PINN PROJECT
Incompressible N-S equations for low speed

Members:

313551012 - Solomonovich Tal
205906308 - Matar Hedi
316278001 - Yaniv Abramov

Date:

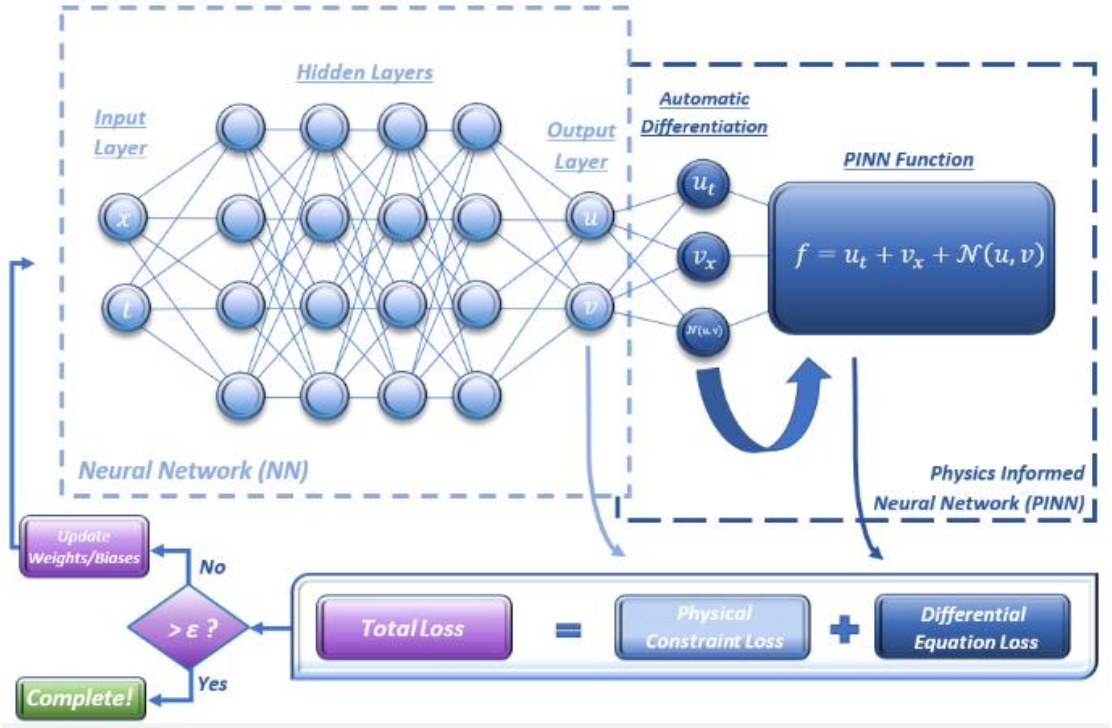
1/07/2024

Lecturer:

Professor Steven Frankel

Teaching Assistant:

Mr. Hemanth Kakumani



Abstract

We present our project on physics-informed neural networks (PINNs), which are neural networks trained to solve supervised learning tasks while adhering to specific laws of physics outlined by the incompressible Navier-Stokes equations. Our work addresses one primary challenge: solving Navier-Stokes equations with PINNs model and figuring out if the solution is accurate in compared to others flow analysis.

In this project, we focus on applying these networks on the low-speed flow of incompressible fluids in a cavity, with and without cylinder, with a specific BC which we will see later. We aim to validate our model outputs using the paper "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method", ensuring the accuracy and reliability of our solutions. By embedding physical laws directly into our neural networks, we aim to create models that are not only accurate and efficient but also capable of providing valuable insights into the behavior of fluid flow systems.

We achieved a strong correlation between the results of the paper and predictions from the PINNs model. However, obtaining a sufficiently low loss function in the PINNs model required a significant number of training epochs.

Introduction

With the rapid growth of available data and advancements in computing resources, recent developments in machine learning and data analytics have produced significant results across various scientific fields. Physics-informed neural networks (PINNs) have emerged as a powerful tool for solving complex physical and engineering problems by integrating data with fundamental physical laws.

Project Focus

In this project, we focus on using PINNs to address fluid dynamics problems governed by the incompressible Navier-Stokes equations. Specifically, we aim to model low-speed fluid flow in a cavity (1) with a given BC of a velocity on one wall (like a conveyer), while all the rest velocities are zeros (on the normal direction to the wall, which is known as no penetration). Our approach leverages the steady state model, making it well-suited for applications such as computational fluid dynamics (CFD) simulations. In addition, we present a solution to a cavity model with a cylinder in the middle.

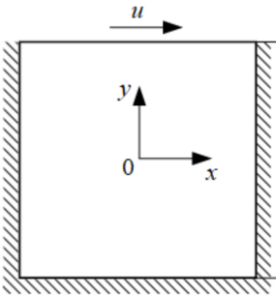


Figure 1-cavity description

Navier-Stokes equations for steady state:

$$\begin{cases} \cancel{\frac{\partial u}{\partial t}} + u \cdot \frac{\partial u}{\partial x} + v \cdot \frac{\partial u}{\partial y} = -\frac{1}{\rho} \cdot \frac{\partial P}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \\ \cancel{\frac{\partial v}{\partial t}} + +u \cdot \frac{\partial v}{\partial x} + v \cdot \frac{\partial v}{\partial y} = -\frac{1}{\rho} \cdot \frac{\partial P}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \end{cases}$$

The continuity equation:

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

Notations

Parameters	Unit	Descriptions
u	$\left[\frac{m}{sec}\right]$	Velocity in x direction
u_i	$\left[\frac{m}{sec}\right]$	Forced velocity of the upper wall
v	$\left[\frac{m}{sec}\right]$	Velocity in y direction
P	$[Pa]$	Pressure
ν	$\left[\frac{m^2}{sec}\right]$	Kinematic viscosity
ρ	$\left[\frac{kg}{m^3}\right]$	Density of fluid
Re	$[-]$	Reynolds number

However, despite the abundance of data and computational power, the cost of data acquisition in analyzing complex physical systems can be prohibitive. This often leads to scenarios where decisions must be made with partial information. Traditional machine learning techniques, which typically require large datasets to perform accurately, may struggle in this "small data" regime. This is where PINNs offer a significant advantage by incorporating physical laws directly into the learning process, thereby reducing the dependence on large datasets and enhancing robustness.

Methods

Governing equations

At first glance, training a deep learning algorithm to accurately identify a nonlinear mapping from a few high-dimensional input-output pairs might seem overly ambitious. However, in the context of physical and engineering systems, we can utilize a wealth of prior knowledge—specifically, the governing physical laws. By embedding these laws into the neural network as

constraints, PINNs restrict the solution space to physically plausible solutions. This constraint acts as a regularization mechanism, preventing the network from learning unrealistic patterns and ensuring that the solutions respect the underlying physics.

The Navier-Stokes equations describe the motion of fluid substances and are essential for understanding fluid dynamics. For incompressible flows, the Navier-Stokes equations in a two-dimensional domain can be written as (as we saw before):

$$\begin{cases} \cancel{\frac{\partial u}{\partial t}} + u \cdot \frac{\partial u}{\partial x} + v \cdot \frac{\partial u}{\partial y} = -\frac{1}{\rho} \cdot \frac{\partial P}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \\ \cancel{\frac{\partial v}{\partial t}} + + u \cdot \frac{\partial v}{\partial x} + v \cdot \frac{\partial v}{\partial y} = -\frac{1}{\rho} \cdot \frac{\partial P}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \end{cases}$$

$$\downarrow$$

$$\begin{cases} u \cdot \frac{\partial u}{\partial x} + v \cdot \frac{\partial u}{\partial y} + \frac{\partial P}{\partial x} = \frac{1}{Re} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \\ u \cdot \frac{\partial v}{\partial x} + v \cdot \frac{\partial v}{\partial y} + \frac{\partial P}{\partial y} = \frac{1}{Re} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \end{cases}$$

The continuity equation:

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

Computational Domain and Boundary Conditions

The computational domain for this study is a square cavity with a cylinder placed inside it. The boundary conditions are critical for the problem definition: The BC are with respect to the domain of the cavity:

$$x \in [0,1] \quad , \quad y \in [0,1]$$

For $u(x,y)$, $v(x,y)$:

$$u(0,y) = u(1,y) = 0$$
$$u(x,0) = 0$$
$$u(x,1) = u_i \, , u_i \; is \; the \; top \; wall's \; velocity.$$
$$v(0,y) = v(1,y) = 0$$
$$v(x,0) = v(x,1) = 0$$

The neural network employed in the PINN model is a fully connected feed-forward network with multiple hidden layers. Each layer consists of neurons activated by non-linear activation functions, we used the tanh function. The input to the network includes x and y parameters, while the output consists of the velocity components and pressure p.

Loss Function

The loss function in a PINN combines physics-based components of Navier-Stokes relations and boundary conditions.

The core of our approach involves training a Physics-Informed Neural Network (PINN) to solve the incompressible Navier-Stokes equations for low-speed fluid flow in a cavity. PINN leverages the inherent physical laws to ensure accurate and physically meaningful solutions. The loss function plays a crucial role in this process by enforcing the constraints dictated by the Navier-Stokes equations and the boundary conditions.

The loss function for the PINN includes two primary components:

1. **Residuals of the Navier-Stokes Equations and Continuity Equation:** This component ensures that the predicted solutions respect the underlying physics of fluid flow. It incorporates the residuals of the Navier-Stokes equations, which govern the momentum balance in the fluid, and the continuity equation, which ensures mass conservation.
2. **Boundary Conditions for u and v:** This component enforces the specified boundary conditions for the velocity components u and v within the cavity. These conditions include zero velocities along the walls and a specified velocity on the top wall.

The combined loss function can be represented as:

$$Loss = Loss_{eq} + 10 \cdot Loss_{BC}$$

Where:

- $Loss_{eq}$ ensures that the predicted solutions satisfy the Navier-Stokes and continuity equations.
- $Loss_{BC}$ ensures that the boundary conditions are met.

To quantify these components, we use the Mean Squared Error (MSE) loss. The MSE loss measures the average squared difference between the predicted values and the actual values, providing a robust metric for optimization. For the Navier-Stokes equations, the MSE loss is computed based on the residuals of the equations. For the boundary conditions, the MSE loss is computed based on the deviation of the predicted velocities from the specified boundary values. By multiplying the boundary condition loss by a factor of 10, we give greater importance to meeting the boundary conditions, ensuring that the PINN predictions adhere closely to the physical constraints imposed by the boundaries.

$$Loss_{eq} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{N}(u_i, v_i, P_i)\right)^2$$

$$Loss_{BC} = \frac{1}{M}\sum_{i=1}^{M}\left(\left(u_j - u_{bc}\right)^2 + \left(v_j - v_{BC}\right)^2\right)$$

Here, $\mathcal{N}(u_i, v_i, P_i)$ represents the residuals Navier-Stokes and continuity equations evaluated at point i, N is the number of points used to evaluate the equations, M is the number of boundary points, and u_{BC}, v_{BC} are the boundary condition values for the velocities.
By minimizing this combined loss function, the PINN is trained to provide solutions that are physically consistent. This approach leverages the inherent regularization provided by the physical constraints, reducing the reliance on large datasets and enhancing the robustness and accuracy of the model predictions.

Optimization with Adam:

To optimize the neural network parameters, we use the Adam optimizer, which is a popular choice in training deep learning models due to its efficiency and effectiveness. It adapts the learning rate for each parameter individually, using estimates of the first and second moments of the gradients to adjust the learning rate dynamically.

The Adam optimizer works by maintaining running averages of both the gradients m_t and the squared gradients v_t. These averages are then used to update the parameters as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad , \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

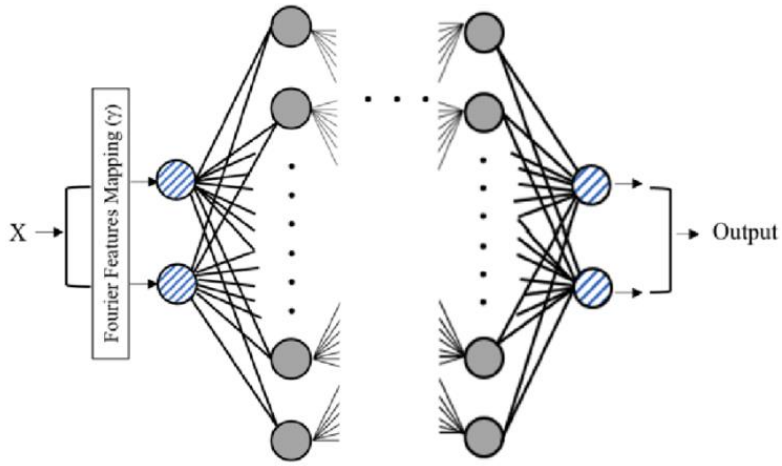
Here, g_t is the gradient at time step t, β_1 and β_2 are the decay rates for the moment estimates, $\hat{m}_t$and $\hat{v}_t$are the bias-corrected estimates, α is the learning rate, ϵ is a small constant to prevent division by zero, and θ_t represents the network parameters.

The adaptive nature of Adam makes it particularly well-suited for problems with sparse gradients or noisy objectives, which are common in the training of neural networks. By employing Adam, we ensure efficient and effective convergence of our PINN, leading to accurate and reliable solutions to the fluid dynamics problems we are addressing.

FFM- Fourier Feature Mapping:

Fourier Feature Mapping (FFM) is a transformative technique in machine learning, particularly within kernel methods. FFM addresses the computational complexities of traditional kernel computations by mapping input data into a higher-dimensional space using sinusoidal functions, thus eliminating the need for explicit kernel computations. This innovative approach reduces computational burdens and enhances scalability, which was previously unattainable with conventional methods. FFM's brilliance lies in its simplicity: by selecting random frequencies and phases, it approximates complex kernel functions, such as the Gaussian radial basis function. This mapping renders the relationship between data points linear in the Fourier space, offering a

versatile tool that surpasses the limitations of traditional kernel methods. FFM is applicable across various machine learning tasks, including regression, classification, and generative modeling, making it an effective solution for real-world challenges involving large datasets and high-dimensional feature spaces. In summary, Fourier Feature Mapping is a catalyst for innovation, enabling scalable, efficient, and flexible machine learning solutions in the era of big data.

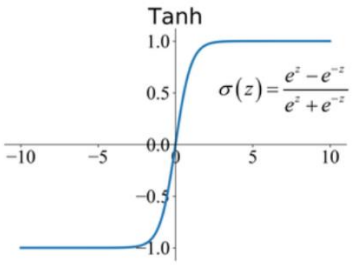


Activation Function- Tanh (hyperbolic tangent):

The tanh (hyperbolic tangent) activation function is a popular choice in neural networks due to its desirable properties. Mathematically, it is defined as

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

which transforms the input values into a range between -1 and 1. This symmetric output range around zero is beneficial as it ensures that the outputs of the neurons are centered, which can lead to faster convergence during training. Unlike the sigmoid function, which outputs values between 0 and 1, tanh allows for both positive and negative activations, enabling better modeling of data with a more significant range of outputs. This characteristic helps in mitigating the vanishing gradient problem, although not completely, making it more suitable for deeper networks. The tanh function is particularly effective in hidden layers where a symmetric activation range is advantageous for learning complex patterns and representations in data.



Neural Network Layers:

The architecture of the Physics-Informed Neural Network (PINN) used in this project is implemented in the PINNs_net class. This neural network is specifically designed to solve the incompressible Navier-Stokes equations for low-speed fluid flow in a cavity. The network architecture consists of an input layer, a feature mapping layer, five hidden layers, and an output layer. Below, we describe the details of each layer in terms of order and size.

Layer Descriptions

- **Dimensions:** 2 (inputs x and y)
- **Function:** Takes the spatial coordinates x and y as input.

2. **Feature Mapping Layer (ffm2):**
3. **Dimensions:** Maps the 2 input dimensions to 128 dimensions using cosine and sine **Input Layer:**
    - transformations.
    - **Function:** Enhances the input features by applying a high-dimensional mapping. Specifically, it projects the inputs using a fixed matrix $\boldsymbol{B}$ and applies cosine and sine functions to the result:

$$x_{map} = \boldsymbol{B} \cdot x$$

$$output = \left[\cos\left(x_{map}\right), \sin\left(x_{map}\right)\right]$$

4. **Hidden Layer 1:**
    - **Dimensions:** 128 (input) → 64 (output)

- **Function:** Applies a linear transformation followed by the tanh activation function to the output of the feature mapping layer.

$$output = \tanh(w_1 x_{map} + b_1)$$

5. **Hidden Layer 2:**
   - **Dimensions:** 64 (input) → 64 (output)
   - **Function:** Applies a linear transformation followed by the tanh activation function to the output of Hidden Layer 1.

$$output = \tanh(w_2 h_1 + b_2)$$

6. **Hidden Layer 3:**
   - **Dimensions:** 64 (input) → 64 (output)
   - **Function:** Applies a linear transformation followed by the tanh activation function to the output of Hidden Layer 2.

$$output = \tanh(w_3 h_2 + b_3)$$

7. **Hidden Layer 4:**
   - **Dimensions:** 64 (input) → 64 (output)
   - **Function:** Applies a linear transformation followed by the tanh activation function to the output of Hidden Layer 3.

$$output = \tanh(w_4 h_3 + b_4)$$

8. **Hidden Layer 5:**
   - **Dimensions:** 64 (input) → 64 (output)
   - **Function:** Applies a linear transformation followed by the tanh activation function to the output of Hidden Layer 4.

$$output = \tanh(w_5 h_4 + b_5)$$

9. **Output Layer:**
   - **Dimensions:** 64 (input) → 3 (output)

- **Function:** Transforms the output of Hidden Layer 5 into three output variables: u (velocity in x-direction), v (velocity in y-direction), and P (pressure).

$$[u, v, P] = w_{out} h_5 + b_{out})$$

Summary of the Layer Sizes

- **Input Layer:** 2 neurons (for x and y)
- **Feature Mapping Layer:** 128 neurons (64 for cosine outputs and 64 for sine outputs)
- **Hidden Layer 1:** 64 neurons
- **Hidden Layer 2:** 64 neurons
- **Hidden Layer 3:** 64 neurons
- **Hidden Layer 4:** 64 neurons
- **Hidden Layer 5:** 64 neurons
- **Output Layer:** 3 neurons (for u, v, and P)

This architecture ensures sufficient capacity to model the complex relationships inherent in the Navier-Stokes equations while maintaining computational efficiency. The use of the feature mapping layer aids in capturing intricate periodic patterns, and the sequence of hidden layers with 64 neurons each provides the depth needed for accurate predictions. The final layer produces the required outputs, which are used to compute the residuals and boundary condition losses during training.

Evaluation and Final Results

Cavity flow

After training the PINNs we will look at the flow in a cavity after 150,000 epochs.

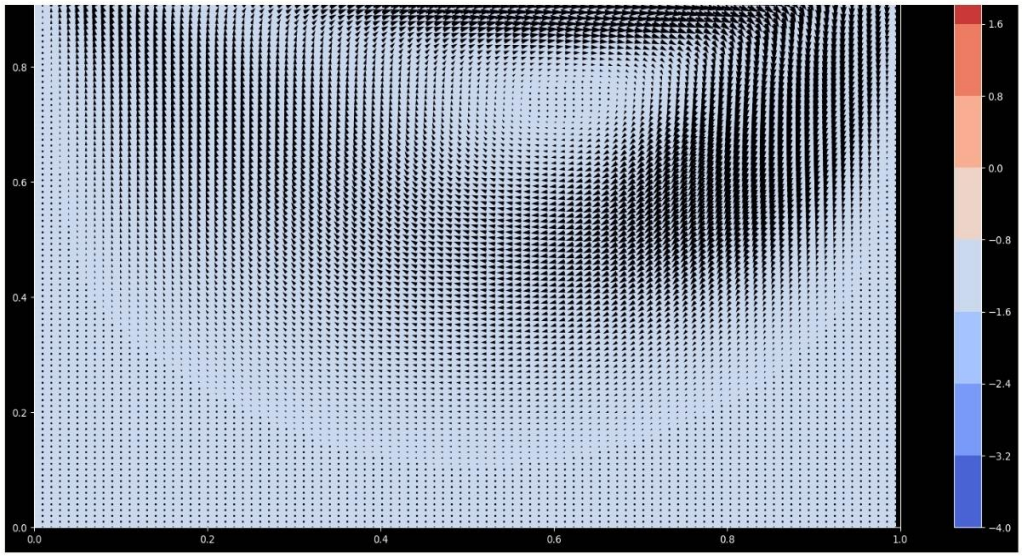


Figure 2-cavity flow

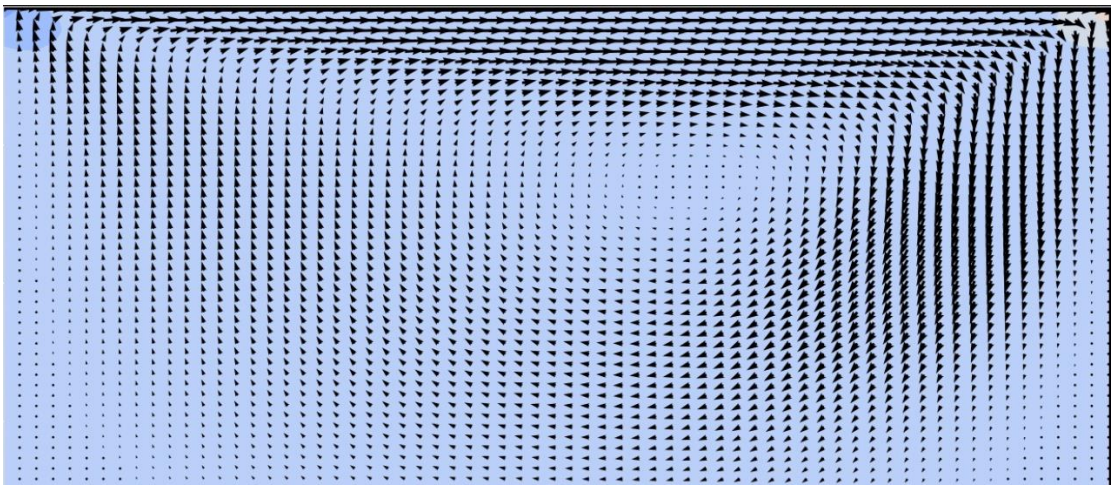


Figure 3-cavity flow zoom

We observe a circular flow pattern, which aligns with the literature. The length of the vectors represents the velocity magnitude. Near the upper wall, the velocity is highest, which is expected due to the lid-driven motion. Conversely, near the lower wall, the velocity is significantly lower. This observation is consistent with the physical behavior of the fluid in a cavity flow scenario.

We can also see the distribution of the velocities:

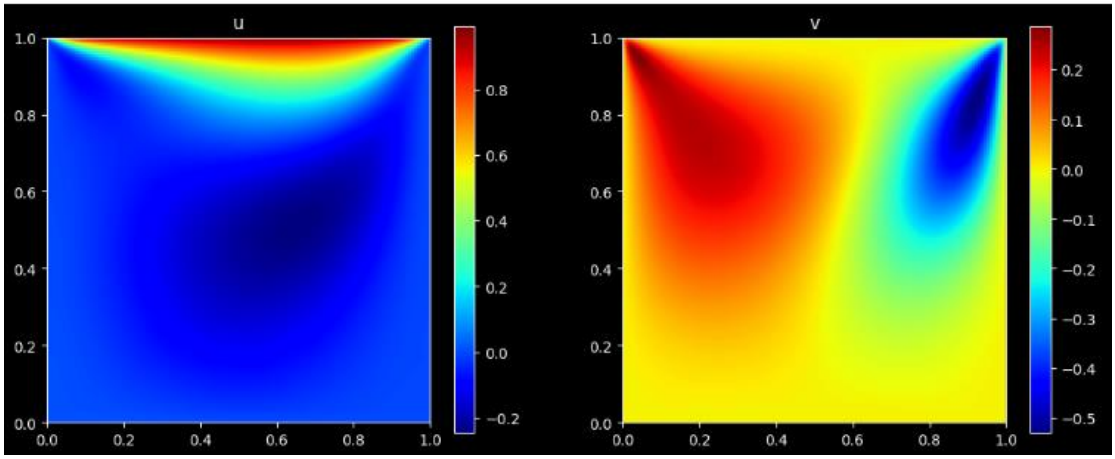


Figure 4-velocities fields distribution

We observe that in the velocity field for u (velocity in the x-direction), the highest velocities are near the upper wall, which aligns with the boundary conditions. In the velocity field for v (velocity in the y-direction), we see a change in direction: blue regions indicate downward velocity, while red regions indicate upward velocity. This behavior is consistent with our expectations and intuition.

Now we can compare our velocities fields with graphs from the article " High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method " Table -1. We will look at the velocity's centerline.

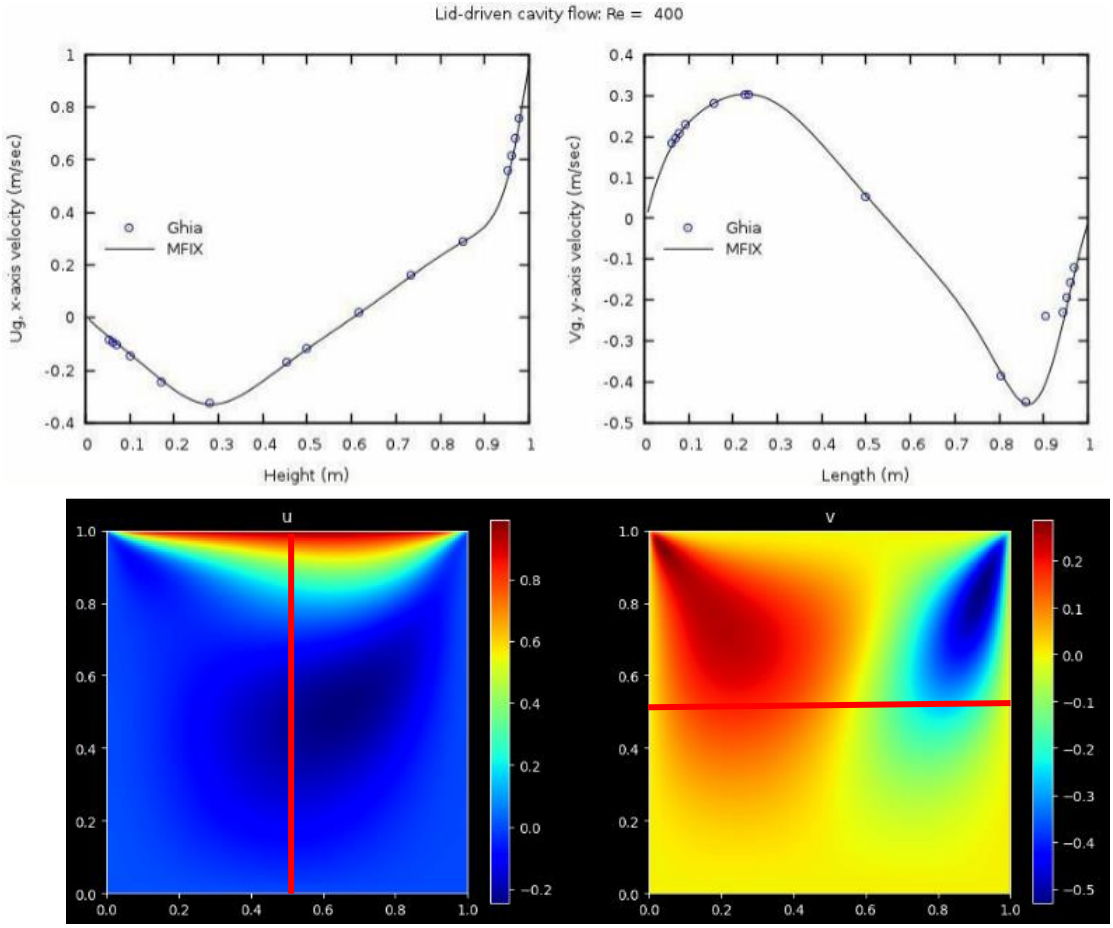


Figure 5- Table 1-article

1) u-velocity centerline.
   - The vertical centerline (x = 0.5) in the lower left graph aligns well with the upper left graph.
   - Velocity starts negative (blue region), decreases further.
   - Then sharply increases towards the upper wall (red/orange region). Which aligned with the boundary conditions.

2) v-velocity centerline.
   - The horizontal centerline (y = 0.5) in the lower right graph aligns well with the upper right graph.
   - Velocity increases (yellow/red region), peaks around mid-length.
   - Then decreases, going negative towards the right side (blue region).

In summary, the centerlines of the lower graphs match the trends shown in the upper graphs, confirming that the results align well with expected physical behavior and boundary conditions.

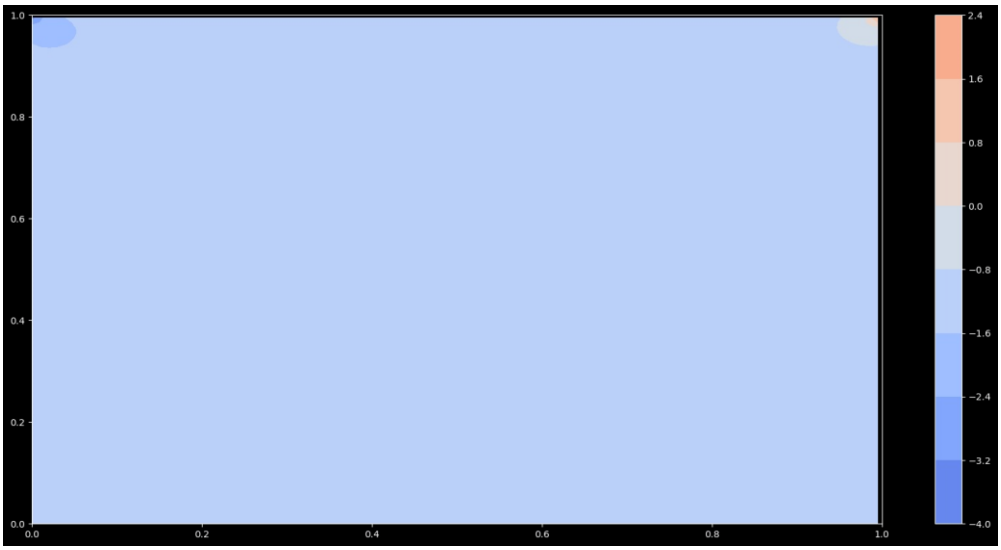


Figure 6-pressure distribution

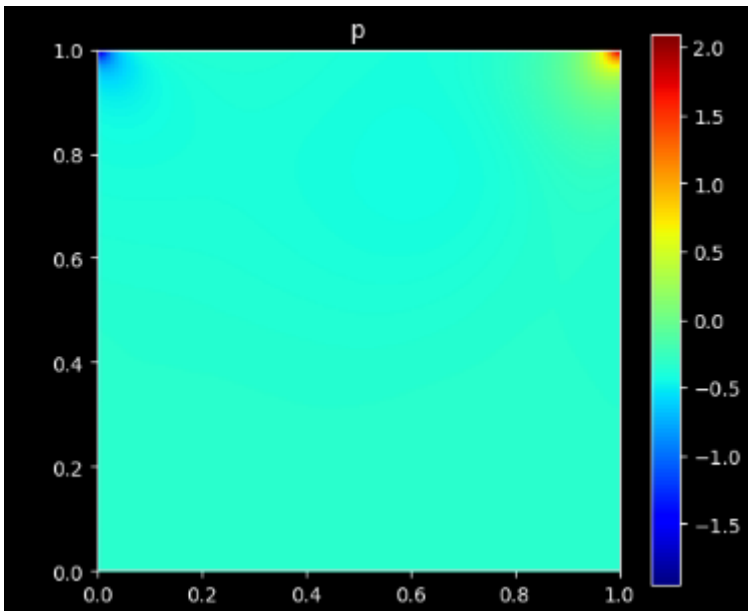


Figure 7-more clearly pressure distribution

In Figures 6 and 7, we observe the pressure distribution within the cavity flow. Notably, there are two distinct regions along the upper wall. On the right side, we see a high-pressure area, while on the left side, there is a low-pressure area. This distribution is logical, as the fluid accelerates along the upper wall and changes direction near the right wall, creating a high-pressure stagnation

point at the corner. A similar but opposite effect occurs at the left corner, where the pressure is low. This pattern aligns with the expected behavior of the fluid dynamics in the cavity.

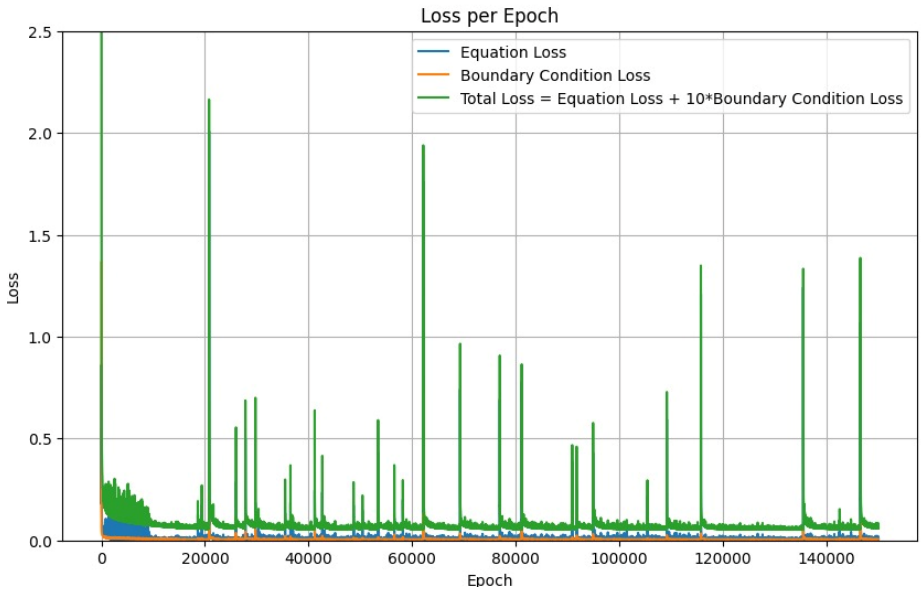


Figure 8-loss function

While the loss function typically decreases with each epoch, it is common to see occasional spikes followed by quick drops. This can occur for several reasons:

- Random initialization**:** Neural network weights are initially random. During early training, these weights might lead to high loss values until the training algorithm adjusts them towards better predictions.
- Optimization noise: Optimization algorithms used for training can exhibit stochastic (random) behavior. This can lead to fluctuations in the loss function, even if the overall trend is downward.

Conclusion

In this project, we utilized Physics-Informed Neural Networks (PINNs) to solve the incompressible Navier-Stokes equations for low-speed fluid flow in a cavity. Our primary goal was to validate the accuracy and reliability of the PINN model by comparing its predictions with established results from the literature.

Summary of Achievements:

- **Model Implementation**: Developed and implemented a PINN to model fluid flow governed by the Navier-Stokes equations.
- **Boundary Conditions**: Applied specific boundary conditions to simulate realistic fluid behavior within the cavity.
- **Validation**: Validated the model outputs against high-resolution solutions from the paper "High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method."

Key Findings:

- **Velocity Fields**: The u-velocity showed the highest values near the upper wall, and the v-velocity exhibited expected directional changes, both aligning well with the boundary conditions and physical intuition.
- **Centerline Analysis**: Centerline velocities in the u and v directions matched closely with the reference data, confirming the accuracy of the PINN model.
- **Pressure Distribution**: Observed distinct high and low-pressure regions along the upper wall, consistent with the expected fluid dynamics behavior.
- **Loss Function**: Achieved a strong correlation between the loss function trends and model predictions, indicating effective training and convergence of the PINN.

BIBILOGRAPHY

- http://www.msaidi.ir/upload/Ghia1982.pdf
- "Understanding deep Learning, Prince, MIT Press, 2023"
- "Tutorial 4"