A novel discretized physics-informed neural network model applied to the Navier–Stokes equations

Physica Scripta

PAPER

A novel discretized physics-informed neural network model applied to the Navier–Stokes equations

Amirhossein Khademi and Steven Dufour

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

E-mail: Amirhossein.khademi@polymtl.ca

Keywords: deep learning, computational physics, Navier-Stokes equations, neural networks, PINN

Supplementary material for this article is available online

OPEN ACCESS

RECEIVED
28 March 2024

REVISED
21 May 2024

ACCEPTED FOR PUBLICATION
7 June 2024

PUBLISHED
20 June 2024

Abstract

The advancement of scientific machine learning (ML) techniques has led to the development of methods for approximating solutions to nonlinear partial differential equations (PDE) with increased efficiency and accuracy. Automatic differentiation has played a pivotal role in this progress, enabling the creation of physics-informed neural networks (PINN) that integrate relevant physics into machine learning models. PINN have shown promise in approximating the solutions to the Navier–Stokes equations, overcoming the limitations of traditional numerical discretization methods. However, challenges such as local minima and long training times persist, motivating the exploration of domain decomposition techniques to improve it. Previous domain decomposition models have introduced spatial and temporal domain decompositions but have yet to fully address issues of smoothness and regularity of global solutions. In this study, we present a novel domain decomposition approach for PINN, termed domain-discretized PINN (DD-PINN), which incorporates complementary loss functions, subdomain-specific transformer networks (TRF), and independent optimization within each subdomain. By enforcing continuity and differentiability through interface constraints and leveraging the Sobolev (H^1) norm of the mean squared error (MSE), rather than the Euclidean norm (L^2), DD-PINN enhances solution regularity and accuracy. The inclusion of TRF in each subdomain facilitates feature extraction and improves convergence rates, as demonstrated through simulations of threetest problems: steady-state flow in a two-dimensional lid-driven cavity, the time-dependent cylinder wake, and the viscous Burgers equation. Numerical comparisons highlight the effectiveness of DD-PINN in preserving global solution regularity and accurately approximating complex phenomena, marking a significant advancement over previous domain decomposition methods within the PINN framework.

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 [1], which makes it possible to bypass numerical discretization and mesh generation, leading to a new machine learning-based technique known as PINN [2, 3]. Since the introduction of the PINN, it has been widely used to approximate solution to PDE [4–16], 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 [17] 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* [18] further extended this concept by incorporating variational techniques and overlapping domain decompositions in PINN. On the other hand, variational PINN [19] and later *hp*-refinement strategies were developed using non-overlapping domain decompositions and projections onto a space of high-order polynomials [20]. The conservative PINN (CPINN) [21] 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 [22]. Subsequently, a parallel PINN [23] was proposed to complement the scenario previously established by the CPINN. Aulakh, Beale, and Pharoah [24] 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 [25]. 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* [26] 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, Teng, and Perdikaris [8] proposed an improved fully connected neural architecture by utilizing transformer networks 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 [27]. Wang explored the application of this improved architecture for studying fluid flow problems [28].

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 contributions 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 threetest problems [3, 29]; the steady-state flow in a two-dimensional lid-driven cavity problem, the time-dependent two-dimensional cylinder wake, both described by the two-dimensional (2D) incompressible Navier–Stokes equations, and the viscous Burgers equation. The results of numerical comparisons in thetest 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.

2. Methodology

2.1. Overview

2.1.1. 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 [30–32]. 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

$$f_i(\boldsymbol{x}_i;\, \boldsymbol{w}_i,\, \boldsymbol{b}_i) = \alpha_i(\boldsymbol{w}_i \cdot \boldsymbol{x}_i + \boldsymbol{b}_i), \tag{1}$$

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 [33], such that equation (1) can be also written as

$$\boldsymbol{u}_\theta(x) = C_K \circ \alpha \circ C_{K-1} \ldots \circ \alpha \circ C_1(x), \tag{2}$$

where, for any $k = 1, 2, \ldots, K$

$$C_k(x_k) = \boldsymbol{W}_k x_k + \boldsymbol{b}_k, \tag{3}$$

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.

2.1.2. 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

$$L = MSE_b + MSE_F, \tag{4}$$

where the mean squared errors are given by:

$$MSE_b(\boldsymbol{\theta}, \{\boldsymbol{x}_u^i\}_{i=1}^{N_u}) = \frac{1}{N_u}\sum_{i=1}^{N_u}|u^i - u_\theta(\boldsymbol{x})_u^i|^2; \tag{5}$$

$$MSE_F(\boldsymbol{\theta}, \{\boldsymbol{x}_F^i\}_{i=1}^{N_F}) = \frac{1}{N_F}\sum_{i=1}^{N_F}|F_\theta(\boldsymbol{x})_F^i|^2. \tag{6}$$

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.

2.1.3. 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,

$$L = MSE_b + MSE_F + MSE_{\tilde{\boldsymbol{u}}} + MSE_R. \tag{7}$$

The MSEs are defined as:

$$MSE_b(\boldsymbol{\theta}_\Omega, \{\boldsymbol{x}_u^i\}_{i=1}^{N_u}) = \frac{1}{N_{u_\Omega}}\sum_{i=1}^{N_{u\Omega}}|u^i - u_{\theta_\Omega}(\boldsymbol{x})_{u_\Omega}^i|^2; \tag{8}$$

$$MSE_F(\boldsymbol{\theta}_\Omega, \{\boldsymbol{x}_F^i\}_{i=1}^{N_F}) = \frac{1}{N_{F_\Omega}}\sum_{i=1}^{N_{F\Omega}}|F_{\theta_\Omega}(\boldsymbol{x})_{F_\Omega}^i|^2; \tag{9}$$

$$MSE_{\tilde{\boldsymbol{u}}}(\boldsymbol{\theta}_\Omega, \{\boldsymbol{x}_I^i\}_{i=1}^{N_I}) = \sum_{\forall\,\Omega^+}\left(\frac{1}{N_{I_\Omega}}\sum_{i=1}^{N_{I\Omega}}|u_{\theta_\Omega}(\boldsymbol{x})_{I_\Omega}^i - \{\{u_{\theta_\Omega}(\boldsymbol{x})_{I_\Omega}^i\}\}|^2\right); \tag{10}$$

$$MSE_R(\boldsymbol{\theta}_\Omega, \{\boldsymbol{x}_I^i\}_{i=1}^{N_I}) = \sum_{\forall\,\Omega^+}\left(\frac{1}{N_{I_\Omega}}\sum_{i=1}^{N_{I\Omega}}|F_{\theta_\Omega}(\boldsymbol{x}_{I_\Omega}^i) - F_{\theta_{\Omega^+}}(\boldsymbol{x}_{I_\Omega}^i)|^2\right). \tag{11}$$

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: $MSE_{\tilde{\boldsymbol{u}}}$ and MSE_R. These terms account for flux and solution continuity at the interfaces [22]. In $MSE_{\tilde{\boldsymbol{u}}}$, 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.

2.2. DD-PINN

2.2.1. Domain discretization

In the domain decomposition technique, the domain is subdivided into several non-overlapping subdomains, as depicted in figure 1. The entire domain can then be defined as

$$\Omega = \bigcup_{i=1}^{N_s}\Omega_i, \tag{12}$$

where

$$\Omega_i \cap \Omega_j = \partial\Omega_{ij}, \tag{13}$$

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

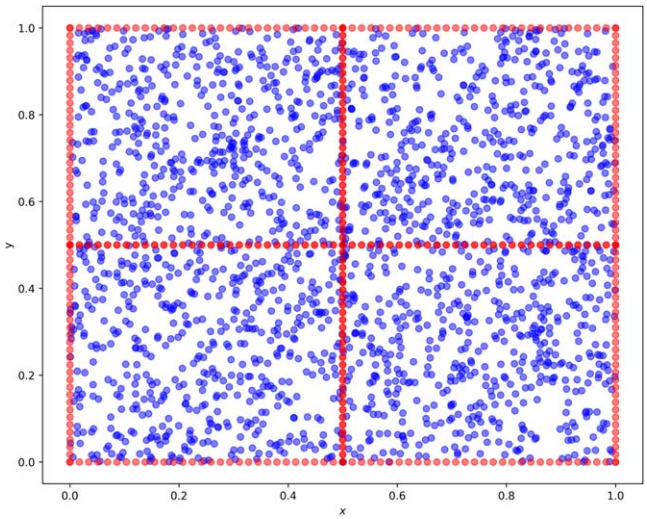


Figure 1. Decomposed domain; collocation points are in blue and the evaluation points of boundary conditions and interface constraints are in red.

$$\boldsymbol{u}_{\theta_i} = N^L(\boldsymbol{x}, \boldsymbol{\theta}_i), \tag{14}$$

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

$$\boldsymbol{u}_\theta = \bigcup_{i=1}^{N_s} \boldsymbol{u}_{\theta_i}. \tag{15}$$

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 (section 2.2.3). 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 (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 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.

2.2.2. 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

$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{1}{Re}\Delta \boldsymbol{u} = 0 \qquad \text{in} \quad \Omega; \tag{16}$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \text{in} \quad \Omega; \tag{17}$$

$$\boldsymbol{u}(x, y) = (1, 0) \qquad \text{on} \quad \Gamma_1; \tag{18}$$

$$\boldsymbol{u}(x, y) = (0, 0) \qquad \text{on} \quad \Gamma_0, \tag{19}$$

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,

$$u_x + v_y = 0. \tag{20}$$

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

$$u = \partial\psi/\partial y, \qquad v = -\partial\psi/\partial x, \tag{21}$$

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:

$$F_{\theta_x} = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right); \tag{22}$$

$$F_{\theta_y} = u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{23}$$

where F_{θ_x} and F_{θ_y} represent the residuals of the momentum equation (16), in the x and y directions, respectively.

2.2.3. 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

$$\begin{aligned} L = {} & MSE_b + MSE_F + MSE_{\psi_{\text{int}}} + MSE_{p_{\text{int}}} + MSE_{\nabla(\psi_{\text{int}})} \\ & + MSE_{\nabla(p_{\text{int}})}, \end{aligned} \tag{24}$$

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 (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 2, is written as

$$MSE_b(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x}, \boldsymbol{y})_u^i\}_{i=1}^{N_b}) = \frac{1}{N_b}\sum_{i=1}^{N_b} |(u, v)^i - (u, v)_{\theta_\Omega}^i|_2^2; \tag{25}$$

$$MSE_F(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x}, \boldsymbol{y})_F^i\}_{i=1}^{N_F}) = \frac{1}{N_F}\sum_{i=1}^{N_F} |F_{\theta_\Omega}(x, y)_F^i|_2^2; \tag{26}$$

$$MSE_\psi(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x}, \boldsymbol{y})_{\text{int}}^i\}_{i=1}^{N_{\text{int}}}) = \sum_{\forall\,\Omega^+}\left(\frac{1}{N_{\text{int}}}\sum_{i=1}^{N_{\text{int}}} |(\psi)_{\theta_\Omega}(x, y)_{\text{int}}^i - (\psi)_{\theta_{\Omega^+}}(x, y)_{\text{int}}^i|_2^2\right); \tag{27}$$

$$MSE_p(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x}, \boldsymbol{y})_{\text{int}}^i\}_{i=1}^{N_{\text{int}}}) = \sum_{\forall\,\Omega^+}\left(\frac{1}{N_{\text{int}}}\sum_{i=1}^{N_{\text{int}}} |(p)_{\theta_\Omega}(x, y)_{\text{int}}^i - (p)_{\theta_{\Omega^+}}(x, y)_{\text{int}}^i|_2^2\right); \tag{28}$$

$$MSE_{\nabla(\psi)}(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x}, \boldsymbol{y})_{\text{int}}^i\}_{i=1}^{N_{\text{int}}}) = \sum_{\forall\,\Omega^+}\left(\frac{1}{N_{\text{int}}}\sum_{i=1}^{N_{\text{int}}} |(\nabla(\psi))_{\theta_\Omega}(x, y)_{\text{int}}^i - (\nabla(\psi))_{\theta_{\Omega^+}}(x, y)_{\text{int}}^i|_2^2\right); \tag{29}$$

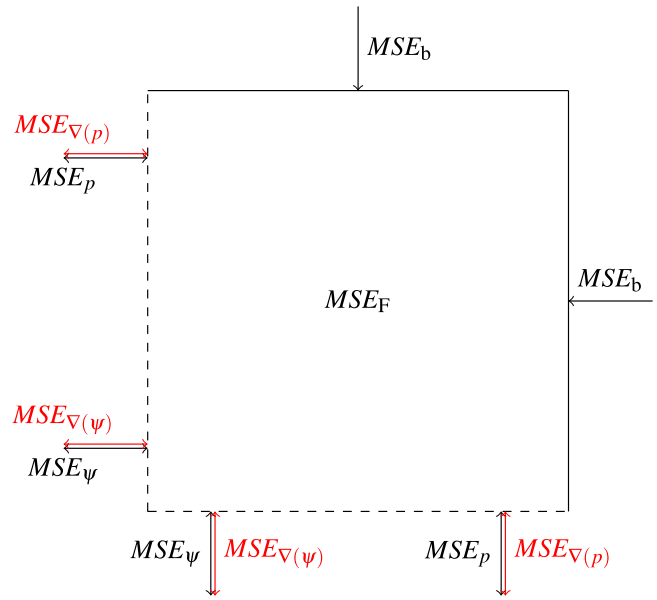


Figure 2. A sample subdomain Ω_i. Dashed lines show interfaces with adjacent subdomains.

$$MSE_{\nabla(p)}(\boldsymbol{\theta}_\Omega, \{(\boldsymbol{x}, \boldsymbol{y})^i_{\text{int}}\}_{i=1}^{N_{\text{int}}}) = \sum_{\forall \Omega^+} \left(\frac{1}{N_{\text{int}}} \sum_{i=1}^{N_{\text{int}}} |(\nabla(p))_{\theta_\Omega}(x, y)^i_{\text{int}} - (\nabla(p))_{\theta_{\Omega^+}}(x, y)^i_{\text{int}}|_2^2 \right), \tag{30}$$

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 [21].

2.2.4. 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.

2.3. DD-PINN with TRF

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 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 [8], in which transformer networks were incorporated inside the FC-NN, we adapt the DD-PINN and propose a new

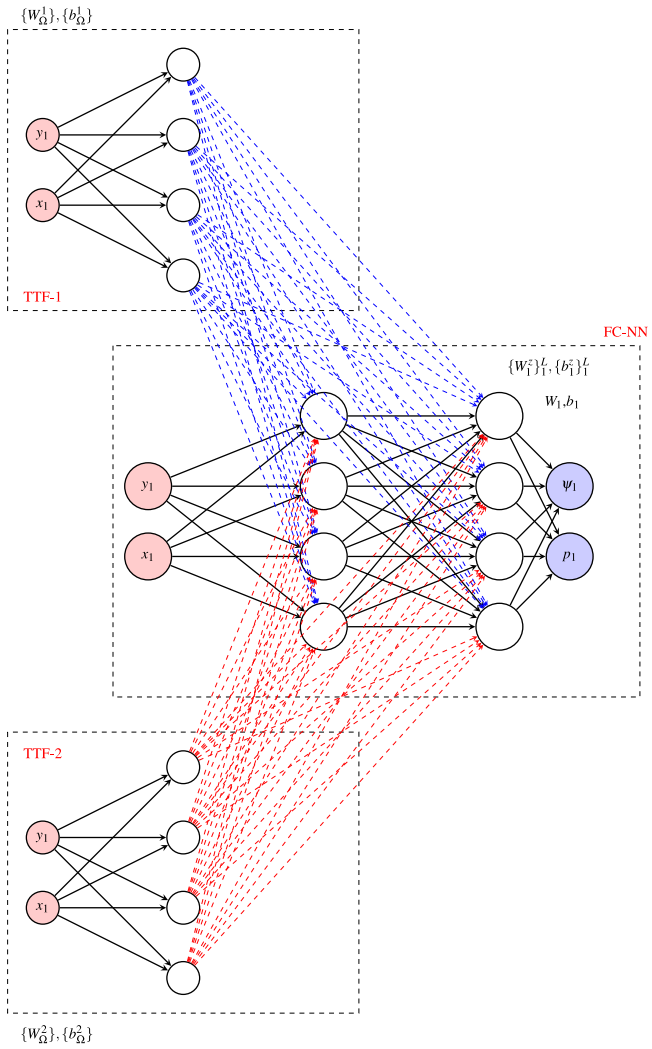


Figure 3. Schematic of the improved FC-NN with transformer networks at each subdomain.

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

$$\boldsymbol{U}_\Omega = \sigma(\boldsymbol{X}\boldsymbol{W}^1 + \boldsymbol{b}^1), \quad \boldsymbol{V}_\Omega = \sigma(\boldsymbol{X}\boldsymbol{W}^2 + \boldsymbol{b}^2), \tag{31}$$

$$\boldsymbol{H}_\Omega^1 = \sigma(\boldsymbol{X}\boldsymbol{W}^{z,1} + \boldsymbol{b}^{z,1}), \tag{32}$$

$$\boldsymbol{Z}_\Omega^k = \sigma(\boldsymbol{H}^k\boldsymbol{W}^{z,k} + \boldsymbol{b}^{z,k}), \quad k = 1,...,L, \tag{33}$$

$$\boldsymbol{H}_\Omega^{k+1} = (1 - \boldsymbol{Z}^k) \odot \boldsymbol{U}_\Omega + \boldsymbol{Z}^k \odot \boldsymbol{V}_\Omega, \quad k = 1,...,L, \tag{34}$$

$$\boldsymbol{f}_{\Omega_\theta}(\boldsymbol{x}) = \boldsymbol{H}_\Omega^{(L+1)}\boldsymbol{W} + \boldsymbol{b}, \tag{35}$$

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

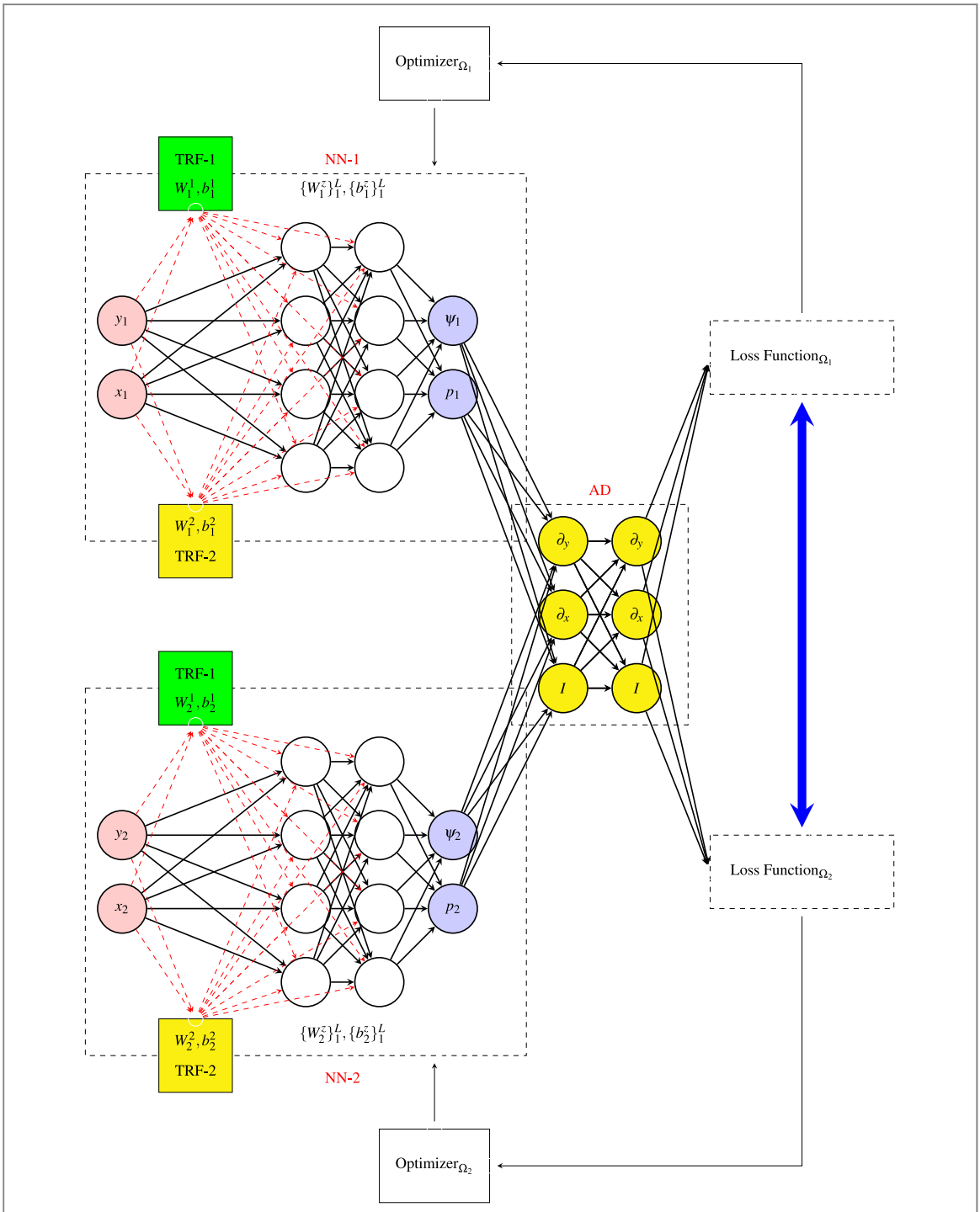


Figure 4. Schematic of the DD-PINN for a case with two subdomains. The blue arrow represents a link between subdomains through interface constraints.

$$\boldsymbol{\theta}_\Omega = \{\boldsymbol{W}_\Omega^1,\ \boldsymbol{b}_\Omega^1,\ \boldsymbol{W}_\Omega^2,\ \boldsymbol{b}_\Omega^2,\ (\boldsymbol{W}_\Omega^{z,l},\ \boldsymbol{b}_\Omega^{z,l})_{l=1}^L,\ W_\Omega,\ b_\Omega\} \tag{36}$$

$$\boldsymbol{\theta} = \{\theta_1,\ \theta_2\ ,...,\ \theta_{N_s}\}, \tag{37}$$

where N_s is the number of subdomains. A schematic of the DD-PINN illustrated through figure 4.

2.3.1. 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 relationships 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. 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 [34], and NN initialization follows the Glorot method [35]. TensorFlow [36] 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.
- **Viscous Burgers Equation**
- To validate the accuracy of the DD-PINN

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 (16)to (19)), is considered. The Reynolds number is $Re = 100$. The problem should then converge to a steady-state solution [29]. 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.

Table 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}

3.1.1. 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) [1] 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 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 1 and figure 5. In figure 5(a) which corresponds to the case with only essential constraints (No.1 in table 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 (figures 5(b) and (c)). The error is improved accordingly as presented in table 1. The most accurate case includes gradients of both the stream function $\psi(x, y)$ and the pressure field $p(x, y)$ (figure 5(e) and case 5 in table 1). This is analogous to using the H^1 norm of the error instead of L^2 in the MSE in the loss function

$$\|\psi\|^2_{H^1} = \|\psi\|^2_{L^2} + \|\nabla\psi\|^2_{L^2}. \tag{38}$$

$$\|P\|^2_{H^1} = \|P\|^2_{L^2} + \|\nabla P\|^2_{L^2}. \tag{39}$$

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.

$$\|\psi\|_{L^2}(\Omega) = \left(\int_\Omega |\psi^2| dx\right)^{\frac{1}{2}} \tag{40}$$

$$\|P\|_{L^2}(\Omega) = \left(\int_\Omega |P^2| dx\right)^{\frac{1}{2}} \tag{41}$$

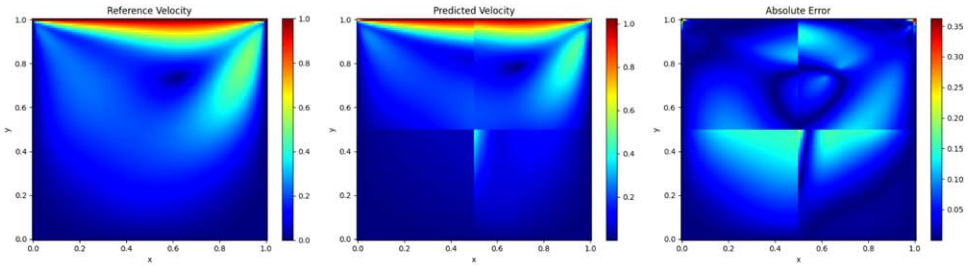
However, differentiability extends the concept of continuity by requiring the existence of well-defined derivatives. Smooth solutions to PDE are often characterized by high degrees of differentiability, allowing for the calculation of gradients. H^1 norms provide a way to measure the differentiability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

$$\|\psi\|_{H^1}(\Omega) = \left(\int_\Omega |\nabla\psi|^2 dx\right)^{\frac{1}{2}} \tag{42}$$

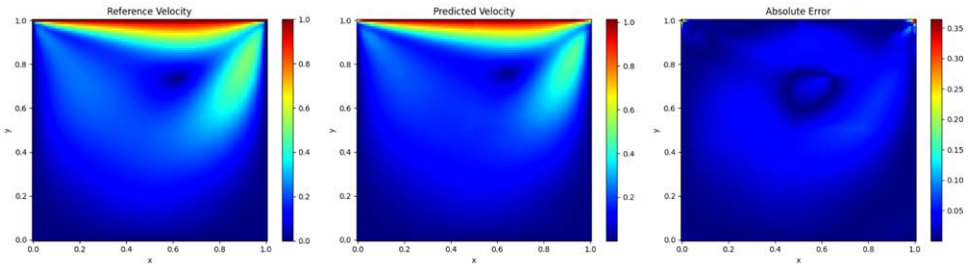
$$\|P\|_{H^1}(\Omega) = \left(\int_\Omega |\nabla P|^2 dx\right)^{\frac{1}{2}}. \tag{43}$$

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

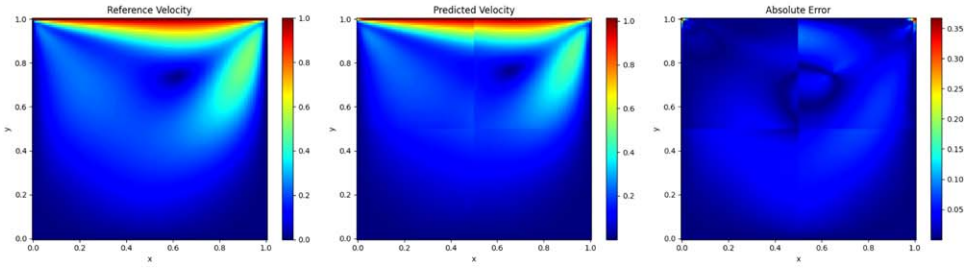
DD-PINN extends the concept of basic domain decomposition techniques, which typically involve constraints on the average solution using the L^2 norm, to include constraints on the H^1 norm within loss terms This extension significantly enhances the smoothness and regularity of solutions. Figure 6 confirms this concept. While using the L^2 norm of MSE in loss functions can preserve solution continuity (figure 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 (figures 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.



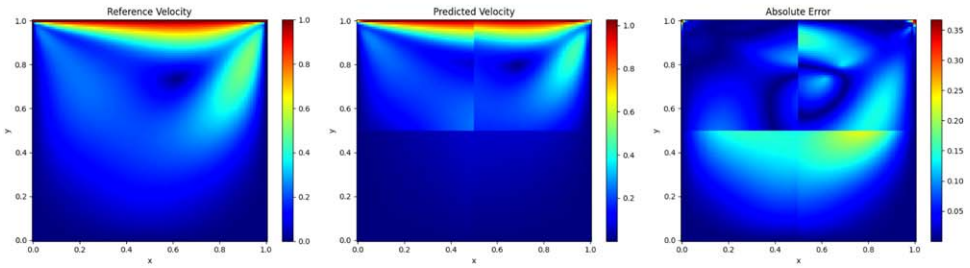
(a) case 1 of Table I



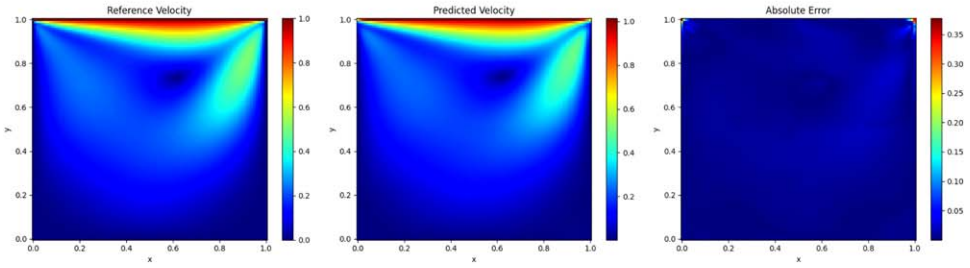
(b) case 2 of Table I



(c) case 3 of Table I



(d) case 4 of Table I

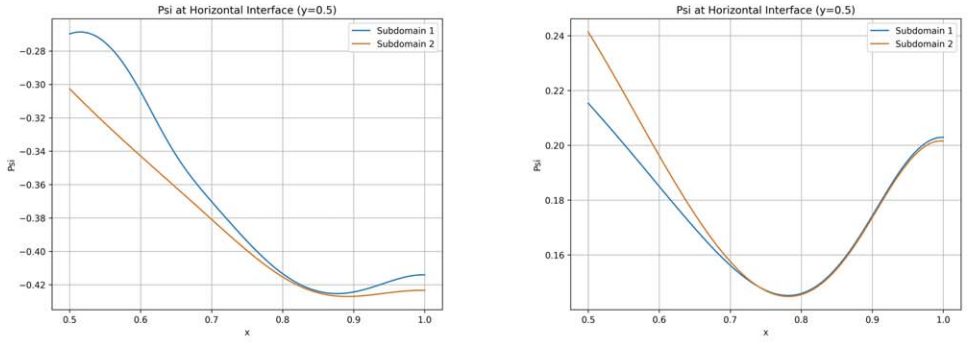


(e) case 5 of Table I

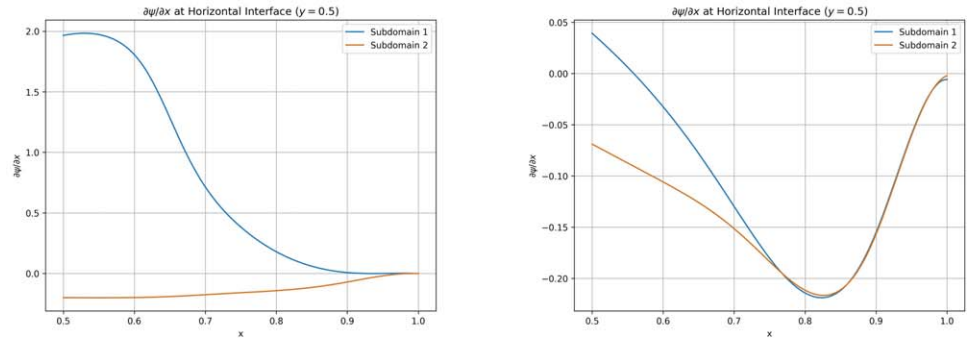
Figure 5. LDC problem: velocity fields.

Smoothness often leads to more predictable behavior of the neural network, making it easier to optimize. The higher accuracy of DD-PINN in table 1, in case number 5 where H^1 norms of loss terms are employed, provides further confirmation of this concept.

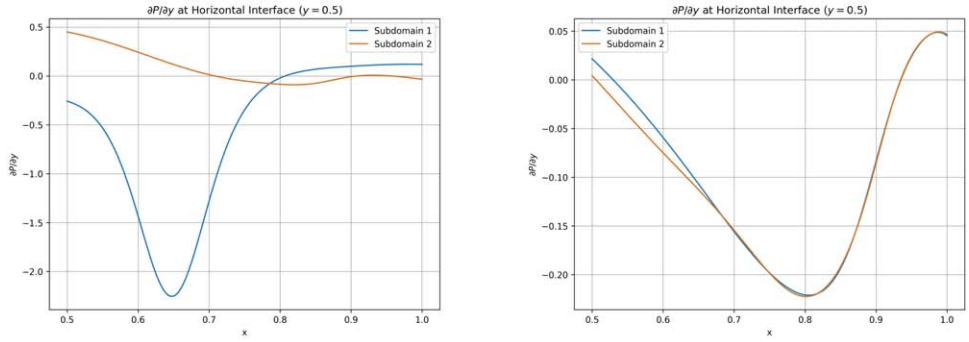
Finally, it is worth mentioning that including the residuals of the momentum equation at the interfaces significantly increases training times (figure 5(d)). Each computation of the corresponding loss term requires several extra passes through the computational graph due to the involvement of several first and second-order derivatives in the PDE residual for each interface. This makes optimization more challenging and convergence less likely.



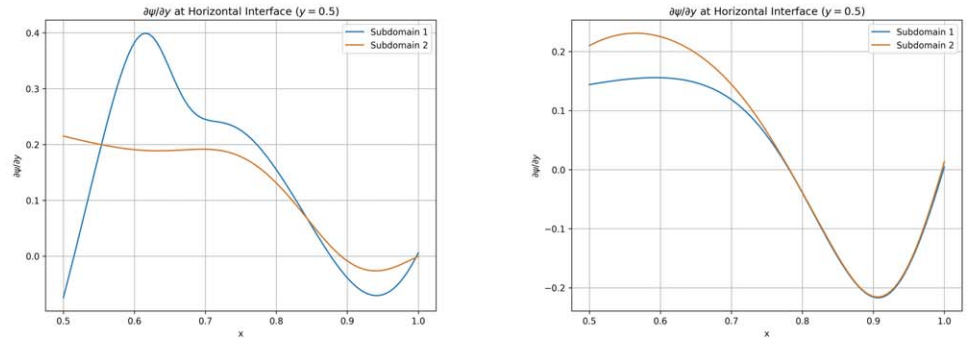
(a) Continuity of ψ at horizontal interface



(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)

Figure 6. LDC problem: Solution regularity analysis; The left column depicts results obtained using the baseline domain decomposition of PINN, whereas the right column shows those derived from the DD-PINN, employing the H^1 norm of the loss terms.

3.1.2. 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,

Table 2. DD-PINN versus 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}

TRF facilitate 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.

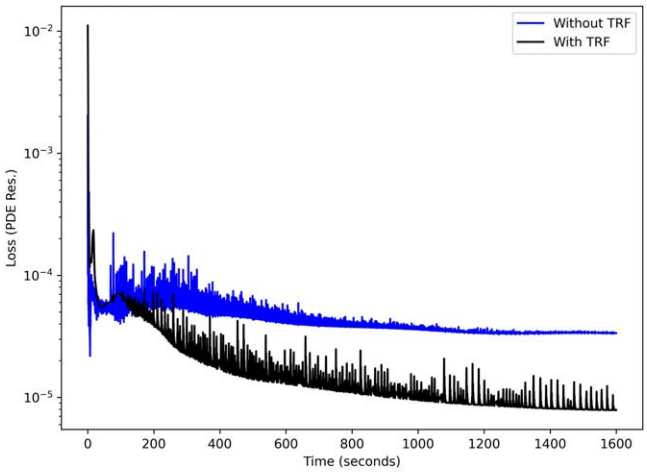
The results of the comparison between the DD-PINN with and without inclusion of two transformer networks show that the use of TRF leads to better accuracy, within a specific time as shown in table 2. By conducting this comparison, we aimed to discern the specific contribution of TRF to the model's performance 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. Quantitative data showing the loss decay rate is presented in figure 7. As shown, convergence is achieved faster in the case with TRFs. Finally, figure 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 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.

3.1.3. Combined contribution

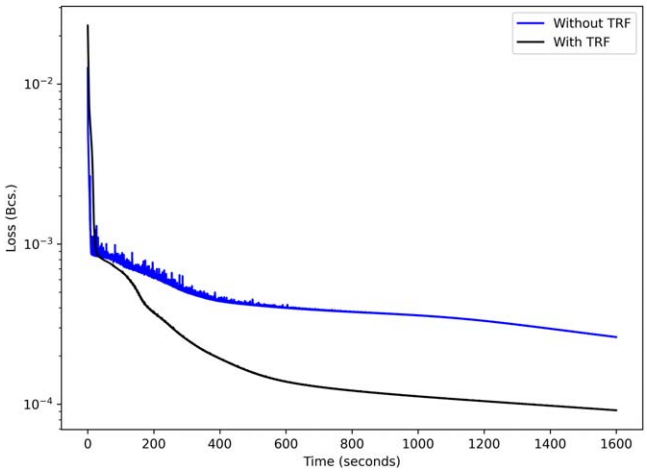
In sections 3.1.1 and 3.1.2, the contributions of DD-PINN and TRF were examined. To ensure accuracy, section 3.1.1 focused on DD-PINN without TRF for a fair comparison. Similarly, in section 3.1.2, DD-PINN with TRF was compared against DD-PINN without TRF to assess the specific contribution of TRF. This part explores the combined improvement achieved by integrating both components, particularly in terms of convergence rate, against baseline domain-decomposed PINN. As shown in figure 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 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 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.



(a) Residual loss



(b) BCs loss

Figure 7. Loss decay rate.

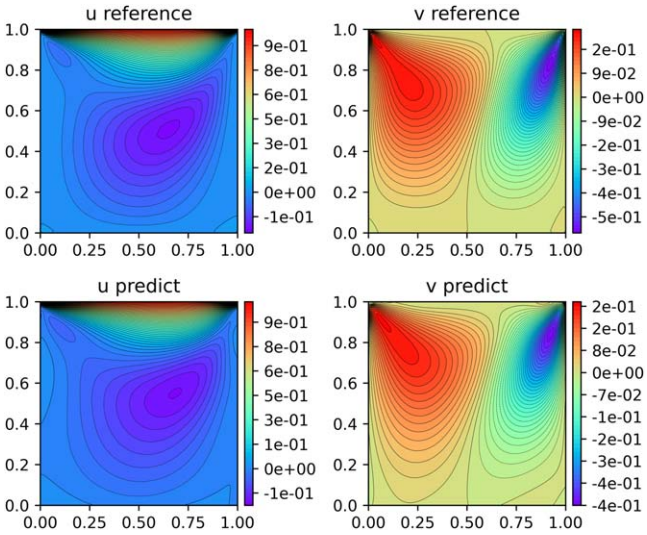
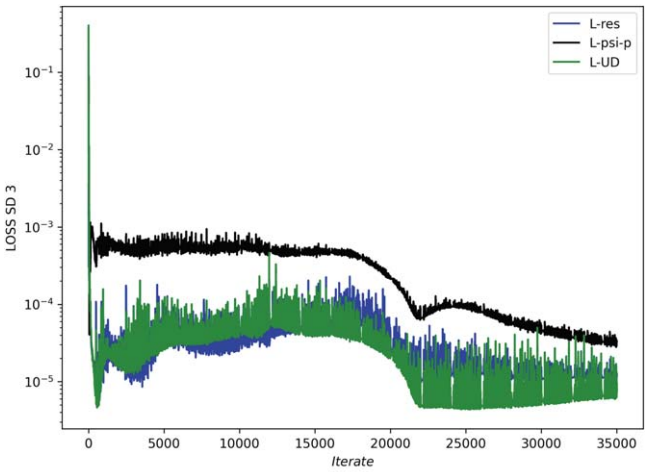
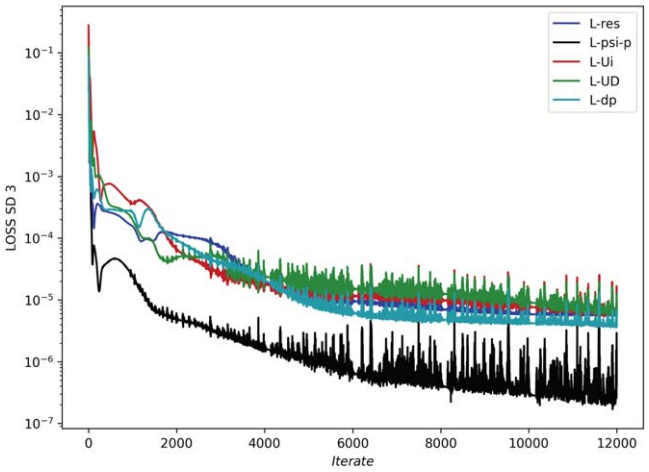


Figure 8. Velocity components contours: from top to bottom: reference solution, and DD-PINN with TRF.



(a) Baseline domain decomposition in PINN



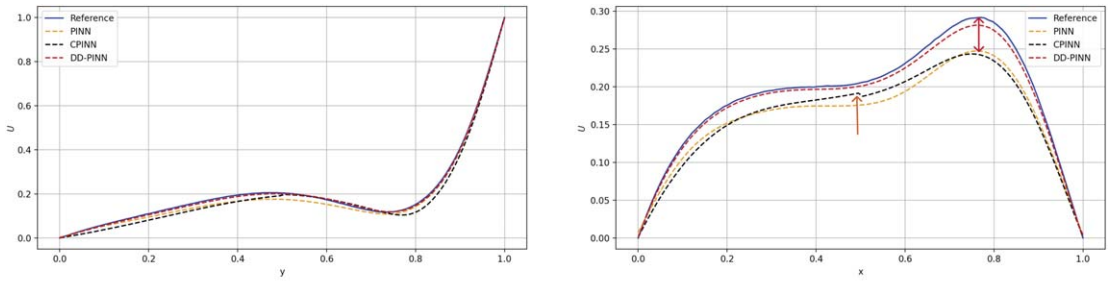
(b) DD-PINN with sub-network TRF

Figure 9. Convergence rate analysis in a subdomain: Contribution of DD-PINN with sub-network TRF compared with basic decomposed-domain PINN (CPINN.

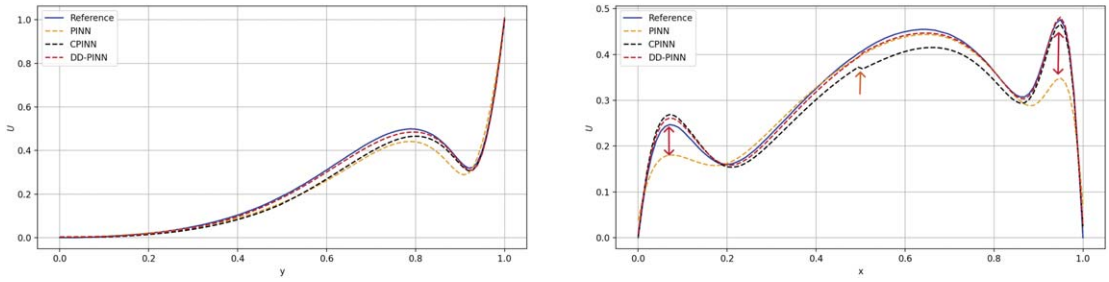
3.2. 2D cylinder wake

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 (16)to (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 [3] 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 [3, 37], 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 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 12 and the color legend in figure 13(b) it becomes evident that the left subdomain exhibits a higher approximation error. Consequently, we enforce the



(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

Figure 10. Numerical validation for lid-driven cavity: solution evaluation against reference solution and comparison against the baseline PINN and recent related works (CPINN).

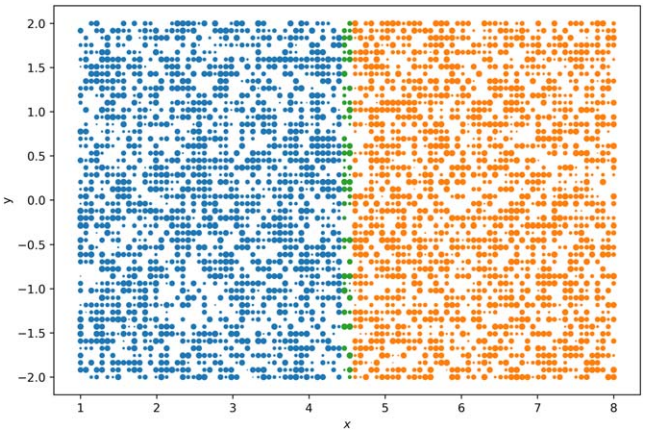


Figure 11. 2D cylinder wake problem domain: blue and orange points represent the training and collocation points in the left and right subdomains and the green points correspond to the interface points.

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 (figures 12 and 13(b), the subdomain at the right), by preserving 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,

$$L = MSE_u + MSE_{\mathrm{F}} + MSE_{P_{\mathrm{int}}}. \tag{44}$$

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 4). This technique not only enhances the accuracy of velocity approximation but also reduces the

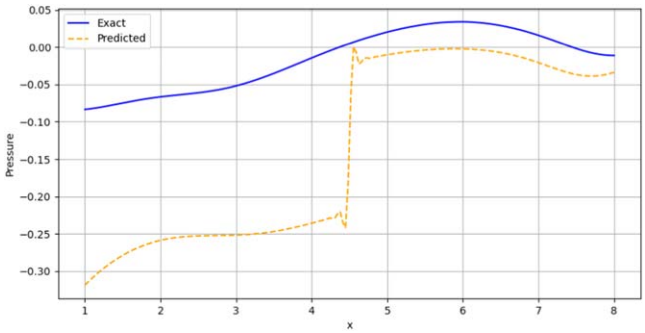


Figure 12. Dominant subdomain: analysis of subdomains in terms of solution accuracy.

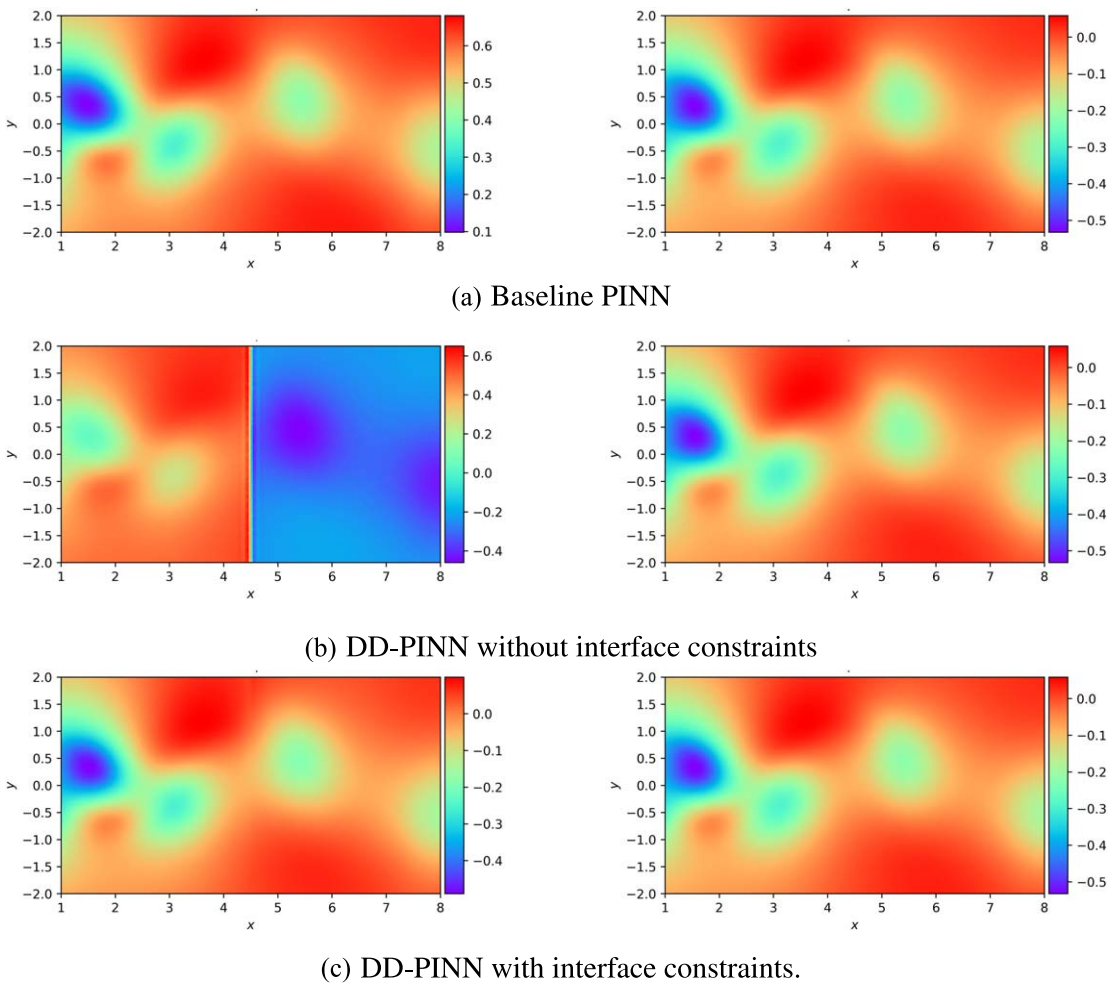


Figure 13. 2D cylinder wake problem; Pressure fields. Left and right columns show approximated and reference pressure fields, respectively.

magnitude differences in the pressure field between the reference and the approximated values, as given in table 3.

Numerical results (pressure magnitude) in figure 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 and figure 13 also confirm this conclusion. Additionally, velocity contours at the snapshot $t = 10$s are shown in figure 15, demonstrating a good agreement between the reference and approximated solutions. As represented in figure 15(b), the DD-PINN was able to capture the asymmetrical

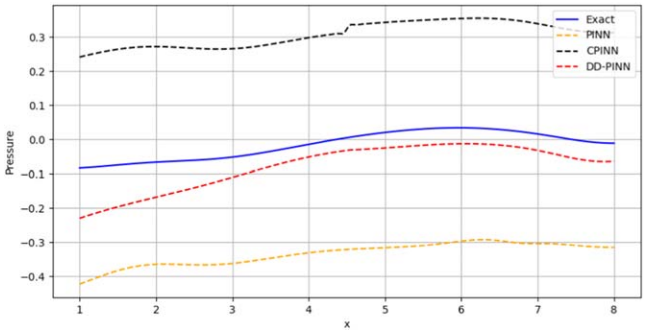
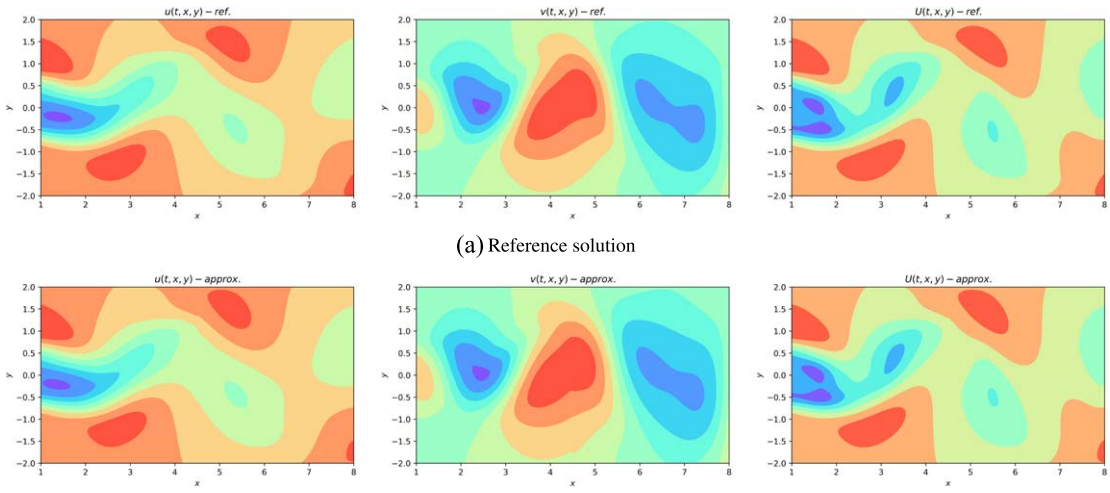


Figure 14. Numerical validation for 2D Cylinder Wake: solution evaluation against reference solution and comparison against the baseline PINN and recent related works (CPINN).



(a) Reference solution

(b) DD-PINN

Figure 15. 2D cylinder wake problem; velocity fields. From the left to the right: velocity components at x direction, y direction and velocity magnitude.

Table 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}	$(9) \times 35 + 7 \times 30$

swirling vortices caused by vortex shedding with slightly better accuracy than the baseline PINN (as shown in table 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:

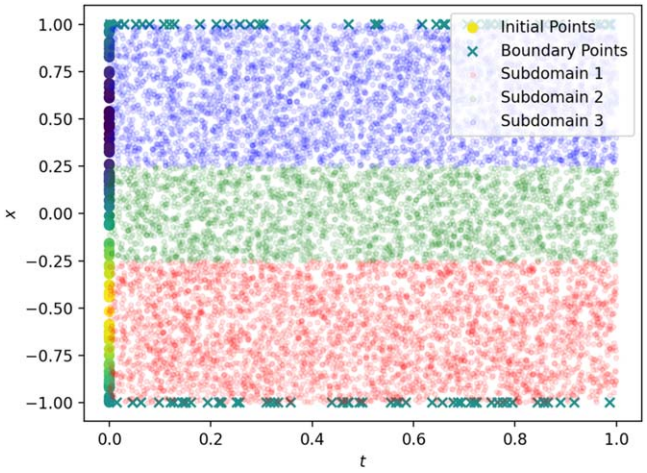


Figure 16. The problem domain: one-dimensional viscous Burgers equation.

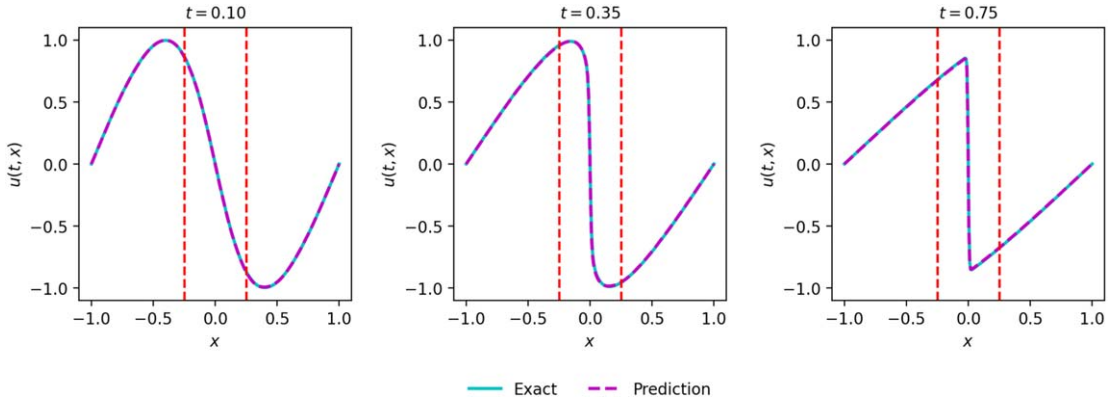


Figure 17. The DD-PINN solution and comparison of exact and predicted solutions at $t = 0.10$, $t = 0.35$, and $t = 0.75$ for the viscous Burgers equation. The red dashed lines indicate the locations of the interfaces.

$$u_t + uu_x = \nu u_{xx}, \qquad x \in [-1,\ 1],\ t \in [0,\ 1] \tag{45}$$

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 [38], 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 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, 11 000, 15 000]. The simulation 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 16, emphasizing the discretization of the domain into three subdomains. Figure 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 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 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 (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}

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.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Author declarations

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

Conflict of interest

The authors have no conflicts to disclose.

ORCID iDs

Amirhossein Khademi https://orcid.org/0009-0004-2290-1480

References

[1] Baydin A G, Pearlmutter B A, Radul A A and Siskind J M 2018 Automatic differentiation in machine learning: a survey *Journal of Marchine Learning Research* **18** 1–43 http://jmlr.org/papers/v18/17-468.html
[2] Raissi M, Perdikaris P and Karniadakis G E 2017 Physics informed deep learning (part i): data-driven solutions of nonlinear partial differential equations arXiv:1711.10561
[3] Raissi M, Perdikaris P and Karniadakis G E 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686–707

[4] Raissi M, Yazdani A and Karniadakis G E 2020 Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations *Science* **367** 1026–30
[5] Steinfurth B and Weiss J 2024 Assimilating experimental data of a mean three-dimensional separated flow using physics-informed neural networks *Phys. Fluids* **36**
[6] Jin X, Cai S, Li H and Karniadakis G E 2021 Nsfnets (Navier-Stokes flow nets): physics-informed neural networks for the incompressible navier-stokes equations *J. Comput. Phys.* **426** 109951
[7] Zhu Y, Kong W, Deng J and Bian X 2024 Physics-informed neural networks for incompressible flows with moving boundaries *Phys. Fluids* **36**
[8] Wang S, Teng Y and Perdikaris P 2021 Understanding and mitigating gradient flow pathologies in physics-informed neural networks *SIAM J. Sci. Comput.* **43** A3055–81
[9] Cao Z, Liu K, Luo K, Cheng Y and Fan J 2023 Efficient optimization design of flue deflectors through parametric surrogate modeling with physics-informed neural networks *Phys. Fluids* **35**
[10] Peng J-Z, Hua Y, Li Y-B, Chen Z-H, Wu W-T and Aubry N 2023 Physics-informed graph convolutional neural network for modeling fluid flow and heat convection *Phys. Fluids* **35**
[11] Chen W, Gao P and Stinis P 2024 Physics-informed machine learning of the correlation functions in bulk fluids *Phys. Fluids* **36**
[12] Zhang S, Deng J, Li X, Zhao Z, Wu J, Li W, Wang Y-G and Jeng D-S 2024 Solving the one dimensional vertical suspended sediment mixing equation with arbitrary eddy diffusivity profiles using temporal normalized physics-informed neural networks *Phys. Fluids* **36**
[13] Mahmoudabadbozchelou M, Karniadakis G E and Jamali S 2022 nn-pinns: non-newtonian physics-informed neural networks for complex fluid modeling *Soft Matter* **18** 172–85
[14] Lei G, Ma N, Sun B, Mao K, Chen B and Zhai Y 2023 Physics-informed neural networks for solving nonlinear bloch equations in atomic magnetometry *Phys. Scr.* **98** 085010
[15] Ye Y, Liu X, Li Y, Fan H and Zhang H 2023 Deep neural network method for solving the fractional burgers-type equations with conformable derivative *Phys. Scr.* **98** 065214
[16] Baleanu D, Karaca Y, Vázquez L and Macías-Díaz J E 2023 Advanced fractional calculus, differential equations and neural networks: Analysis, modeling and numerical computations *Phys. Scr.* **98** 110201
[17] Mao Z, Jagtap A D and Karniadakis G E 2020 Physics-informed neural networks for high-speed flows *Comput. Meth. Appl. Mech. Eng.* **360** 112789
[18] Li K, Tang K, Wu T and Liao Q 2019 D3m: a deep domain decomposition method for partial differential equations *IEEE Access* **8** 5283–94
[19] Kharazmi E, Zhang Z and Karniadakis G E 2019 Variational physics-informed neural networks for solving partial differential equations arXiv:1912.00873
[20] Kharazmi E, Zhang Z and Karniadakis G E 2021 hp-vpinns: variational physics-informed neural networks with domain decomposition *Comput. Meth. Appl. Mech. Eng.* **374** 113547
[21] Jagtap A D, Kharazmi E and Karniadakis G E 2020 Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems *Comput. Meth. Appl. Mech. Eng.* **365** 113028
[22] Jagtap A D and Karniadakis G E 2021 Extended physics-informed neural networks (xpinns): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations *Commun. Comput. Phys.* **28** 2002–41
[23] Shukla K, Jagtap A D and Karniadakis G E 2021 Parallel physics-informed neural networks via domain decomposition *J. Comput. Phys.* **447** 110683
[24] Aulakh D J S, Beale S B and Pharoah J G 2022 A generalized framework for unsupervised learning and data recovery in computational fluid dynamics using discretized loss functions *Phys. Fluids* **34**
[25] Zhou W, Miwa S and Okamoto K 2024 Advancing fluid dynamics simulations: a comprehensive approach to optimizing physics-informed neural networks *Phys. Fluids* **36**
[26] Qin S-M, Li M, Xu T and Dong S-Q 2023 Am-gpinn algorithm and its application in a variable-coefficient resonant nonlinear schrödinger equation *Phys. Scr.* **98** 025219
[27] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Ł and Polosukhin I 2017 Attention is all you need *Advances in Neural Information Processing Systems* **30**
[28] Wang S, Wang H, Seidman J H and Perdikaris P 2022 Random weight factorization improves the training of continuous neural representations arXiv:2210.01274
[29] Bruneau C-H and Saad M 2006 The 2d lid-driven cavity problem revisited *Comput. Fluids* **35** 326–48
[30] Hornik K, Stinchcombe M and White H 1989 Multilayer feedforward networks are universal approximators *Neural Netw.* **2** 359–66
[31] Berman D S, Buczak A L, Chavis J S and Corbett C L 2019 A survey of deep learning methods for cyber security *Information* **10** 122
[32] Cuomo S, Di Cola V S, Giampaolo F, Rozza G, Raissi M and Piccialli F 2022 Scientific machine learning through physics-informed neural networks: where we are and what's next *J. Sci. Comput.* **92** 88
[33] Mishra S and Molinaro R 2022 Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes *IMA J. Numer. Anal.* **42** 981–1022
[34] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization arXiv:1412.6980
[35] Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proceedings of the XIII International Conference on Artificial Intelligence and Statistics (JMLR Workshop and Conference Proceedings* pp 249–56
[36] Abadi M *et al* 2016 Tensorflow: a system for large-scale machine learning *Osdi* vol 16 (Savannah) pp 265–83
[37] Cheng C and Zhang G-T 2021 Deep learning method based on physics informed neural network with resnet block for solving fluid flow problems *Water* **13** 423
[38] Basdevant C, Deville M, Haldenwang P, Lacroix J, Ouazzani J, Peyret R, Orlandi P and Patera A 1986 Spectral and finite difference solutions of the burgers equation *Comput. Fluids* **14** 23–41