Project Report

03.03.2023

MACHINE LEARNING METHODS IN MECHANICS

Prateek Vilas Patankar (3509498)

Contents

1. Abstract

Physics Informed Neural Networks or PINNs are a deep learning approach to solving partial differential equations. Numerical methods such as finite difference, volume and element methods are formulated on discrete meshes to approximate the derivatives and are computationally expensive. While purely data driven neural networks are less expensive in the operational stage, they require a lot of training data, to give satisfactory results.
In neural networks, the derivates of activations with respect to previous layers are already available. PINNs use this inherent property to provide differential operations directly, using automatic differentiation.
This report provides details of application of a PINN to solve a case of 2D steady state Navier Stokes Equations to predict the flow in a square cavity.

2. Introduction

This project attempts to build a physics informed neural network to provide a solution the classical Lid Driven Cavity Problem in Fluid Mechanics. It is based on the concept of Physics Informed Neural Networks, or PINNs as described in the work of Raissi et. al. [1]. It attempts to provide a solution for a well posed version of the 2D steady state Navier Stokes Equations, which along with the incompressibility condition, describe fluid flow.
The network does not require the use of flow data for training, as the loss function incorporates the governing equations of the boundary value problem, in the form of the residual loss arising from the main governing partial differential equation, as well as the boundary loss, arising from the boundary conditions prescribed for the lid driven cavity scenario. Such a loss function involves the calculation of the derivatives of the main network outputs.
The networks incorporate a custom layer which calculates the derivates that are required, by automatic differentiation. This is an intuitive solution as the derivatives of the activations for all the layers are already available because of the back propagation process in neural networks.
The network also makes use of a popular convergent optimizer, which used the ‘Limited Memory Broyden–Fletcher–Goldfarb–Shanno’ algorithm, or the L-BFGS algorithm, which is a popular quasi-Newton optimization method.
The results from the network are plotted and compared to a solution calculated by a traditional finite volume method.

3. Problem Definition

3.1. Lid Driven Cavity

The lid-driven cavity, described in the review of Kuhlmann et. al. [2], is an example of an incompressible flow in a confined volume which is driven by the tangential motion of a bounding wall. It is an important fluid mechanical system serving as a benchmark for testing numerical methods and for studying fundamental aspects of solution methodologies.

The following diagram illustrates the domain, along with its characteristic boundary conditions, have been considered in this project:

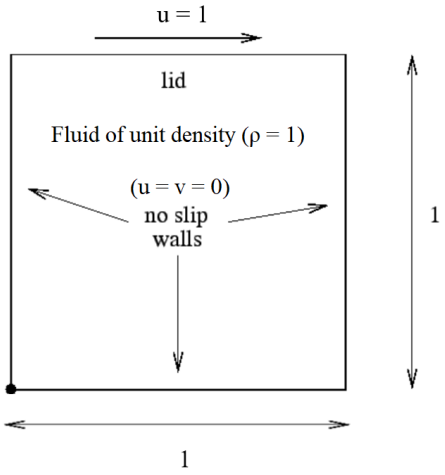


Fig. 1: Lid Driven Cavity Problem where u is x-velocity and v is y-velocity of the fluid

The following points can be significantly notes:

- The domain is a bi-unit square with origin considered at the bottom left corner.
- The walls of the so-called container are the boundaries of the domain, and represent essential boundary, as described in the diagram. They include:
  - The no slip boundary condition the walls i.e., u = v = 0
  - The tangential unit velocity on the lid i.e., u = 1
- The domain is made up of incompressible fluid of unit density.
- The fluid also has a finite kinematic viscosity value 'ν', which was set to 0.01, which gave a Reynold's number value of 100, to the flow.
- Different values of Reynold's number were tested by changing the 'ν' value.

3.2. Boundary Value Problem

Considering the above boundary conditions, the problem, in its well posed form can be expressed as follows:

For Approximating a solution for u(x, y), v(x, y), p(x, y): D → R

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

$$u.\frac{\delta u}{\delta x} + v.\frac{\delta u}{\delta y} + \frac{1}{\rho}.\frac{\delta p}{\delta x} - \nu.\left(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2}\right) = 0 \qquad x, y \in [0,1]$$

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

Subject to:

$$u\,(x,0) = v(x,0) = 0 \quad for\ \text{x}\ \in [0,1]$$

$$u\,(0,y) = v(0,y) = 0 \quad for\ \text{y}\ \in [0,1)$$

$$u\,(1,y) = v(1,y) = 0 \quad for\ \text{y}\ \in [0,1)$$

$$u(x,1) = 1, v(x,1) = 0 \quad for\ \text{x}\ \in [0,1]$$

(1)

3.3. Stream Function Ψ

For incompressible flow, if we define the velocities with respect to stream function Ψ as follows:

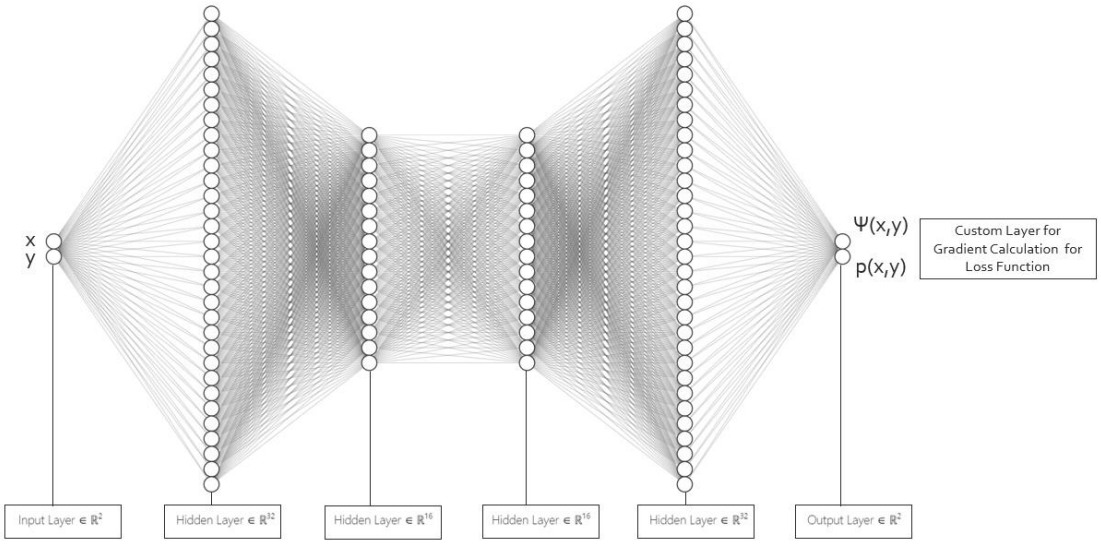
$$\text{u} = \frac{\delta\Psi}{\delta\text{x}}, \qquad \text{v} = -\frac{\delta\Psi}{\delta\text{y}}$$

(2)

This ensures that incompressibility condition is automatically satisfied. Moreover, it also allows us to simplify the network, by only predicting the scalar stream function and then calculating the velocities as derivatives of the result.

Thus, our network will take the coordinates (x, y) as an input, and will give the scalar values of (Ψ, p) as output.

4. Network Architecture



4.1. Choice of Various Network Features

4.1.1. Neurons and Layers:

The following configuration of 4 hidden layers is utilized.

(32, 16, 16, 32)

where, each number represents the number of neurons in a layer.

Recent developments in PINNs such as that detailed by R. Bischof and M. A. Kraus [3], tend to suggest that networks that are shallow but wide (more neurons, less layers) tend to provide a better performance.
The more prominently apparent reason might be that less layers entail less steps in the automatic differentiation process, which probably leads to less noisy gradients.
However, in this case, a wide network with 2 hidden, while faster to train, provided results very different from the numerical solution.
The above-mentioned structure provided reasonable optimum between computation time and accuracy.

4.1.2. Activation Functions:

The choice of activation function is of especially high importance in PINNs, as the differentiability of the function with respect to the inputs is important in calculating the gradients.
Activations with vanishing gradients cannot be used.
Activations such as 'swish' or 'mish' functions which are as follows are more suitable:

$$\boldsymbol{swish(x) = x.sigmoid(x) = \frac{x}{1 + e^{-\beta x}} \quad with\ \beta = 1}$$

$$\boldsymbol{mish(x) = tanh(softplus(x)) = tanh(log\ (e^{x} + 1))} \qquad \textbf{(3)}$$

According to Serengil et. al [4], using swish function as an activation function in artificial neural networks improves the performance, compared to ReLU and sigmoid functions. It is believed that one reason for the improvement is that the swish function helps alleviate the vanishing gradient problem during backpropagation.
Swish seemed to provide better results than mish.

4.2. Definition of loss function:

The loss function of a PINN is what provides it with the ability to, so to speak, respect the underlying physics of the problem. As such, it has to include:

1. Information about the governing differential equation/equations, in the form of their residuals, which can be minimized during training. This is incorporated as the residual loss. In our case, the residual loss will be the steady state Navier Stoke's equation expressed in 2 dimensions. They would be as follows:

$$\text{residual in x} \Rightarrow u * u\_x + v * u\_y + p\_x/self.rho - self.nu * (u\_xx + u\_yy)$$

$$\text{residual in y} \Rightarrow u * v\_x + v * v\_y + p\_y/self.rho - self.nu * (v\_xx + v\_yy)$$

Note that the continuity condition is already satisfied for an incompressible flow in the consideration of a stream function interpretation of the equation.

2. Information about the initial and/or boundary condition, constrained to which the differential equation is acting.

All the derivatives (u_x, u_y, u_xx etc.) will be calculated in a custom layer, by automatic differentiation from the backpropagation process of the constructed network.

5. Training

5.1. Training Database:

For a purely physics informed neural network, which does not rely on labels generated from previously established databases, the loss function is directly responsible for steering the training of the neural network, by making it abide by the built-in residual and boundary constraints.

5.1.1. Training Input

The training input consists of points in the domain, in this case in 2D space. Popular literature states that it is suitable to have equal number of points for the interior of the domain, where the residual needs to be satisfied, and the boundary, where the boundary conditions need to be satisfied. During testing of the network, it was seen that about 15000 training points, on both the interior and the boundary, needed to be provided, for the solution to have sufficient similarity with the numerical solution. The selection of the points was randomized for the training stage. A strategic sampling of training points would most probably yield better results.

5.1.2. Training Output

Since the boundary conditions are enforced in terms of velocity, but the network output is in terms of the stream function, the output training data has to be modified to the correct dimensions. It is essentially just null tensors, but with the exception of the boundary points where y = 1, where there would be a tangential velocity value

5.2. Testing Database:

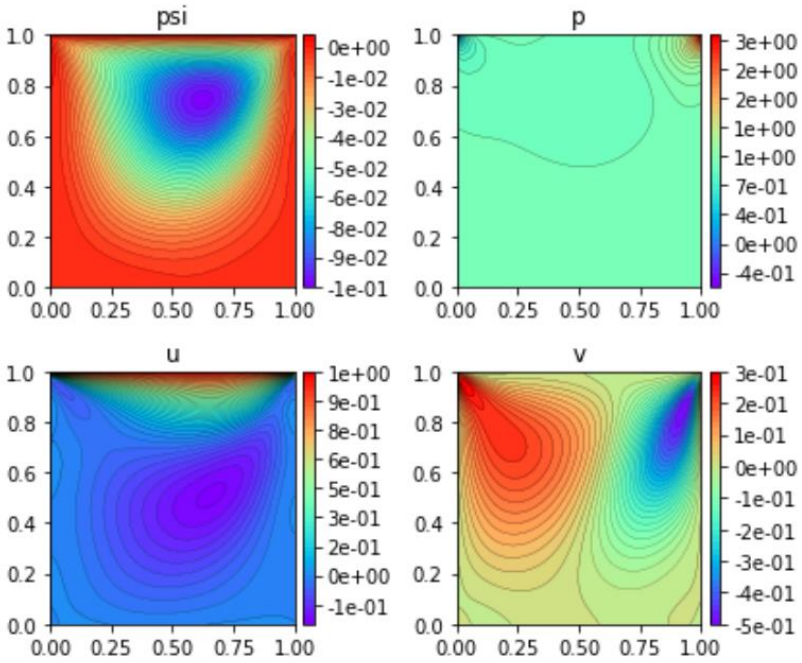
The testing database is the database that provides coordinates at which the values will be predicted and will be plotted. As such it is simply a uniform grid of appropriate density that will provide a sufficiently detailed output plot.
A grid of 100 x 100 was sufficient to show the results correctly.

5.3. Choice of Optimizer:

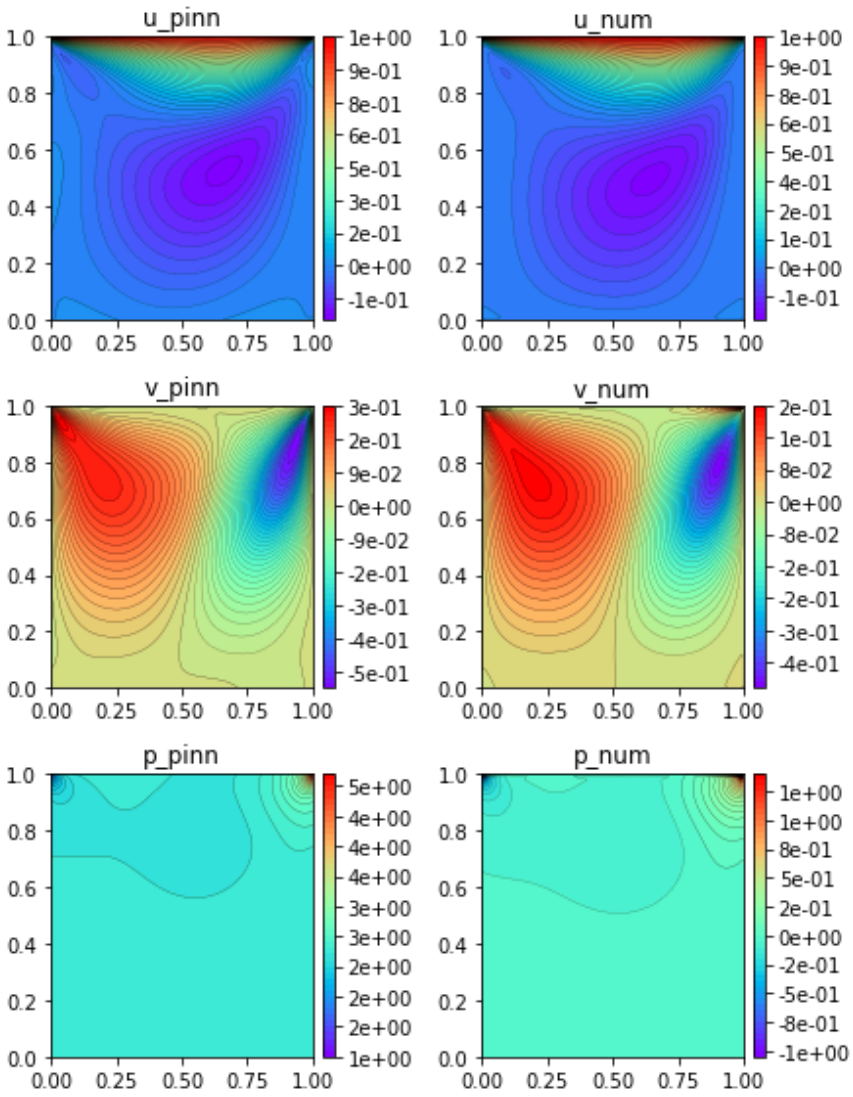
The training was initially done using the standard Adam optimizer. However, as pointed out by Raissi [1], quasi Newton methods, which use an approximation of the Hessian matrix to compute the gradient updates, are much better suited as convergent optimizers for PINNs.
As such, a L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) type convergent optimizer, with a logcosh type activation was utilized, and provided better results.
Further work can be done to fine tune the optimizer in order to achieve better results.

6. Results

- Results were plotted as contour plots for the fluid flow inside the cavity, for the values of Ψ(stream function), p (absolute pressure), u (x velocity) and v (y velocity) as the outputs of the PINN, and were as follows:



- For a Reynold's Number of 100, the u, v, and p fields were compared to a numerical solution obtained by a finite volume approach, using Chorin's projection. They show that the results of the neural network capture effects of the fields with highly comparable accuracy to that of the numerical method, thereby validating the concept.



7. Conclusions

The results show that, at low Reynold's numbers values, the PINN provides a good steady state approximation of an incompressible flow in a square cavity, by capturing the field quantities and the effects such as the vorticity of the flow, thereby validating the concept. To extend the effectiveness of the network to higher Reynold's numbers, as well as unsteady state transient results, further work needs to be done. Some important areas would be:

- Strategic sampling of training data
- Further work into the optimization process
- To do an ensemble of multiple networks, with each having its own activation function, and then aggregate their predictions. This can be done with the use of a gating network that weights each PINN's contribution based on the input and has been a popular approach [3]
- By using CNN style residual connections, to better facilitate the flow of gradients.

8. References

1. M. Raissi, et al., Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, arXiv: 1711.10561 (2017).

2. Kuhlmann, Hendrik & Romanò, Francesco. (2018). The Lid-Driven Cavity. 10.1007/978-3-319-91494-7_8

3. R. Bischof and M. A. Kraus, “Mixture-of-Experts-Ensemble Meta-Learning for Physics-Informed Neural Networks”, Proceedings of 33. Forum Bauinformatik, 2022

4. Serengil, Sefik Ilkin (2018-08-21). "Swish as Neural Networks Activation Function". Machine Learning, Math. Archived from the original on 2020-06-18. Retrieved 2020-06-18.