Physics-Informed Neural Networks for Fluid Mechanics: A Collocation Point-Based Approach

CH5710 Course Project

Shivansh Gupta
3rd Year B.Tech
Department of Chemical Engineering
ch22b011@smail.iitm.ac.in

Shrey Malik
3rd Year B.Tech
Department of Chemical Engineering
ch22b004@smail.iitm.ac.in

Om Mineeyar
3rd Year B.Tech
Department of Chemical Engineering
ch22b014@smail.iitm.ac.in

Bharath Sajeev
3rd Year B.Tech
Department of Chemical Engineering
ch22b012@smail.iitm.ac.in

Akanksh CJ
3rd Year B.Tech
Department of Chemical Engineering
ch22b010@smail.iitm.ac.in

Ojas Phadake
3rd Year B.Tech
Department of Chemical Engineering
ch22b007@smail.iitm.ac.in

ABSTRACT

This work explores the application of Physics-Informed Neural Networks (PINNs) to solve complex fluid dynamics problems, including the Burgers' equation, the Navier-Stokes equations for 2D flow, Falkner–Skan boundary layers, and turbulent boundary layers under zero-pressure gradient (ZPG) conditions. The study demonstrates the ability of PINNs to efficiently predict velocity and pressure fields with minimal training data by leveraging physics-based constraints. Results show that PINNs can achieve high accuracy in modeling both laminar and turbulent flow behaviors, with predictions closely matching reference solutions and DNS data. The findings highlight the robustness and computational efficiency of PINNs, making them a promising tool for solving challenging fluid dynamics problems in engineering applications.

I. INTRODUCTION

In recent years, machine-learning (ML) methods have started to play a revolutionary role in many scientific disciplines. The emergence of various architectures, e.g., convolutional neural networks (CNNs) and long short-term memory (LSTM), has led to the development of a variety of deep-learning (DL)-based frameworks for modeling complex physical systems by accounting for spatiotemporal dependencies in the predictions. Deep learning applications have been demonstrated in many scientific and engineering disciplines, e.g., astronomy, climate modeling, solid mechanics, chemistry, and sustainability. Fluid mechanics & reaction engineering has been one of the active research topics for the development of innovative ML-based approaches. The contribution of ML in fluid mechanics & reaction engineering problems is mainly in the contexts of data-driven digital twinning, uncertainty quantification, and surrogate modeling. More recently, exploiting the universal-approximation property of neural networks for solving complex partial differential equation (PDE) systems has brought attention, aiming to provide efficient solvers that approximate the solution.

Deep learning provides powerful modeling approaches for data-rich domains such as vision, language, and speech. However, learning interpretable and generalizable models is still challenging, especially for domains with limited data available such as complex physical systems. Purely data-driven approaches based on deep learning demand large datasets for training that may not be available for many scientific problems. Moreover, these models generally do not take into account the physical constraints and may fit the observational data very well but lack in complying with the underlying physical principles. Therefore, integrating governing physical laws and domain knowledge into the model training may lead to more accurate and robust models. The domain knowledge can act as an informative prior for teaching the model about the physical or mathematical understanding of the system besides the observational data. Physics-informed neural networks (PINNs), introduced by Raissi et al.[1], are deep-learning-based frameworks able to integrate data and physical laws in the form of governing partial differential equations (PDEs) for learning. PINNs have been demonstrated to be well suited for the solution of forward and inverse problems related to several different types of PDEs. PINNs have been used to simulate vortex-induced vibrations[2] and tackle ill-posed inverse fluid mechanics problems[3]. Moreover, PINNs have been employed for super-resolution and denoising of 4D-flow magnetic resonance imaging (MRI)[4] and prediction of near-wall blood flow from sparse data[5] among other applications[6-10]. Jin et al.[11] showed the applicability of PINNs for the simulation of turbulence directly, where good agreement was obtained between the direct numerical simulation (DNS) results and the PINNs simu-

lation results. Cai et al.[12] proposed a method based on PINNs to infer the full continuous three-dimensional velocity and pressure fields from snapshots of three-dimensional temperature fields obtained by tomographic background-oriented Schlieren (Tomo-BOS) imaging. Recently, Eivazi and Vinuesa[13] applied PINNs for super-resolution of flow-field data both in time and space from a limited set of noisy measurements without having any high-resolution reference data. A detailed discussion on the prevailing trends in embedding physics into machine-learning algorithms and diverse applications of PINNs can be found in the work by Karniadakis et al[14].

In this study, we employ Physics-Informed Neural Networks (PINNs) for solving the Navier–Stokes (NS) equations for incompressible flows and the Burger's Equation for unsteady case. Traditional solvers require the introduction of modeling assumptions to close the system. We introduced an alternative approach to tackle this problem by using the information from a few data examples and the underdetermined system of equations for the training of a neural network that solves the system of equations. In particular, we use the data on the domain boundaries along with the NS and Burger's equations that guide the learning process toward the solution. The spatial coordinates are the inputs of the model, and the mean-flow quantities, i.e., velocity components, pressure, etc. are the outputs. Automatic differentiation (AD)[15] is applied to differentiate the outputs with respect to the inputs to construct the required equations. Only the data on the domain boundaries are considered the training dataset for the supervised-learning part, while a set of points inside the domain together with the points on the boundaries are used to calculate the residual of the governing equations, which acts as the unsupervised loss. The Reynolds number is set through the governing equations. We take mean-flow quantities obtained from CFD simulation obtained through Ansys Fluent Software as the reference.

This article is organized as follows: in Sec. II, we provide an overview of the physics-informed neural networks and discuss the theoretical background; in Sec. III, we discuss the application of PINNs for solving various equations with results; and finally in Sec IV, we provide a summary and the conclusions of the study.

II. METHODOLOGY

In this section, we provide the methodological background of Physics-Informed Neural Networks (PINNs) and their application for solving partial differential equations. We also discuss the implementation of PINNs for solving various cases & their respective equations. The goal of PINNs is to integrate the information from the governing physical laws in the form of partial differential equations into the training process of a deep neural network so that the solution can be approximated by only having a limited set of training samples, e.g., initial and boundary conditions.

A PINN model comprises a multi-layer perceptron (MLP) that maps the coordinates to the solution and a so-called residual network, which calculates the residual of the governing equations.

A. PINNs

Physics-informed neural networks (PINNs) represent a novel approach to integrating data and physical laws within a unified learning framework. By embedding prior knowledge from mathematical models, such as partial differential equations (PDEs), directly into machine learning algorithms, PINNs achieve accurate, interpretable, and robust predictions, even when faced with noisy, incomplete, or missing data. They excel in solving ill-posed and inverse problems, scaling efficiently for complex, high-dimensional systems. PINNs bridge the gap between the classical physics-driven paradigm and modern data-driven approaches, making them invaluable for diverse real-world applications.

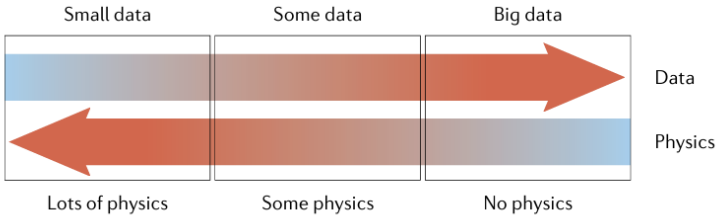


Figure 1: Diagram illustrating the interplay between physics and data across different regimes.

This integration is visually represented in Fig. 1, which highlights three regimes of problem-solving based on the extent of physical knowledge and data availability:

- **Small Data with Lots of Physics (Left):** The classical paradigm assumes well-known physics, where limited data, such as initial and boundary conditions, suffices to solve the governing PDEs.
- **Intermediate Regime (Some Data, Some Physics):** The most common real-world scenario lies in the center, where physics is partially understood (e.g., conservation laws with missing constitutive relationships), and scattered data is used to infer missing parameters or unknown terms in the equations. PINNs shine in this regime by seamlessly combining partial knowledge of physics with data.
- **Big Data with No Physics (Right):** In this regime, large datasets are available, but the underlying physical laws are unknown. Data-driven methods, like operator regression, are used to uncover new physics directly from the data.

The diagram's horizontal and vertical axes depict the transition from "Lots of Physics" to "No Physics" and "Small

Data" to "Big Data," respectively. The gradient arrows indicate the balance between physics and data in different regimes. PINNs effectively operate across this spectrum, leveraging both physics and data to solve problems ranging from small-data scenarios to big-data challenges.

B. Mathematics Behind Physics-Informed Neural Networks (PINNs)

PINNs integrate the rigor of physics-based modeling with the adaptability of deep learning by embedding physical laws directly into the neural network's learning process. The approach minimizes a comprehensive loss function, ensuring that the network's predictions respect governing physical laws, boundary conditions, and, optionally, observational data. This section details the mathematical framework and the deep learning principles underpinning PINNs.

Loss Function Composition

The total loss function L_{total} is a combination of multiple terms, each addressing a specific aspect of the problem:

$$L_{\text{total}} = L_{\text{physics}} + L_{\text{data}} + L_{\text{boundary}} + L_{\text{initial}}$$

- **Physics Loss (L_{physics})**: This term ensures that the network predictions satisfy the governing partial differential equations (PDEs). By leveraging *automatic differentiation*, the network computes derivatives of its outputs with respect to inputs. These derivatives are substituted into the PDE, and the residual error is penalized. For a PDE of the form $\mathcal{N}[u] = 0$, the physics loss is defined as:

$$L_{\text{physics}} = \frac{1}{N_p} \sum_{j=1}^{N_p} \left(\mathcal{N}[\hat{u}(x_j, t_j; \theta)]\right)^2$$

  where N_p represents the number of collocation points used to enforce the PDE.

- **Data Loss (L_{data})**: This term aligns the network's predictions with available observational data points. For measured data $(x_i, t_i, u_i^{\text{true}})$, the data loss is:

$$L_{\text{data}} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left(\hat{u}(x_i, t_i; \theta) - u_i^{\text{true}}\right)^2$$

  Here, $\hat{u}(x_i, t_i; \theta)$ is the network's prediction, and u_i^{true} is the observed value. In cases where no observational data is available (e.g., solving a PDE from scratch) as in our case, this term may be excluded.

- **Boundary Loss (L_{boundary})**: Ensures that the network predictions satisfy specified boundary conditions, such as fixed or periodic constraints. For a boundary condition $u(x, t) = u_{\text{boundary}}$, the loss is formulated as:

$$L_{\text{boundary}} = \frac{1}{N_b} \sum_{k=1}^{N_b} \left(\hat{u}(x_k, t_k; \theta) - u_k^{\text{boundary}}\right)^2$$

- **Initial Loss (L_{initial})**: For time-dependent problems, this term ensures consistency with initial conditions, penalizing deviations at $t = 0$:

$$L_{\text{initial}} = \frac{1}{N_i} \sum_{l=1}^{N_i} \left(\hat{u}(x_l, 0; \theta) - u_l^{\text{initial}}\right)^2$$

Deep Learning Perspective: How PINNs Work

1. **Neural Network Architecture**: PINNs use fully connected feedforward neural networks with parameters θ (weights and biases). The input layer accepts the spatial and temporal variables (x, t), and the output layer predicts the dependent variable $\hat{u}(x, t; \theta)$. Activation functions like ReLU or $\tanh$ introduce non-linearity, enabling the network to model complex relationships.

2. **Forward Pass and Automatic Differentiation**: During the forward pass, the network computes $\hat{u}(x, t; \theta)$ for all inputs. Using *automatic differentiation*, the required spatial and temporal derivatives (e.g., $\frac{\partial \hat{u}}{\partial x}$, $\frac{\partial^2 \hat{u}}{\partial t^2}$) are computed efficiently. These derivatives are substituted into the PDE to calculate the physics loss.

3. **Loss Backpropagation**: The total loss L_{total} is backpropagated through the network to compute gradients of the loss with respect to the network parameters θ. These gradients are used to update the parameters via an optimization algorithm, such as Adam or stochastic gradient descent (SGD).

4. **Training Objective**: The network iteratively minimizes L_{total}, balancing the contributions of data consistency, physics adherence, and boundary/initial condition enforcement. Over iterations, the network learns a function $\hat{u}(x, t; \theta)$ that satisfies both the PDE and any additional constraints.

C. PINNs for Burger's Equation

Burgers' equation is a fundamental partial differential equation and convection–diffusion equation which occurs in various areas of applied mathematics such as such as fluid mechanics, nonlinear acoustics, gas dynamics, and traffic flow. As a physical model we tried to simulate and compare our model for the Burgers equation. This equation is a very simple, yet non-linear and non-trivial, model equation that can lead to interesting shock formations. It represents a well-studied PDE, which (unlike Navier-Stokes) does not include any additional constraints such as conservation of mass. Hence, it's a very good starting point for experiments. It contains an advection term (motion / transport) and a diffusion term (dissipation due to the second law of thermodynamics). It is given by:

$$\frac{\partial u_x}{\partial t} + u \nabla u = \nu \nabla \cdot \nabla u_y$$

For our study, we considered a simplified one-dimensional form of Burgers' equation defined over the domain $\mathcal{R} = \{(x,t)\,|\,x \in [-1,1],\, t \in [0,1]\}$. The governing initial and boundary conditions were as follows:

- Initial Condition: $u(x,0) = -\sin(\pi x)$
- Boundary Conditions: $u(-1,t) = u(1,t) = 0 \quad \forall t \in [0,1]$

Our PINN architecture is designed to predict the solution $u(x,t)$ directly from spatial and temporal coordinates (x,t). The inputs to the neural network are the spatial and temporal points generated over a grid of 25×25 in the (x,t)-domain. Automatic differentiation is utilized to compute the derivatives required for the advection and diffusion terms in the equation.

The training process for the PINN model is detailed in the Results and Discussion section. For now, we focus on the setup and formulation of the problem, which leverages the inherent ability of PINNs to learn the solution by enforcing the physical laws embedded within the Burgers' equation.

This formulation allowed us to simulate the system efficiently, making it a robust benchmark to evaluate the effectiveness of PINNs for solving non-linear partial differential equations.

D. PINNs for Navier-Stokes Equation (2D Flow Between Parallel Plates)

The 2D Navier-Stokes equations are used to model and solve the velocity profile of fluid flow between infinite parallel plates. This setup is particularly relevant in understanding viscous fluid behavior and hydrodynamic principles. The flow domain, illustrated below in Fig. 2, comprises an entrance region where the fluid enters with a nearly uniform velocity profile.

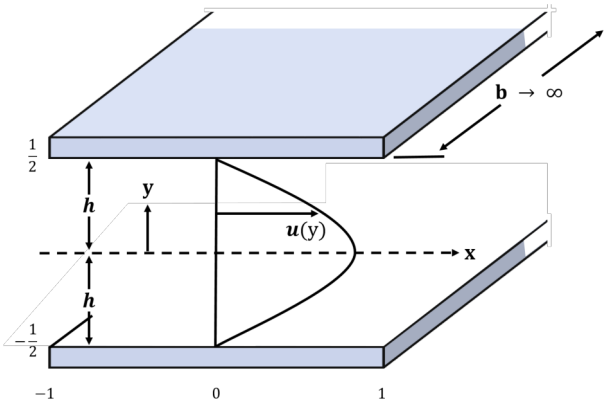


Figure 2: Diagram illustrating the fluid domain for 2D Flow Between Parallel Plates.

As the fluid progresses between the plates, viscous forces cause the velocity at the plate boundaries to become zero, adhering to the no-slip boundary condition. This results in the formation of a boundary layer, leading to gradual evolution in the velocity profile along the length of the plates (x). Beyond the hydrodynamic entrance region (also called the entrance length), the velocity profile reaches a steady state, becoming invariant with respect to x.

Problem Properties

The physical properties of the system are defined as follows:

$$\rho = 1\,\text{kg/m}^3, \quad \mu = 1\,\text{N·s/m}^2, \quad D = 2h = 1\,\text{m},$$
$$L = 2\,\text{m}, \quad u_{\text{in}} = 1\,\text{m/s}, \quad \nu = \frac{\mu}{\rho}.$$

2D Navier-Stokes Equations and Boundary Conditions

The governing 2D steady-state Navier-Stokes equations for velocity components $u(x,y)$ and $v(x,y)$ are given by:

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{1}{\rho}\frac{\partial p}{\partial x} - \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0,$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{1}{\rho}\frac{\partial p}{\partial y} - \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0,$$

The governing 2D continuity equation for velocity components $u(x,y)$ and $v(x,y)$ is given by:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.$$

Boundary Conditions

1. No-Slip Condition at Plate Boundaries: At the top and bottom plate boundaries ($y = \pm\frac{D}{2}$), the velocity components are zero:
$$u(x,y) = 0, \quad v(x,y) = 0, \quad \text{at } y = \pm\frac{D}{2}.$$
2. Inlet Boundary: At the inlet ($x = -1$), the horizontal velocity is equal to the inlet velocity u_{in}, and the vertical velocity is zero:
$$u(-1,y) = u_{\text{in}}, \quad v(-1,y) = 0.$$
3. Outlet Boundary: At the outlet ($x = 1$), the pressure is zero, and the vertical velocity is zero:
$$p(1,y) = 0, \quad v(1,y) = 0.$$

PINN Formulation

For solving this problem using PINNs, the spatial and temporal coordinates (x,y) are used as inputs to a neural network that predicts the velocity components $u(x,y)$, $v(x,y)$, and pressure $p(x,y)$. Automatic differentiation computes the derivatives required in the Navier-Stokes equations, which are enforced as constraints in the loss function.

The loss function consists of: - A *physics loss* to enforce the Navier-Stokes equations. - *Boundary losses* to impose the no-slip and inlet/outlet conditions.

The detailed training process and results of the PINN model for this setup are discussed in the *Results and Discussion* section.

E. PINNs for Falkner–Skan boundary layer (FSBL) and ZPG turbulent boundary layer

In this study, we employ PINNs for solving the Reynolds-averaged Navier–Stokes (RANS) equations for incompressible turbulent flows without introducing any specific model or assumptions for turbulence. The loss of information in the averaging process of the RANS equations leads to an underdetermined system of equations, and traditional solvers require the introduction of modeling assumptions to close the system.

We introduced an alternative approach to tackle this problem by using data from a few examples along with the underdetermined system of equations to train a neural network that solves the system. Specifically, we use data on the domain boundaries (including Reynolds-stress components) along with the RANS equations to guide the learning process towards the solution.

In this framework, spatial coordinates serve as inputs to the model, and the outputs are the mean-flow quantities such as velocity components, pressure, and Reynolds-stress components. Automatic differentiation (AD) is employed to differentiate the outputs with respect to the inputs to construct the RANS equations. Only the data on the domain boundaries are considered as the training dataset for supervised learning, while a set of points inside the domain, as well as points on the boundaries, are used to calculate the residual of the governing equations, which acts as the unsupervised loss. The Reynolds number is set through the governing equations. We use mean-flow quantities obtained from Direct Numerical Simulation (DNS) or well-resolved Large Eddy Simulation (LES) of canonical turbulent flow cases as the reference.

A schematic for the RANS equations is as follows:

$$U\frac{\partial U}{\partial x}+V\frac{\partial U}{\partial y}+\frac{1}{\rho}\frac{\partial P}{\partial x}-\nu\left(\frac{\partial^2 U}{\partial x^2}+\frac{\partial^2 U}{\partial y^2}\right)+\frac{\partial \overline{u^2}}{\partial x}+\frac{\partial \overline{uv}}{\partial y}=0$$

$$U\frac{\partial V}{\partial x}+V\frac{\partial V}{\partial y}+\frac{1}{\rho}\frac{\partial P}{\partial y}-\nu\left(\frac{\partial^2 V}{\partial x^2}+\frac{\partial^2 V}{\partial y^2}\right)+\frac{\partial \overline{uv}}{\partial x}+\frac{\partial \overline{v^2}}{\partial y}=0$$

$$\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}=0$$

In a general two-dimensional setup, the spatial coordinates (x and y) are the inputs to a Multilayer Perceptron (MLP), and the outputs are the mean streamwise and wall-normal components of velocity (U and V, respectively), pressure (P), and Reynolds-stress components ($\bar{u}^2$, $\bar{u}\bar{v}$, and $\bar{v}^2$). Automatic differentiation is applied to differentiate outputs with respect to inputs and formulate the RANS equations, including continuity and momentum equations.

In our setup, only the data on the domain boundaries are used as the training dataset for supervised learning. The total loss function is the summation of the supervised loss and the residual of the governing equations, expressed as:

$$L = L_e + L_b$$

$$L_e = \frac{1}{N_e}\sum_{i=1}^{3}\sum_{n=1}^{N_e}|\epsilon_i^n|^2$$

$$L_b = \frac{1}{N_b}\sum_{n=1}^{N_b}\left|\mathbf{U}_b^n - \tilde{\mathbf{U}}_b^n\right|^2$$

Here, L_e and L_b represent the loss-function components corresponding to the residual of the RANS equations and the boundary conditions, respectively. N_e is the number of collocation points, and N_b is the number of boundary points. The residuals of the governing equations are calculated inside the domain as well as at the boundaries. We also consider weighting coefficients to balance the terms in the loss function and accelerate the convergence in the training process.

III. RESULTS AND DISCUSSION

This section presents a detailed exploration of the application of Physics-Informed Neural Networks (PINNs) to a series of challenging fluid dynamics problems. These include the Burgers' equation, Navier-Stokes equations for two-dimensional flow, laminar boundary layers, and turbulent boundary layers under zero-pressure gradient (ZPG) conditions. The results and discussions herein provide a holistic view of the robustness, efficiency, and versatility of PINNs in solving partial differential equations (PDEs) arising in fluid mechanics.

A. Burgers' Equation

The unsteady Burgers' equation, a fundamental PDE in fluid mechanics and nonlinear dynamics, was selected as the first test case. This equation is often employed as a benchmark due to its ability to encapsulate essential features of nonlinear convection-diffusion systems. For our experiments, the model inputs included the spatial coordinate x and time t, while the output was the velocity component $u(x,t)$. This simple yet powerful problem allowed us to test the predictive accuracy and computational efficiency of PINNs.

Our implementation leveraged the PyTorch framework, enabling seamless integration of automatic differentiation and model training. To optimize performance, extensive hyperparameter tuning was conducted. This included experimenting with network architectures ranging from 3 to 6 hidden layers and varying the number of neurons per layer across a wide range (20, 30, 40, 50, 64, and 128). Optimization strategies such as Adam, SGD, RMSprop, Adagrad, AdamW, and LBFGS were tested with learning rates ranging from 0.001 to 0.1.

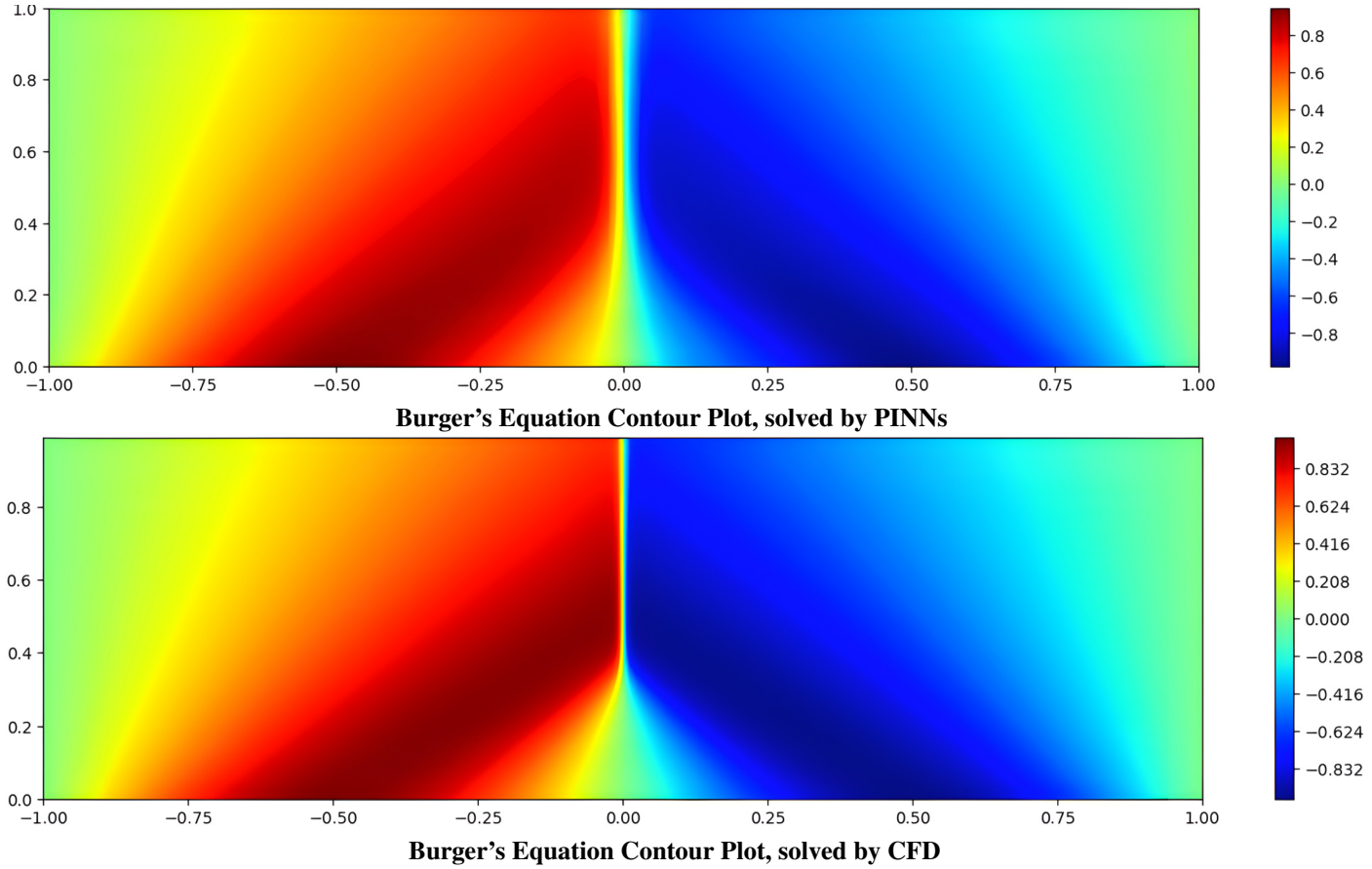


Figure 3: BG

To systematically evaluate the performance of different configurations, we utilized a 25×25 grid of spatial and temporal points as input data. Iterative grid search methods using the `itertools` library were employed to identify the optimal hyperparameters. The results, shown in Fig. 3, illustrate the contour plots of u predicted by the PINN compared to the reference solutions derived from analytical methods. The predictions demonstrated excellent agreement with minimal deviation, indicating that the PINN effectively captured the dynamics of the system.

The most optimal configuration was found to consist of 4 hidden layers with 64 neurons per layer, trained using the Adam optimizer with a learning rate of 0.005. This configuration balanced model complexity with computational efficiency, achieving convergence within a few seconds. Notably, the study underscores the adaptability of PINNs to varying geometries and boundary conditions, requiring only minimal adjustments to the input features and loss functions.

B. Navier-Stokes Equations for 2D Flow between Parallel Plates

To evaluate the capability of PINNs in solving multi-dimensional fluid flow problems, the Navier-Stokes equations for incompressible, steady-state, two-dimensional flow between parallel plates were considered. This canonical problem represents a simplified model of Couette or Poiseuille flow, depending on the imposed boundary conditions. Here, the PINN was tasked with predicting the velocity components u and v, along with the pressure field p, using spatial coordinates (x, y).

The model architecture was implemented using the DeepXDE library, a robust tool for defining and solving differential equations via PINNs. Architectures ranging from 3 to 5 hidden layers with neuron counts between 64 and 128 per layer were explored. In addition, a variety of weight initialization techniques, including Glorot uniform, He normal, Lecun uniform, and zero initialization, were tested to assess their impact on convergence. Optimizers such as Adam, SGD, RMSprop, Nadam, and Adagrad were employed with varying learning rates (10^{-3} and 10^{-4}) to identify the optimal configuration.

The results of this study are presented in Fig. 4, showcasing the contour plots of u, v, and p. These predictions are compared against high-resolution CFD data, demonstrating excellent agreement across the domain. A detailed comparison of the velocity profiles at the outlet against the analytical solution confirmed the model's accuracy. The fully developed velocity profile, expressed as:

$$u(y) = \frac{3V_{\text{avg}}}{2}\left[1 - \left(\frac{y}{h}\right)^2\right],$$

was reproduced with minimal error, as depicted in Fig. 4.

The results highlight the efficiency of PINNs in resolving flow features without requiring dense training data, owing to their physics-informed loss formulation.

C. Falkner–Skan Boundary Layer (FSBL)

The Falkner–Skan boundary layer problem models laminar boundary layer flow under the influence of an adverse pressure gradient, which is crucial for understanding flow separation and aerodynamic performance. The boundary condition is defined in terms of a similarity solution to the Navier–Stokes equations under boundary layer approximations. The specific parameters for this study have been outlined above, allowing for an investigation of Physics-Informed Neural Networks (PINNs) in predicting flow characteristics near separation conditions.

The results of the PINN simulation are illustrated in Fig. 5, showing contour plots of the predicted streamwise velocity component (U), wall-normal velocity component (V), and pressure field (P). Despite training solely on boundary data, the PINN accurately reproduced the interior solutions, demonstrating its ability to generalize and interpolate effectively. A comparison of the predicted velocity profiles with the reference solutions revealed excellent agreement, with minimal deviation observed across the domain.

In addition to the contour plots, the wall shear stress distribution was analyzed to assess the influence of the adverse pressure gradient on flow development. The PINN predictions captured the expected thinning of the boundary layer and the corresponding increase in shear stress near the wall. This highlights the model's capability to resolve subtle flow features that are crucial for characterizing aerodynamic performance under adverse conditions.

Overall, the Falkner–Skan case study highlights the robustness of PINNs in solving boundary layer problems. The ability to achieve high accuracy using only boundary data significantly reduces the need for dense datasets, making this approach highly scalable for more complex geometries and flow regimes.

D. Zero-Pressure Gradient (ZPG) Turbulent Boundary Layer

To evaluate the capabilities of PINNs in modeling complex flows, we analyzed the turbulent boundary layer under zero-pressure gradient (ZPG) conditions, a key problem in turbulence research with broad engineering relevance. The domain and conditions, including Reynolds numbers and reference DNS data, are as outlined above.

The governing equations—continuity and streamwise momentum—were incorporated into the PINN loss function, with outputs including mean streamwise velocity (U), wall-normal velocity (V), and shear Reynolds stress mean(uv). Inputs were the spatial coordinates x and y, allowing the model to capture turbulence statistics across the boundary layer.

The results of the PINN simulation are shown in Fig. 6, with contour plots of the predicted streamwise velocity (U), wall-normal velocity (V), and shear Reynolds stress ($\overline{uv}$) compared against DNS data. The relative L_2-norm errors for U, V, and $\overline{uv}$ were 1.02%, 4.25%, and 6.46%, respectively, indicating high accuracy despite the complexities of the turbulent regime.

To further validate the predictions, key turbulence statistics such as the shape factor ($H_{12} = \delta/h$) and the skin-friction coefficient ($c_f = 2\left(\frac{u_\tau}{U_\infty}\right)^2$) were computed from the PINN outputs. These quantities are critical for characterizing the boundary layer and its interaction with the wall. The predicted values closely matched the DNS data, confirming the model's ability to capture the essential physics of turbulent flows.

The near-wall region, which poses significant challenges due to steep velocity gradients, was analyzed using inner-scaled quantities. These included the velocity profile ($U^+ = U/u_\tau$) and the wall-normal coordinate ($y^+ = y/\ell$), where u_τ is the friction velocity and $\ell = \nu/u_\tau$ is the viscous length scale. The PINN successfully captured the logarithmic velocity profile in the overlap region and the linear behavior near the wall, demonstrating its effectiveness in modeling near-wall turbulence dynamics.

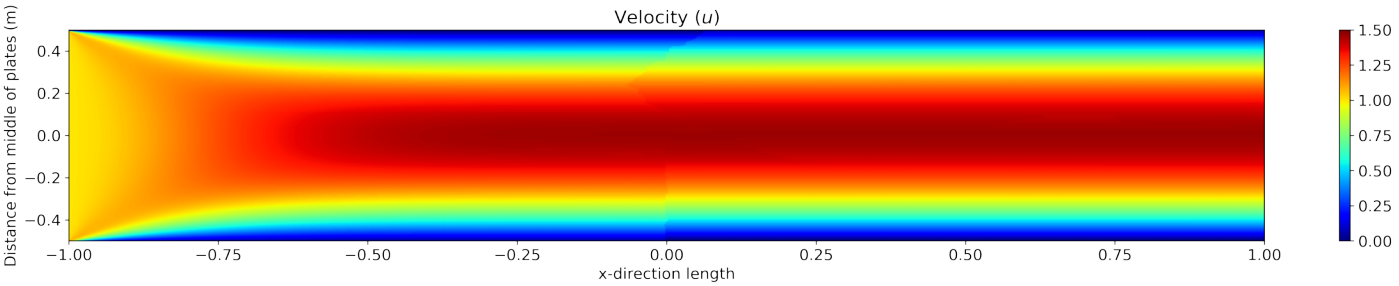
IV. SUMMARY AND CONCLUSIONS

This study demonstrated the effectiveness of Physics-Informed Neural Networks (PINNs) in solving complex fluid dynamics problems, including the Burgers' equation, Navier-Stokes equations for 2D flow, Falkner–Skan boundary layers, and turbulent boundary layers under zero-pressure gradient (ZPG) conditions.

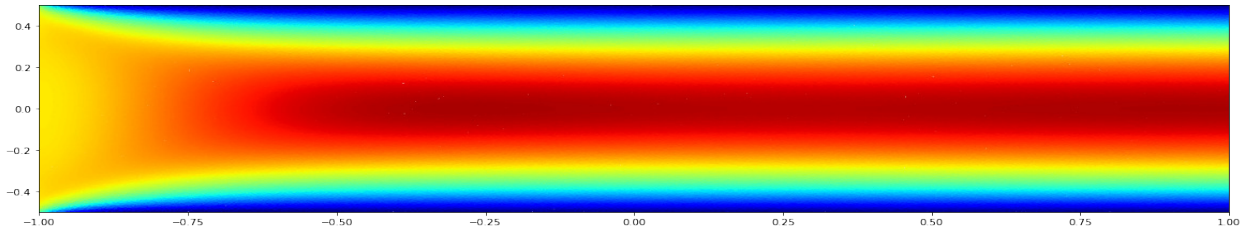
The Burgers' equation was successfully modeled with excellent agreement to reference solutions, using a PINN with 4 hidden layers and 64 neurons per layer. The Navier-Stokes problem showed accurate predictions for velocity and pressure in 2D flow, validating the model's ability to resolve fluid dynamics efficiently.

In the Falkner–Skan boundary layer problem, the PINN accurately predicted flow characteristics near separation conditions, requiring only boundary data. For the ZPG turbulent boundary layer, the model captured turbulence statistics and near-wall behavior with high accuracy, matching DNS data.

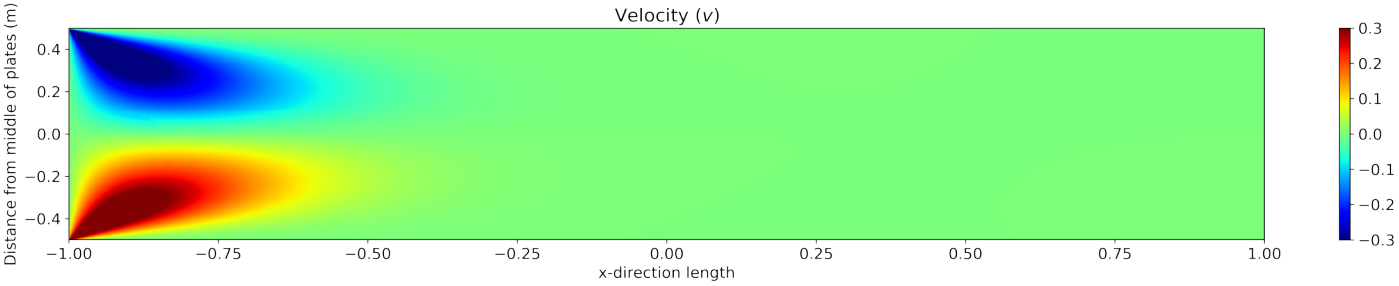
Overall, PINNs have proven to be robust and efficient for solving fluid dynamics problems, providing physically consistent solutions with minimal computational cost. Their ability to incorporate physical constraints directly into the learning process offers a promising avenue for addressing real-world engineering challenges in fluid mechanics.



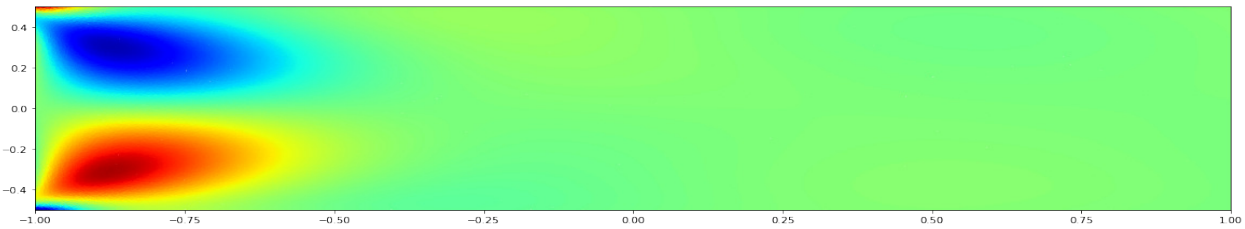
U-Velocity Contour Plot, obtained by CFD Simulation



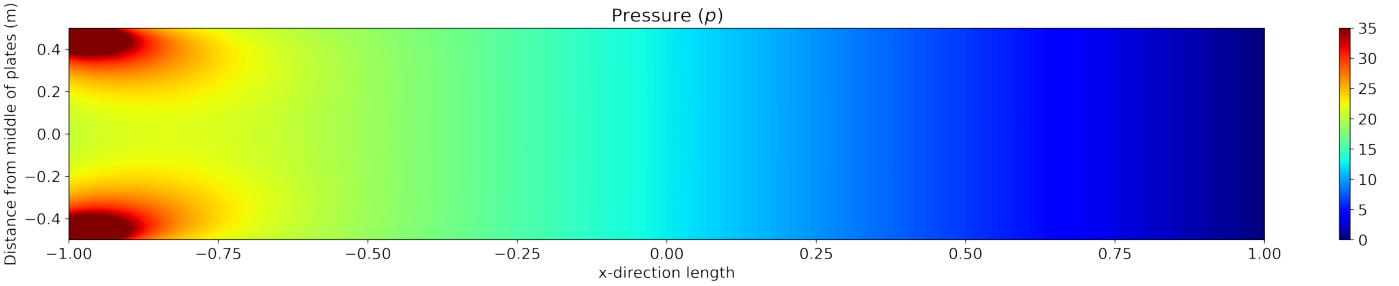
U-Velocity Contour Plot, obtained by PINNs



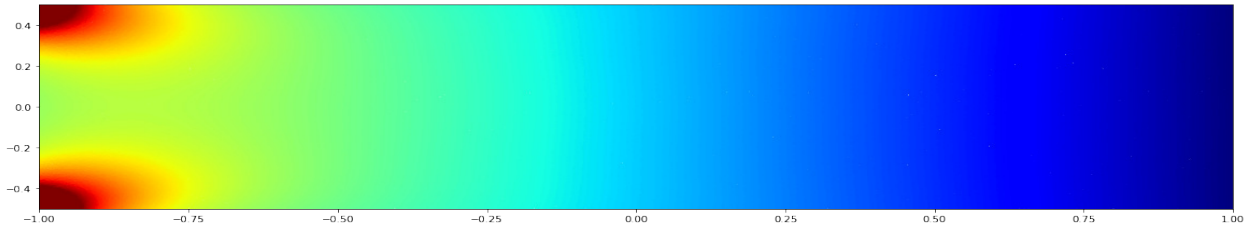
V-Velocity Contour Plot, obtained by CFD Simulation



V-Velocity Contour Plot, obtained by PINNs



Pressure Contour Plot, obtained by CFD Simulation



Pressure Contour Plot, obtained by PINNs

Figure 4: Contour Plots for Navier-Stokes Equations for 2D Flow between Parallel Plates

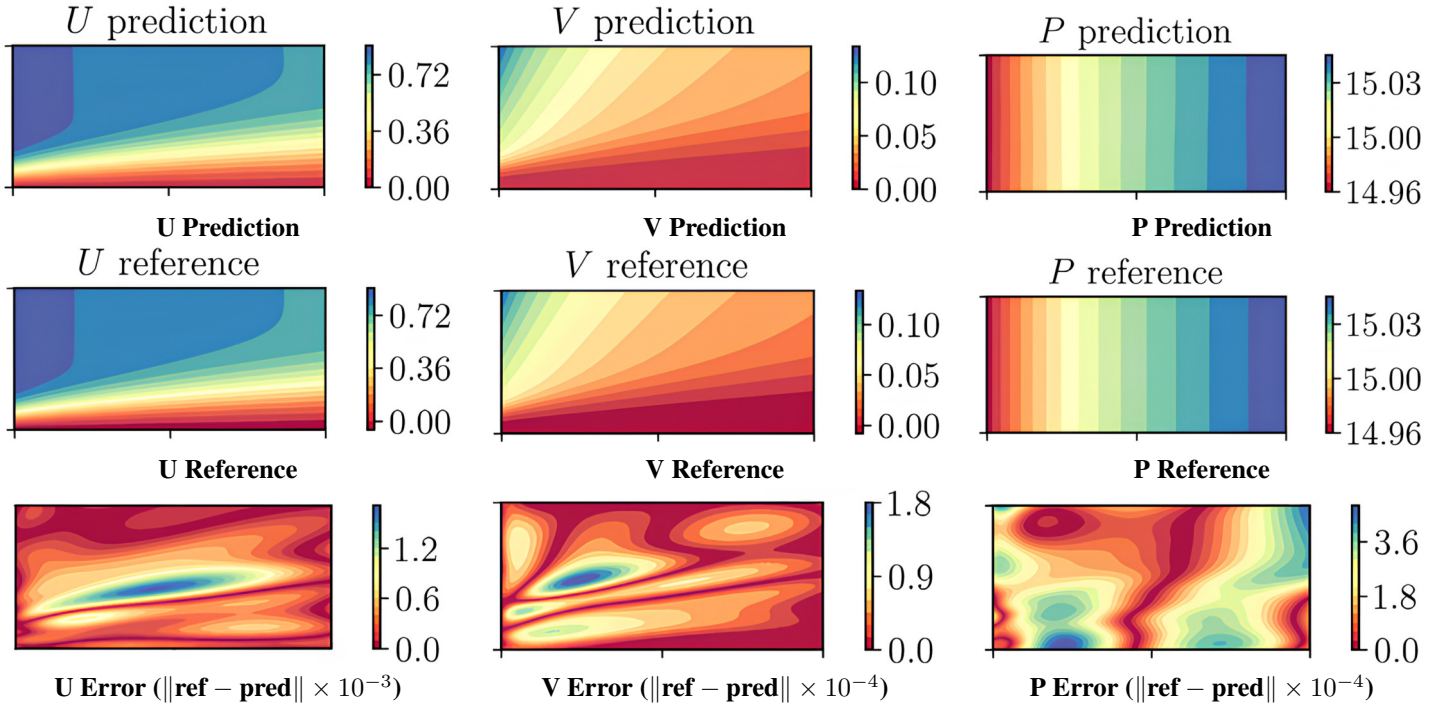


Figure 5: Simulation results(Contour Plots) of the Falkner–Skan boundary layer using PINNs in comparision with the reference data and the absolute error

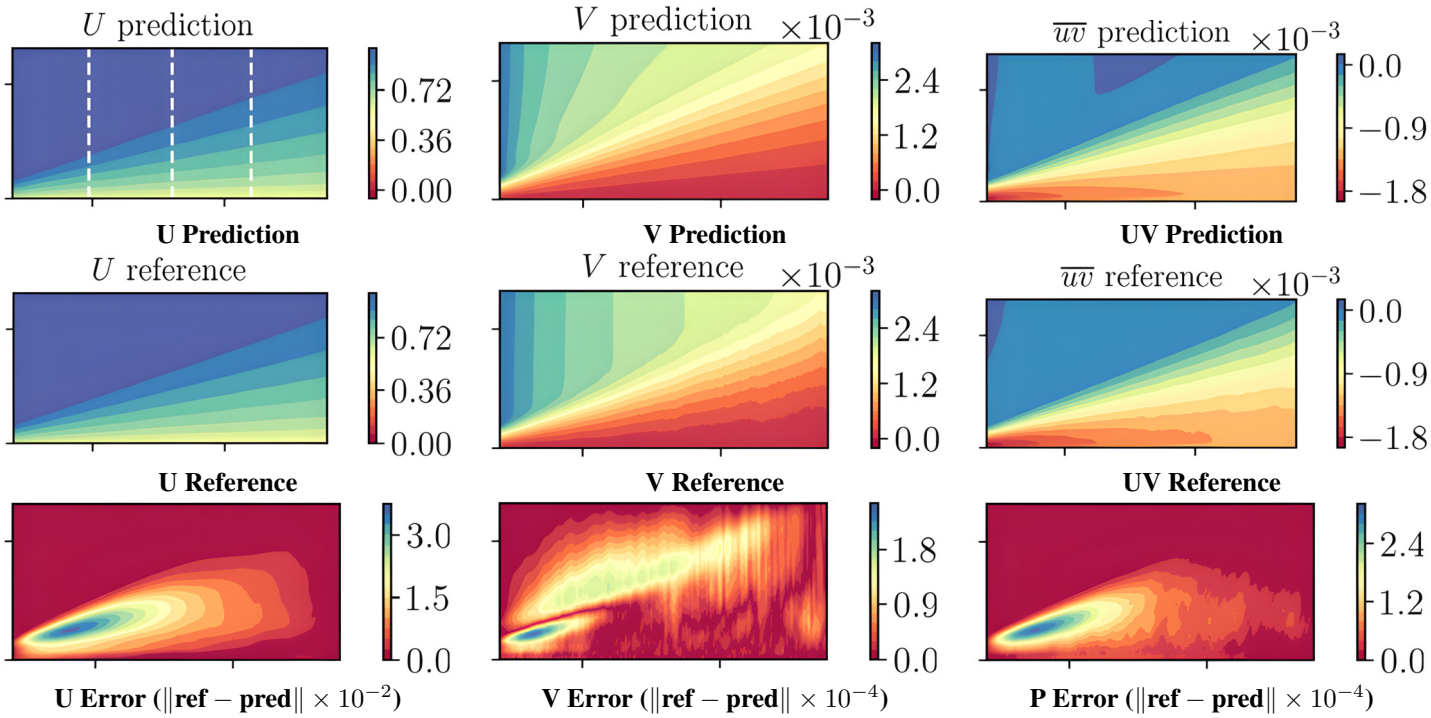


Figure 6: Simulation results(Contour Plots) and the absolute error for the ZPG turbulent boundary layer using PINNs in comparison with the reference data

ACKNOWLEDGMENTS

We extend our heartfelt gratitude to Prof. Himanshu Goyal for his invaluable guidance and instruction in the course *Applications of Machine Learning in Reaction Engineering*. The concepts taught during the course greatly enriched our understanding and enabled us to successfully build a PINN model and execute this project.

We also express our sincere thanks to the Teaching Assistants, Varun Kumar and Vamsi Pyla, for providing well-crafted case studies during tutorial sessions, which significantly enhanced our learning experience.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Shivansh Gupta: Conceptualization; Investigation; Methodology (equal); Software; Writing – Abstract, Introduction, Results & Discussions, Summary & Conclusions, References; Resources(Graphics & Figures); Proofreading(Primary); Git.
Om Mineeyar: Software; Writing: Methodology (equal).
Shrey Malik: Software; Writing: Methodology (equal).
Ojas Phadake: Writing: Methodology (equal); Proofreading(Secondary).
Bharath Sajeev: Writing: Methodology (equal).
Akanksh CJ: Writing: Methodology (equal).

DATA AVAILABILITY

As mentioned in the Methodology section, we did not use any training data and instead used the residual loss minimization method to train our model. For that we generated collocation points inside the domain and also points along the boundary of the 2D region. For comparison of the accuracy of our model, we simulated the system using Ansys Fluent and validated our model predictions.

CODE REPOSITORY

The code repository for this project can be found on GitHub: `Link`.

REFERENCES

1. 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," J. Comput. Phys. 378, 686-707 (2019). Link.
2. M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, "Deep learning of vortex-induced vibrations," J. Fluid Mech. 861, 119–137 (2019). Link.
3. M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," Science 367, 1026–1030 (2020). Link.
4. M. F. Fathi, I. Perez-Raya, A. Baghaie, P. Berg, G. Janiga, A. Arzani, and R. M. D'Souza, "Super-resolution and denoising of 4d-flow mri using physics-informed deep neural nets," Comput. Methods Prog. Biomed. 197, 105729 (2020). Link.
5. A. Arzani, J.-X. Wang, and R. M. D'Souza, "Uncovering near-wall blood flow from sparse data with physics-informed neural networks," Phys. Fluids 33, 071905 (2021). Link.
6. S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," Acta Mech. Sin. 37, 1727–1738 (2021). Link.
7. H. Wang, Y. Liu, and S. Wang, "Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network," Phys. Fluids 34, 017116 (2022). Link.
8. R. Qiu, R. Huang, Y. Xiao, J. Wang, Z. Zhang, J. Yue, Z. Zeng, and Y. Wang, "Physics-informed neural networks for phase-field method in two-phase flow," Phys. Fluids 34, 052109 (2022). Link.
9. 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," Int. J. Heat Fluid Flow 96, 109002 (2022). Link.
10. Y. Chen and L. Dal Negro, "Physics-informed neural networks for imaging and parameter retrieval of photonic nanostructures from near-field data," APL Photonics 7, 010802 (2022). Link.
11. 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," J. Comput. Phys. 426, 109951 (2021). Link.
12. S. Cai, Z. Wang, F. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis, "Flow over an espresso cup: Inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks," J. Fluid Mech. 915, A102 (2021). Link.
13. H. Eivazi and R. Vinuesa, "Physics-informed deep-learning applications to experimental fluid mechanics," arXiv:2203.15402 (2022). Link.
14. G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nat. Rev. Phys. 3, 422–440 (2021). Link.
15. A. G. Baydin, B. A. Pearlmutter, A. Andreyevich Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *arXiv:1502.05767* (2018). Link.