G.-T. Zhang, "Deep learning based on pinn for solving 2 dof vortex induced vibration of cylinder," *Ocean Engineering*, vol. 240, p. 109932, 2021.

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

[82] Y. Patel, V. Mons, O. Marquet, and G. Rigas, "Turbulence model augmented physics-informed neural networks for mean-flow reconstruction," *Physical Review Fluids*, vol. 9, no. 3, p. 034605, 2024.

[83] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and 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.

[84] L. Sliwinski and G. Rigas, "Mean flow reconstruction of unsteady flows using physics-informed neural networks," *Data-Centric Engineering*, vol. 4, p. e4, 2023.

[85] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations," *Journal of Computational Physics*, vol. 426, p. 109951, 2021.

[86] C.-C. Yu, Y.-C. Tang, and 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.

[87] S. Qian, H. Liu, C. Liu, S. Wu, and H. San Wong, "Adaptive activation functions in convolutional neural networks," *Neurocomputing*, vol. 272, pp. 204–212, 2018.

[88] M. Dushkoff and R. Ptucha, "Adaptive activation functions for deep networks," *Electronic Imaging*, vol. 28, pp. 1–5, 2016.

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

[90] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *Journal of Computational Physics*, vol. 404, p. 109136, 2020.

[91] A. D. Jagtap, K. Kawaguchi, and G. Em Karniadakis, "Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks," *Proceedings of the Royal Society A*, vol. 476, no. 2239, p. 20200334, 2020.

[92] V. Dwivedi, N. Parashar, and B. Srinivasan, "Distributed learning machines for solving forward and inverse problems in partial differential equations," *Neurocomputing*, vol. 420, pp. 299–316, 2021.

[93] K. Tang, X. Wan, and C. Yang, "Das-pinns: A deep adaptive sampling method for solving high-dimensional partial differential equations," *Journal of Computational Physics*, vol. 476, p. 111868, 2023.

[94] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[95] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar, "Physics-informed neural operator for learning partial differential equations," *ACM/JMS Journal of Data Science*, vol. 1, no. 3, pp. 1–27, 2024.

[96] O. Fuks and H. A. Tchelepi, "Limitations of physics informed machine learning for nonlinear two-phase transport in porous media," *Journal of Machine Learning for Modeling and Computing*, vol. 1, no. 1, 2020.

[97] B. Steinfurth and J. Weiss, "Assimilating experimental data of a mean three-dimensional separated flow using physics-informed neural networks," *Physics of Fluids*, vol. 36, no. 1, 2024.

[98] Y. Zhu, W. Kong, J. Deng, and X. Bian, "Physics-informed neural networks for incompressible flows with moving boundaries," *Physics of Fluids*, vol. 36, no. 1, 2024.

[99] Z. Cao, K. Liu, K. Luo, Y. Cheng, and J. Fan, "Efficient optimization design of flue deflectors through parametric surrogate modeling with physics-informed neural networks," *Physics of Fluids*, vol. 35, no. 12, 2023.

[100] J.-Z. Peng, Y. Hua, Y.-B. Li, Z.-H. Chen, W.-T. Wu, and N. Aubry, "Physics-informed graph convolutional neural network for modeling fluid flow and heat convection," *Physics of Fluids*, vol. 35, no. 8, 2023.

[101] W. Chen, P. Gao, and P. Stinis, "Physics-informed machine learning of the correlation functions in bulk fluids," *Physics of Fluids*, vol. 36, no. 1, 2024.

[102] S. Zhang, J. Deng, X. Li, Z. Zhao, J. Wu, W. Li, Y.-G. Wang, and D.-S. Jeng, "Solving the one dimensional vertical suspended sediment mixing equation with arbitrary eddy diffusivity profiles using temporal normalized physics-informed neural networks," *Physics of Fluids*, vol. 36, no. 1, 2024.

[103] M. Mahmoudabadbozchelou, G. E. Karniadakis, and S. Jamali, "nn-pinns: Non-newtonian physics-informed neural networks for complex fluid modeling," *Soft Matter*, vol. 18, no. 1, pp. 172–185, 2022.

[104] G. Lei, N. Ma, B. Sun, K. Mao, B. Chen, and Y. Zhai, "Physics-informed neural networks for solving nonlinear bloch equations in atomic magnetometry," *Physica Scripta*, vol. 98, no. 8, p. 085010, 2023.

[105] Y. Ye, X. Liu, Y. Li, H. Fan, and H. Zhang, "Deep neural network method for solving the fractional burgers-type equations with conformable derivative," *Physica Scripta*, vol. 98, no. 6, p. 065214, 2023.

[106] D. Baleanu, Y. Karaca, L. Vázquez, and J. E. Macías-Díaz, "Advanced fractional calculus, differential equations and neural networks: Analysis, modeling and numerical computations," *Physica Scripta*, vol. 98, no. 11, p. 110201, 2023.

[107] K. Li, K. Tang, T. Wu, and Q. Liao, "D3m: A deep domain decomposition method for partial differential equations," *IEEE Access*, vol. 8, pp. 5283–5294, 2019.

[108] D. J. S. Aulakh, S. B. Beale, and J. G. Pharoah, "A generalized framework for unsupervised learning and data recovery in computational fluid dynamics using discretized loss functions," *Physics of Fluids*, vol. 34, no. 7, 2022.

[109] W. Zhou, S. Miwa, and K. Okamoto, "Advancing fluid dynamics simulations: A comprehensive approach to optimizing physics-informed neural networks," *Physics of Fluids*, vol. 36, no. 1, 2024.

[110] S.-M. Qin, M. Li, T. Xu, and S.-Q. Dong, "Am-gpinn algorithm and its application in a variable-coefficient resonant nonlinear schrödinger equation," *Physica Scripta*, vol. 98, no. 2, p. 025219, 2023.

[111] S. Wang, H. Wang, J. H. Seidman, and P. Perdikaris, "Random weight factorization improves the training of continuous neural representations," *arXiv preprint arXiv:2210.01274*, 2022.

[112] C.-H. Bruneau and M. Saad, "The 2d lid-driven cavity problem revisited," *Computers & fluids*, vol. 35, no. 3, pp. 326–348, 2006.

[113] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, p. 122, 2019.

[114] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: Where we are and what's next," *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, 2022.

[115] S. Mishra and R. Molinaro, "Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes," *IMA Journal of Numerical Analysis*, vol. 42, no. 2, pp. 981–1022, 2022.

[116] X. Glorot and 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.

[117] 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.

[118] C. Basdevant, M. Deville, P. Haldenwang, J. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, and A. Patera, "Spectral and finite difference solutions of the burgers equation," *Computers & fluids*, vol. 14, no. 1, pp. 23–41, 1986.

[119] F. Zacchei, F. Rizzini, G. Gattere, A. Frangi, and A. Manzoni, "Neural networks based surrogate modeling for efficient uncertainty quantification and calibration of mems accelerometers," *International Journal of Non-Linear Mechanics*, vol. 167, p. 104902, 2024.

[120] W. Jia, M. Hu, W. Zan, and F. Ni, "Data-driven methods for the inverse problem of suspension system excited by jump and diffusion stochastic track excitation," *International Journal of Non-Linear Mechanics*, vol. 166, p. 104819, 2024.

[121] Y. Zhang, R. P. Dwight, M. Schmelzer, J. F. Gómez, Z.-h. Han, and S. Hickel, "Customized data-driven rans closures for bi-fidelity les–rans optimization," *Journal of Computational Physics*, vol. 432, p. 110153, 2021.

[122] Z. Gou, X. Tu, S. V. Lototsky, and R. Ghanem, "Switching diffusions for multiscale uncertainty quantification," *International Journal of Non-Linear Mechanics*, p. 104793, 2024.

[123] V. Mahesh, "Artificial neural network (ann) based investigation on the static behaviour of piezo-magneto-thermo-elastic nanocomposite sandwich plate with cnt agglomeration and porosity," *International Journal of Non-Linear Mechanics*, vol. 153, p. 104406, 2023.

[124] S. K. Mitusch, S. W. Funke, and M. Kuchta, "Hybrid fem-nn models: Combining artificial neural networks with the finite element method," *Journal of Computational Physics*, vol. 446, p. 110651, 2021.

[125] F. M. de Lara and E. Ferrer, "Accelerating high order discontinuous galerkin solvers using neural networks: 3d compressible navier-stokes equations," *Journal of Computational Physics*, p. 112253, 2023.

[126] W. Chen, Q. Wang, J. S. Hesthaven, and C. Zhang, "Physics-informed machine learning for reduced-order modeling of nonlinear problems," *Journal of computational physics*, vol. 446, p. 110666, 2021.

[127] Y. Kim, Y. Choi, D. Widemann, and T. Zohdi, "A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder," *Journal of Computational Physics*, vol. 451, p. 110841, 2022.

[128] A. Khademi and S. Dufour, "A novel discretized physics-informed neural network model applied to the navier-stokes equations," *Physica Scripta*, 2024.

[129] L. McClenny and U. Braga-Neto, "Self-adaptive physics-informed neural networks using a soft attention mechanism," *arXiv preprint arXiv:2009.04544*, 2020.

[130] S. Wang, S. Sankaran, and P. Perdikaris, "Respecting causality is all you need for training physics-informed neural networks," *arXiv preprint arXiv:2203.07404*, 2022.

[131] M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, vol. 357, pp. 125–141, 2018.

[132] S. K. Biswas and N. Anand, "Three-dimensional laminar flow using physics informed deep neural networks," *Physics of Fluids*, vol. 35, no. 12, 2023.

[133] S. Hu, M. Liu, S. Zhang, S. Dong, and R. Zheng, "Physics-informed neural network combined with characteristic-based split for solving navier–stokes equations," *Engineering Applications of Artificial Intelligence*, vol. 128, p. 107453, 2024.

[134] S. Hijazi, M. Freitag, and N. Landwehr, "Pod-galerkin reduced order models and physics-informed neural networks for solving inverse problems for the navier–stokes equations," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 10, no. 1, p. 5, 2023.

[135] J. Cho, S. Nam, H. Yang, S.-B. Yun, Y. Hong, and E. Park, "Separable physics-informed neural networks," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[136] S. J. Anagnostopoulos, J. D. Toscano, N. Stergiopulos, and G. E. Karniadakis, "Residual-based attention in physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 421, p. 116805, 2024.

[137] G. Jin, J. C. Wong, A. Gupta, S. Li, and Y.-S. Ong, "Fourier warm start for physics-informed neural networks," *Engineering Applications of Artificial Intelligence*, vol. 132, p. 107887, 2024.

[138] S. Wang, M. Nikfar, J. C. Agar, and Y. Liu, "Stacked deep learning models for fast approximations of steady-state navier–stokes equations for low re flow," *Intelligent computing*, vol. 3, p. 0093, 2024.

[139] A. Beck, D. Flad, and C.-D. Munz, "Deep neural networks for data-driven les closure models," *Journal of Computational Physics*, vol. 398, p. 108910, 2019.

[140] Y. Zhao, H. D. Akolekar, J. Weatheritt, V. Michelassi, and R. D. Sandberg, "Rans turbulence model development using cfd-driven machine learning," *Journal of Computational Physics*, vol. 411, p. 109413, 2020.

[141] S. Hanrahan, M. Kozul, and R. Sandberg, "Studying turbulent flows with physics-informed neural networks and sparse data," *International Journal of Heat and Fluid Flow*, vol. 104, p. 109232, 2023.

[142] H. Schlichting, K. Gersten, H. Schlichting, and K. Gersten, "Boundary-layer control (suction/blowing)," *Boundary-Layer Theory*, pp. 291–320, 2000.

[143] F. M. White and J. Majdalani, *Viscous fluid flow*. McGraw-Hill New York, 2006, vol. 3.

[144] M. Lee and R. D. Moser, "Direct numerical simulation of turbulent channel flow up to," *Journal of fluid mechanics*, vol. 774, pp. 395–415, 2015.

[145] S. Hoyas and J. Jiménez, "Scaling of the velocity fluctuations in turbulent channels up to reτ= 2003," *Physics of fluids*, vol. 18, no. 1, 2006.

[146] J. C. Del Alamo and J. Jiménez, "Spectra of the very large anisotropic scales in turbulent channels," *Physics of Fluids*, vol. 15, no. 6, pp. L41–L44, 2003.

[147] S. Cant, "Sb pope, turbulent flows, cambridge university press, cambridge, uk, 2000, 771 pp." *Combustion and Flame*, vol. 125, no. 4, pp. 1361–1362, 2001.

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

[149] Q. Lou, X. Meng, and G. E. Karniadakis, "Physics-informed neural networks for solving forward and inverse flow problems via the boltzmann-bgk formulation," *Journal of Computational Physics*, vol. 447, p. 110676, 2021.

[150] S. Wang, B. Li, Y. Chen, and P. Perdikaris, "Piratenets: Physics-informed deep learning with residual adaptive networks," *Journal of Machine Learning Research*, vol. 25, no. 402, pp. 1–51, 2024.