King Fahd University of Petroleum and Minerals

College of Computing and Mathematics

Mathematics Department

PETE547-Computational Multi-Physics Modeling

Term 231

Project

Lid-Driven Cavity in 2D via Physics informed Neural Network and finite difference method

Name	KFUPM ID
Hashim Al-Sadah	201578370
Abdulwahab Alghamdi	201734070
Hussain Al-Sinan	202205120

Instructor: Dr. Abeeb A. Awotunde

Abstract

Due to the rapid growth in the applications of machine learning and deep learning in various scientific and industrial sectors, researchers are becoming more invested in harnessing the power of deep neural network in scientific computing and the modelling and simulation of dynamical systems. To simulate or predict the behavior of a certain system, we need to solve or approximate the solution of one or more partial differential equation (PDE). Physics Informed Neural Networks (PINNs) is deep learning technique, which enforces the physical laws involved in the problem by incorporating the governing physical equations, which consist of the PDEs along with their boundary and initial conditions. PINN can be used to overcome the difficulties faced by classical methods such as approximating the solution of high dimensional PDEs and complex mesh generation. In this project, we simulate the behavior of an incompressible fluid in the two dimensional lid-driven cavity problem using physical informed neural networks and compare the acquired approximation against the approximated solution obtained by the finite difference method (FDM).

1 Introduction

Due to the increase power of computation and the success of deep learning models to solve different problems in various fields such as computer vision, pattern recognition, and natural language and speech processing, researchers were inspired to apply deep neural network to solve problems in the field of scientific computing. One of the common and challenging problems is the problem of solving partial differential equations (PDEs). Different systems and processes are modelled by PDEs whether it is a natural system such as modelling a biological or a physical phenomena, or whether it is not a natural system such as simulating socioeconomic or financial models. Deep learning-based PDEs solver surpass classical methods such as finite element or finite difference in certain cases. One of them is the case of high dimensional PDEs since classical solvers require discretizing the PDEs' domain into a mesh, which causes the number of computations to increase exponentially with the increase of the dimensions and this is known as the curse of dimensionality. On the other hand, deep learning

models are mesh-free models and they only require training data from the PDEs domain. Other situations where deep neural networks exceeds the traditional solvers is when the PDE is nonlinear or non-smooth, which makes it difficult to discretize.

Physics-informed Neural Network (PINN) is the most basic and widely used model for approximating PDEs' solutions. PINN is unsupervised learning model since it does not require labelled data to learn the approximated function or solution. Similar to the to the fully connected neural network, the physics-informed neural network consists of an input layer, hidden layers, and an output layer. First, the input layer is used to feed the training data into the neural network. Then, the hidden layers map the data to higher dimensions though a series of of linear transformation followed by a nonlinear activation function. Finally, the hidden layers map the data to the output layer, which provides the output of the approximated function. The physics-informed neural network (PINN) updates the learning parameters by minimizing a loss function that takes into account the losses due the boundary and initial conditions as well as the PDEs' residual. After the training process, the neural network should be able to approximate a function that obeys the laws provided by the PDEs. Therefore, the procedure of including the PDEs' residual helps to restrict the number of possible solutions.

Another famous model, which has been developed recently, is the Neural Operator model. What makes neural operators different from physics-informed neural networks (PINNs) is that the former learns a mapping between infinite function spaces unlike PINN where the mapping occurs between finite spaces or sets. This enables neural operators to learn an operator instead of a function and hence the name Neural Operator. The key difference between PINN and neural operators in terms of architecture is that the hidden layers in neural operators consist of linear operators, usually integral operators, followed by a nonlinear activation function. Therefore, neural operators are considered a generalization of physics-informed neural network.

Overall, deep learning methods for PDEs are powerful solvers. They have the potential for resolving various problems and challenges faced by the classical methods. The advantage of neural networks models is that they are mesh-free methods, which enables them to approximate the solution for nonlinear, non-smooth, and high di-

mensional PDEs without suffering from the cures of dimensionality. Furthermore, they utilize the algorithm of automatic differentiation to compute the residual of the PDEs. The algorithm is implemented by the majority of deep learning libraries, which makes the overall implementation process simple and easy.

In the following section, we introduce the problem of lid-driven cavity in 2D for incompressible fluid. Then we provide a detailed description on how to approximate the solution via finite difference method. Next, we use PINNs to simulate the behavior of the fluid by approximating the velocity and the pressure as function of time and space. Finally, the approximated solution of PINNs is compared against the approximation obtained by the finite difference method. All the codes for the work done in this report is available on https://github.com/HashimAlSadah/PETE547_Project

2 Problem Statement

We will consider the problem of lid-driven cavity in 2D for an incompressible fluid with a velocity slip on the upper boundary. The domain of the problem is $[0, l_x] \times [0, l_y]$.

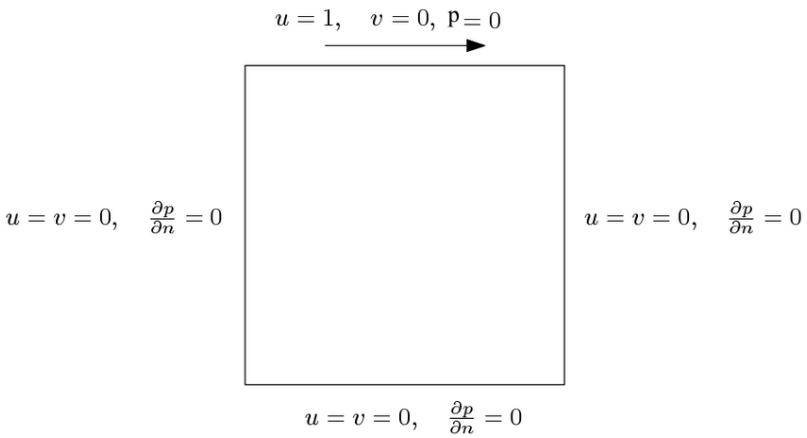

Figure 1: Problem geometry and boundary conditions

To model this problem, we use the reduced Navier-Stokes equations in 2D along with the condition for the incompressible fluid, which is derived from the continuity equation for fluids' flow.

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \vec{u} \tag{1}$$

$$\nabla \cdot \vec{u} = 0 \tag{2}$$

Where $\vec{u}(t, x, y)$ is the velocity vector, t is time, $p(t, x, y)$ is the pressure, ρ is the fluid density, and ν is the viscosity of the fluid.

$$\text{Let} \quad \vec{u} = \begin{bmatrix} u \\ v \end{bmatrix}$$

Where u and v are the velocities in the x-direction and the y-direction, respectively.

Therefore,

$$\vec{u} \cdot \nabla \vec{u} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} \end{bmatrix}, \qquad \nabla^2 \vec{u} = \begin{bmatrix} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \\ \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \end{bmatrix}$$

Substituting these in Eq. 1, we get the following equations.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{3}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{4}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{5}$$

Looking at Eq. 5, we notice that there is no coupling between the velocity and the pressure, which makes it difficult to solve the system of differential equations, so we must find a way to couple the three equations. To do so, we differentiate Eq. 3 with respect to x and Eq. 4 with respect to y. Then we add the equations together and include Eq. 5 as a constraint. The procedure is the following.

$$\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}\right)+\left(\frac{\partial u}{\partial x}\right)^2+u\frac{\partial^2 u}{\partial x^2}+\frac{\partial v}{\partial x}\frac{\partial u}{\partial y}+v\frac{\partial^2 u}{\partial x\partial y}=-\frac{1}{\rho}\frac{\partial^2 p}{\partial x^2}+\nu\left(\frac{\partial^3 u}{\partial x^3}+\frac{\partial^3 u}{\partial x\partial y^2}\right) \quad (6)$$

$$\frac{\partial}{\partial t}\left(\frac{\partial v}{\partial y}\right)+\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}+u\frac{\partial^2 v}{\partial y\partial x}+\left(\frac{\partial v}{\partial y}\right)^2+v\frac{\partial^2 v}{\partial y^2}=-\frac{1}{\rho}\frac{\partial^2 p}{\partial y^2}+\nu\left(\frac{\partial^3 v}{\partial y\partial x^2}+\frac{\partial^3 v}{\partial y^3}\right) \quad (7)$$

We add Eq. 6 and Eq. 7 and we rearrange

$$\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)+\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2+2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}+u\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 v}{\partial y\partial x}\right)+v\left(\frac{\partial^2 u}{\partial x\partial y}+\frac{\partial^2 v}{\partial y^2}\right)$$
$$=-\frac{1}{\rho}\left(\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}\right)+\nu\left(\frac{\partial^3 u}{\partial x^2}+\frac{\partial^3 u}{\partial x\partial y^2}+\frac{\partial^3 v}{\partial y\partial x^2}+\frac{\partial^3 v}{\partial y^3}\right) \quad (8)$$

The last step is to include Eq. 5 into Eq. 8. We will also include its first and second derivatives with respect to x and y, which are the following

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 v}{\partial y\partial x}=0$$
$$\frac{\partial^2 u}{\partial x\partial y}+\frac{\partial^2 v}{\partial y^2}=0$$
$$\frac{\partial^3 u}{\partial x^3}+\frac{\partial^3 v}{\partial x^2\partial y}=0$$
$$\frac{\partial^3 u}{\partial y^2\partial x}+\frac{\partial^3 v}{\partial y^3}=0$$

By substituting these equations along with Eq. 5 into Eq. 8, we get following

$$\left(\frac{\partial u}{\partial x}\right)^2+2\left(\frac{\partial u}{\partial y}\right)\left(\frac{\partial v}{\partial x}\right)+\left(\frac{\partial v}{\partial y}\right)^2=-\frac{1}{\rho}\left(\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}\right) \quad (9)$$

Therefore, our final equations are

$$u_t+u\,u_x+v\,u_y=-\frac{1}{\rho}\,p_x+\nu\left(u_{xx}+u_{yy}\right) \quad (10)$$

$$v_t + u\, v_x + v\, v_y = -\frac{1}{\rho}\, p_y + \nu\,(v_{xx} + v_{yy}) \tag{11}$$

$$u_x^2 + 2u_y\, v_x + v_y^2 = -\frac{1}{\rho}\,(p_{xx} + p_{yy}) \tag{12}$$

Note: We have used u_x instead of $\frac{\partial u}{\partial x}$ for simplicity and similarly for other the derivatives.

The boundary conditions for the velocity are the following.

$$u(t, x, l_y) = 1, \quad u(t, x, 0) = 0, \quad u(t, 0, y) = 0, \quad u(t, l_x, y) = 0$$

$$v(t, x, l_y) = 0, \quad v(t, x, 0) = 0, \quad v(t, 0, y) = 0, \quad v(t, l_x, y) = 0$$

We consider the case, $l_x = l_y = 1$

The boundary condition for the pressure

$$\frac{\partial p}{\partial x}\Big|_{x=0} = 0, \quad \frac{\partial p}{\partial x}\Big|_{x=l_x} = 0, \quad \frac{\partial p}{\partial y}\Big|_{y=0} = 0, \quad p(t, x, l_y) = 0$$

For the initial condition we assume that the velocity and the pressure are zero everywhere except at the boundary (we apply the boundary conditions).

3 Finite Difference

3.1 Discreatization

To use the finite difference method to solve the problem, we need to discretize the space and the time dimensions. We have two spatial components, which are x and y and the time component t

We discretize the x component into equal intervals, $\Delta x = h$, where each point is given by $x_i = ih + x_0$, for $i = 0, 1, 2, \ldots, H$.

We do the same for the y component. We divide it into equal intervals $\Delta y = k$ and each point is denoted by $y_j = jk + y_0$ for $j = 0, 1, 2, \ldots, K$.

We use a fixed step, Δt, for the time component, where $t_n = 0, 1, 2, \ldots$.

We start by discretizing Eq. 10.

$$u_t + u\, u_x + v\, u_y = -\frac{1}{\rho}\, p_x + \nu\,(u_{xx} + u_{yy})$$

We will use the forward Euler formula to approximate the time derivative, the central difference for the first and the second derivative in space.

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} + u_{i,j}\frac{u_{i+1,j}^{n} - u_{i-1,j}^{n}}{2h} + v_{i,j}\frac{u_{i,j+1}^{n} - u_{i,j-1}^{n}}{2k} = -\frac{1}{\rho}\frac{p_{i+1,j}^{n} - p_{i-1,j}^{n}}{2h} + \nu\left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{h^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{k^2}\right)$$

Rearrange

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \Delta t\left[- u_{i,j}\frac{u_{i+1,j}^{n} - u_{i-1,j}^{n}}{2h} - v_{i,j}\frac{u_{i,j+1}^{n} - u_{i,j-1}^{n}}{2k} - \frac{1}{\rho}\frac{p_{i+1,j}^{n} - p_{i-1,j}^{n}}{2h} + \nu\left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{h^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{k^2}\right)\right] \quad (13)$$

Similarly, we discretize Eq. 11

$$v_{i,j}^{n+1} = v_{i,j}^{n} + \Delta t\left[- u_{i,j}\frac{v_{i+1,j}^{n} - v_{i-1,j}^{n}}{2h} - v_{i,j}\frac{v_{i,j+1}^{n} - v_{i,j-1}^{n}}{2k} - \frac{1}{\rho}\frac{p_{i,j+1}^{n} - p_{i,j-1}^{n}}{2k} + \nu\left(\frac{v_{i+1,j}^{n} - 2v_{i,j}^{n} + v_{i-1,j}^{n}}{h^2} + \frac{v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j-1}^{n}}{k^2}\right)\right] \quad (14)$$

Eq. 12 is used to compute the pressure using the values of the velocities obtained

from Eqs. 13 and 14.

$$u_x^2 + 2u_y\, v_x + v_y^2 = -\frac{1}{\rho}\left(p_{xx} + p_{yy}\right)$$

The equation above works well when solving the problem analytically, but when it is discretized it needs a special treatment.

To understand the problem, we need to look at the time discretization for the Navier-Stokes equation

$$\frac{\vec{u}^{n+1} - \vec{u}^n}{\Delta t} + \vec{u}^n \cdot \nabla \vec{u}^n = -\frac{1}{\rho}\nabla p^{n+1} + \nu \nabla^2 \vec{u}^{n+1}$$

Solve for the pressure

$$\Delta t\, \frac{1}{\rho}\nabla p^{n+1} = -\vec{u}^{n+1} + \vec{u}^n - \Delta t\, \vec{u}^n \cdot \nabla \vec{u}^n + \Delta t\, \nu \nabla^2 \vec{u}^{n+1}$$

Take the divergence of both sides.

$$\Delta t\, \frac{1}{\rho}\nabla^2 p^{n+1} = -\nabla \cdot \vec{u}^{n+1} + \nabla \cdot \vec{u}^n - \Delta t\, \nabla \cdot (\vec{u}^n \cdot \nabla \vec{u}^n) + \Delta t\, \nu \nabla \cdot \nabla^2 \vec{u}^{n+1} \tag{15}$$

The condition for the compressible fluid should guarantee $\nabla \cdot \vec{u}^n = 0$, but due the numerical error in computation, the condition might not be satisfied. Therefore, we cannot remove it from the equation as we did when we derived Eq. 12. But, we can enforce this condition on $\vec{u}^{n+1}$; That is $\nabla \cdot \vec{u}^{n+1} = 0$.

Therefore, the equation will be simplified to the following.

$$\nabla^2 p^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot \vec{u}^n - \rho \nabla \cdot (\vec{u}^n \cdot \nabla \vec{u}^n)$$

$$p_{xx}^{n+1} + p_{yy}^{n+1} = \frac{\rho}{\Delta t}\left(u_x^n + v_y^n\right) - \rho\left(u_x^2 + uu_{xx} + v_x u_y + vu_{yx} + u_y v_x + uv_{xy} + v_y^2 + vv_{yy}\right)$$

For a first order correction or approximation, we can drop any second derivative velocity.

$$p_{xx}^{n+1} + p_{yy}^{n+1} = \frac{\rho}{\Delta t}\left(u_x + v_y\right) - \rho\left(u_x^2 + 2u_y v_x + v_y^2\right) \tag{16}$$

Note: We have dropped the superscript n from the velocities for simplicity. Since it is understandable we need known velocity to compute the pressure.

We notice that the only difference between Eq. 12 and Eq. 16 is the first term on the right-hand side of Eq. 16. We can think of this term as a corrector for the numerical errors that occurred when computing the velocities in the previous iteration.

The discretization of Eq. 16 is the following.

$$\frac{p^{n+1}_{i+1,j} - 2p^{n+1}_{i,j} + p^{n+1}_{i-1,j}}{h^2} + \frac{p^{n+1}_{i,j+1} - 2p^{n+1}_{i,j} + p^{n+1}_{i,j-1}}{k^2} = \frac{\rho}{\Delta t}\left(\frac{u_{i+1,j} - u_{i-1,j}}{2h} + \frac{v_{i,j+1} - v_{i,j-1}}{2k}\right) - \rho\left[\left(\frac{u_{i+1,j} - u_{i-1,j}}{2h}\right)^2 + 2\left(\frac{u_{i,j+1} - u_{i,j-1}}{2k}\right)\left(\frac{v_{i+1,j} - v_{i-1,j}}{2h}\right) + \left(\frac{v_{i,j+1} - v_{i,j-1}}{2k}\right)\right] \quad (17)$$

The equation above is a Poisson equation, so to find the pressure we need to solve a linear system of equations. An alternative way to solve the problem is to use Jacobi's method. Where we use the pressure from the previous iteration $p^{n+1}_{i+1,j} = p^{n}_{i+1,j}$ and the same is applied for other pressure terms except $p^{n+1}_{i,j}$. Therefore, our final equation is the following.

$$p^{n+1}_{i,j} = \frac{h^2k^2}{2(h^2+k^2)}\left\{\frac{p^{n}_{i+1,j} + p^{n}_{i-1,j}}{h^2} + \frac{p^{n}_{i,j+1} + p^{n}_{i,j-1}}{k^2} - \frac{\rho}{\Delta t}\left(\frac{u_{i+1,j} - u_{i-1,j}}{2h} + \frac{v_{i,j+1} - v_{i,j-1}}{2k}\right) + \rho\left[\left(\frac{u_{i+1,j} - u_{i-1,j}}{2h}\right)^2 + 2\left(\frac{u_{i,j+1} - u_{i,j-1}}{2k}\right)\left(\frac{v_{i+1,j} - v_{i-1,j}}{2h}\right) + \left(\frac{v_{i,j+1} - v_{i,j-1}}{2k}\right)\right]\right\} \quad (18)$$

To find the next value of the pressure, we need to iterate through Eq. 18 until the solution converges up to a certain tolerance.

3.2 FDM Simulation Results

To perform the simulation, we have discretized the x-axis and the y-axis into 50 points; That is $\Delta x = h = 0.02$ and $\Delta y = k = 0.02$. The time step was taken as $\Delta t = 0.001$ and. Furthermore, we have used 50 iterations to smooth out the pressure, Eq. 18. The results obtained for $T = 1.0$, which is 1000 iterations.

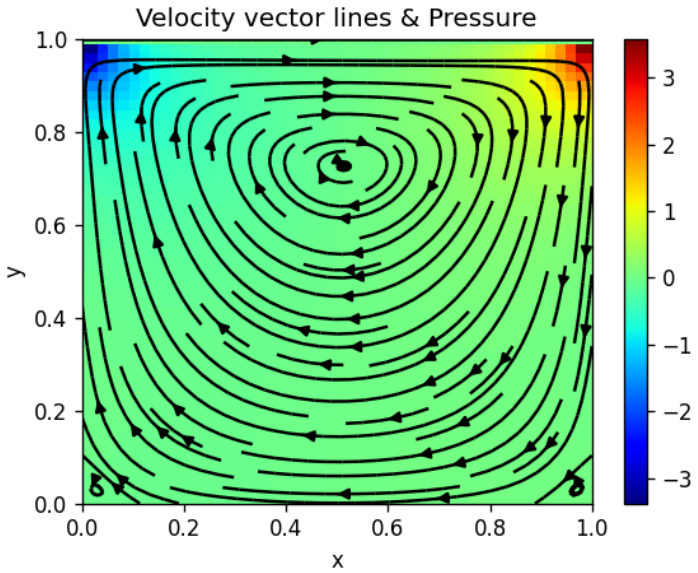

Figure 2: Pressure and velocity vector lines for FDM at $T = 1.0$

Figure 2 shows the pressure as function of space and the vector lines of the velocity in the cavity. The simulation is showing the expected behavior, where the horizontal velocity at the top is equal 1; That is $u = 1$ and since the velocity is moving the fluid from left to right (from $x = 0$ to $x = 1$) at $y = 1$, the pressure is decreasing at the upper left boundary and increasing at the upper right boundary. The difference in pressure will cause fluid to flow from right to left to keep the pressure at zero as required by the boundary condition $p(t, x, y = 1) = 0$.

Figure 3 shows the plot of pressure, horizontal velocity u, and vertical velocity v as function of space at $T = 1.0$. We can notice that the velocity in the horizontal decreases as we go further from the slip, which is the expected. Also, the vertical velocity is pointing away from the upper right boundary since it is flowing away from

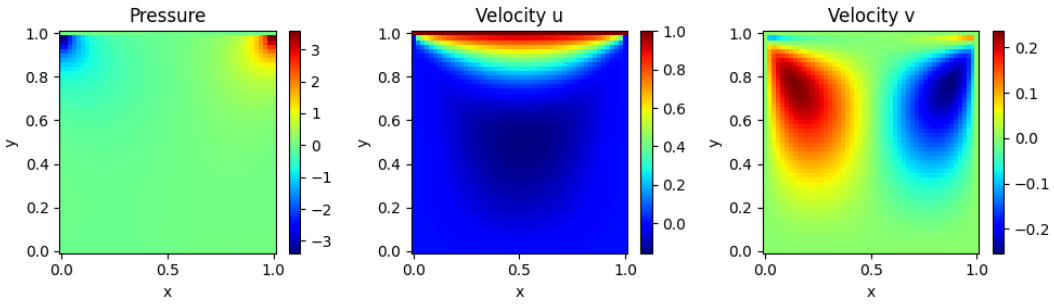

Figure 3: Pressure and horizontal and vertical velocities at $T = 1.0$ for FDM

the high pressure region and it is pointing toward the low pressure region and that causes the creation of the vortex.

4 PINN's

4.1 Architecture & Procedure

Similar to conventional neural network methods, the architecture of PINNs takes an input, passes it through hidden layers, and produces an output. In our case, our PINN architecture takes (x, y, t) as our input, passes it through 7 hidden layers, and outputs (u, v, p). Then it calculates the total loss term composed of PDE losses, Boundary condition losses, and Initial condition losses. To optimize the results, it uses an Adam optimizer to minimize the cost function through gradient descent. The gradient values are obtained by back-propagation. The neural network keeps iterating through the whole process until the it reaches the epoch threshold. The used neural network architecture is shown in Figure 4.

4.2 Simulation & Results

Figure 5 shows that the results for the PINN's simulation show that the fluid have a similar behavior to one obtained by the FDM. However, we can see a little disturbance at the lower left and right boundaries.

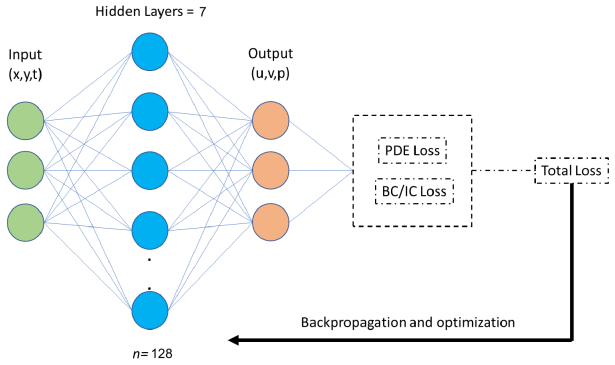

Figure 4: Neural network architecture

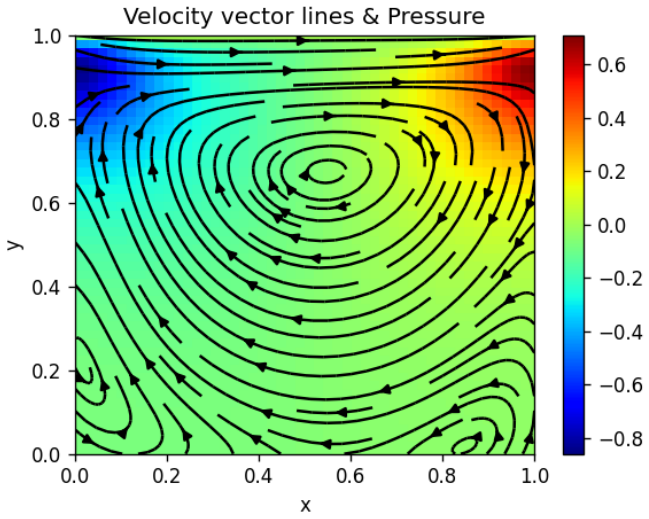

Figure 5: Pressure and velocity vector lines for PINN at $T = 1.0$

Figure 6 shows the pressure and the velocities as function of space and visually we can notice the similarity between the velocities in both directions for the FDM and the PINN. We notice that this not the case for the pressure, where the values are dramatically different. The reason for that is the value of pressure function itself is not as important as the difference in pressure or the pressure gradient. Since the difference in pressure is what causes the fluid to flow from one region to another.

Therefore, the pressure can be different, but the pressure gradient is the same.

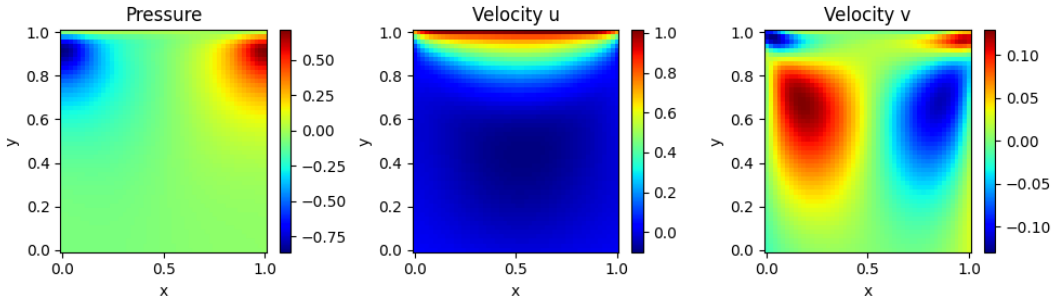

Figure 6: Pressure and horizontal and vertical velocities at $T = 1.0$ for PINN

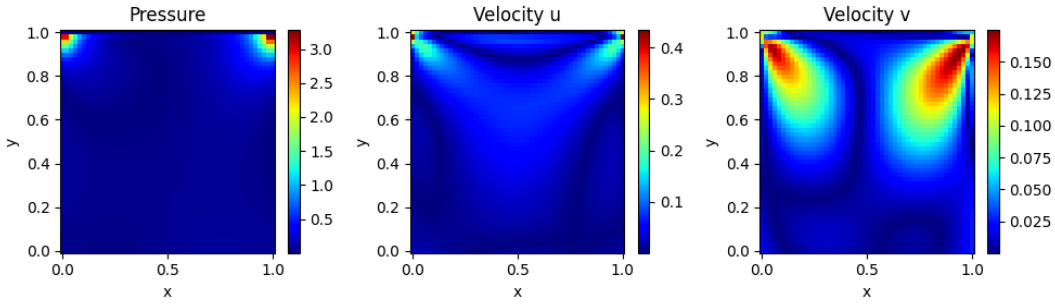

Figure 7: The difference between the results of FDM and PINN

Figure 7 shows the absolute difference between the outputs of the finite difference method and the PINN simulations. We can see the noticeable difference in pressure, but as mentioned earlier, we are more concerned about the pressure gradient in particular. The mean squared error is of the order 10^-2 for the horizontal and vertical velocities.

5 Fluid Flow in Porous media

We have tried to extends our work to another problem in fluid dynamics, which is the problem of fluid flow in 1D in porous media. The problem is given by the following

partial differential equation

$$\frac{\partial}{\partial x}\left(\frac{\beta_c k A}{\mu B}\frac{\partial p}{\partial x}\right) + q_{sc} = \frac{V\phi c_t}{\alpha_c B}\frac{\partial p}{\partial t} \tag{19}$$

Where $p(x,t)$ is the pressure as function of space x and time t, which is the value that we are trying to approximate. α_c, β_c, and B are conversion factors, A is the cross sectional area, μ is the fluid viscosity, and V_b is the bulk volume.

The fluid is subject to the following boundary conditions:

A leakage at the left boundary, which is given by following equation

$$\frac{q_{x0}}{B} = \left(\frac{\beta_c k A}{\mu B}\frac{\partial p}{\partial x}\right)_{x=0,t} \tag{20}$$

Where q_{x0}/B is the leakage rate.

The right boundary is held at constant pressure; That is

$$p(x = L, t) = p_{xL} = 5200 \tag{21}$$

The initial pressure $p(x, 0) = p_i = 5000$

Th domain of the problem is $(x,t) \in [0, L] \times [0, t_f]$. Where we are considering the case $L = 10000$ and $t_f = 50$.

Similar to the procedure that we have performed for the lid-cavity problem, we have assembled points from are the specified domain and fed them to the neural network and computed the losses due to the initial condition, boundary conditions, and the partial differential equation. The resultant losses were huge and the convergence was slow despite the fact that we have used only 1000 points from the domain. After looking into the problem, we have found out that PINNs struggle with approximating large domain problem according to Moseley et al. [2023]. In an attempt to resolve the issue, we have normalized the domain, but that does not seem to fix the problem. Moseley et al. [2023] suggested dividing the large domain into sub-domains and use a

neural network to train each sub-domain, but the solution of each sub-domain must be communicated to the other sub-domains and thus Moseley et al. [2023] proposed the use of overlapping sub-domains. The developed method is known as the finite-basis physics informed neural networks (FBPINNs) and it is inspired by the method of finite element method where the domain is divided into sub-domains or elements. For Further details on FBPINNs refer to reference [9].

Due to the time span of the project, we could not complete this problem, but we have attached the program for the PINN with the project. Any updates in the future can be found on the GitHub repository https://github.com/HashimAlSadah/PETE547_Project.

6 Conclusion

In conclusion, this investigation has effectively employed Physics Informed Neural Networks (PINNs) for the resolution of the system, thereby facilitating the simulation of incompressible fluid behavior within a two-dimensional lid-driven cavity scenario. Through the incorporation of governing physical equations, encompassing partial differential equations (PDEs) and pertinent boundary and initial conditions, PINNs have demonstrated their efficacy in encapsulating the intricate dynamics inherent in fluid flow phenomena. The comparative analysis with the finite difference method (FDM) has elucidated the distinctive advantages inherent in PINNs, particularly with respect to their adept handling of high-dimensional PDEs and mitigation of challenges associated with intricate mesh generation processes. The outcomes underscore the discernible potential of machine learning methodologies, specifically PINNs, in augmenting the precision and computational efficiency of simulations within the realm of scientific computing. As the pervasive integration of deep learning methodologies continues across diverse domains, the outcomes of this investigation contribute substantively to the extant body of knowledge regarding the applicative utility of PINNs in addressing nuanced complexities within fluid dynamics simulations. This project serves to illuminate the broader prospects of PINNs in scientific and industrial domains, indicative of their potential for expansive applicability and advancement.

References

[1] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural Operator: Learning Maps Between Function Spaces. arXiv:2108.08481 [cs, math], December 2021.

[2] Beck, C., Hutzenthaler, M., Jentzen, A., & Kuckuck, B. (2023). An overview on deep learning-based approximation methods for partial differential equations. Discrete and Continuous Dynamical Systems - B, 28(6), 3697–3746. https://doi.org/10.3934/dcdsb.2022238

[3] Grossmann, T. G., Komorowska, U. J., Latz, J., & Sch onlieb, C.- B. (2023). Can physics-informed neural networks beat the finite element method? arXiv preprint arXiv:2302.04107.

[4] Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific machine learning through physics–informed Neural Networks: Where we are and what's next. Journal of Scientific Computing, 92(3). https://doi.org/10.1007/s10915-022-01939-z

[5] Hao, Z., Liu, S., Zhang, Y., Ying, C., Feng, Y., Su, H., & Zhu, J. (2023, March 7). Physics-informed Machine Learning: A Survey on problems, methods and applications. arXiv.org. http://dx.doi.org/10.48550/arXiv.2211.08064

[6] Barba, L. (2013). Lorena A. Barba group. Lorena A Barba Group. https://lorenabarba.com/blog/cfd-python-12-steps-to-navier-stokes/

[7] E, W., & Liu, J.-G. (1995). Projection method I: Convergence and numerical boundary layers. SIAM Journal on Numerical Analysis, 32(4), 1017–1057. https://doi.org/10.1137/0732047

[8] Chorin, A. J. (1968). Numerical Solution of the Navier-Stokes Equations. Mathematics of Computation, 22(104), 745–762. https://doi.org/10.2307/2004575

[9] Moseley, B., Markham, A., & Nissen-Meyer, T. (2023). Finite basis physics-informed Neural Networks (fbpinns): A scalable domain decomposition approach for solving differential equations. Advances in Computational Mathematics, 49(4). https://doi.org/10.1007/s10444-023-10065-9