EG2605: UNDERGRADUATE RESEARCH OPPORTUNITIES PROGRAMME (UROP)

PROJECT REPORT

Physics Informed Machine Learning Surrogates for Predicting Viscous Magnus Flow Characteristics

Arijit Dasgupta

Department of Mechanical Engineering, Faculty of Engineering

National University of Singapore

AY 2019/20 Semester 2

Abstract

This work aims to predict the two-dimensional viscous flow field of the flow past a cylinder rotating at a constant rate of rotation in an incompressible flow (termed as the viscous Magnus Flow) using a Physics-Informed Neural Network (PINN) which is trained using data from a Computational Fluid Dynamics (CFD) simulation of the flow. The flow is numerically simulated using an open source CFD software tool *OpenFOAM* version 5 at different Reynolds numbers and cylinder rotational speeds, generating the data for the PINN. The PINN model is trained to obey the governing Navier-Stokes equations as a penalising regularisation loss using automatic differentiation in the open source machine learning software tool *TensorFlow* in which the PINN is developed. Hyperparameter tuning of the PINN is done using the flow past a stationary cylinder. These hyperparameters are then adopted to train the PINN to predict the velocity and pressure flow fields at a range of Reynolds numbers and cylinder rotational speeds. The PINN is also shown to have better performance than a neural network (NN) which is not physics based but trained with CFD Data, especially with scarce data. The influence of other parameters such as training data and collocation data size to improve the performance of the PINN are also considered. Finally, a Magnus PINN is developed that can predict the flow field for a given Reynolds number.

Keywords: *Physics-Informed Neural Networks, Rotating Cylinder, Predictive Modelling*

Table of Contents

Acknowledgements

I would like to express my deep gratitude to Professor Murali Damodaran, my research supervisor, for his constant patience and guidance to me throughout the duration of this project. He has taught me a lot from his extensive knowledge of fluid mechanics and given many words of advice on conducting proper research methodology. I will continue to follow his teachings in my future research endeavours. I would also like to thank Professor Khoo Boo Cheong for his support in arranging this project for my UROP.

1. INTRODUCTION

The ability of deep neural networks to act as effective function approximators has been well known for decades. In recent times, the merging fields of scientific computing and machine learning, namely 'scientific machine learning' has gained more popularity with its increasing effectiveness in scientific predictions. In particular, the present study is interested in implementing the Physics-Informed Neural Network (PINN) proposed by Raissi [1]. The general aim of a predictive model in this context is to reliably predict the Quantities of Interest of thermofluids of any flow scenario. A normal feed-forward Neural Network (NN) can be trained using computational, analytical or experimental data to predict quantities like pressure and velocity flow fields. A PINN follows the same ideology, but the loss function further penalises the network for not conforming to its governing equations. As the inputs and outputs are functionally related by activations, weights and biases, the partial differentials for any governing equation can be determined using chain rule, a method commonly known as automatic differentiation as outlined by Hoffman [2]. By adding this term to the loss function, the network will backpropagate to reduce this loss, hence constricting the Quantities of Interest to the physics of the flow. It has been shown that a PINN requires less training data than a feed-forward NN to accurately predict the flow. Given the expensive computational costs of computer simulations, such an advantage is of great interest.

The classical flow past a rotating cylinder in an incompressible uniform flow is known for the generation of a lift force on the cylinder, commonly termed as the Magnus effect [3-4]. The flow problem of interest in this study is the viscous flow past a rotating cylinder, termed here viscous Magnus Flow. The viscous Magnus flow is a classic problem that has been well studied by researchers [5-7].

The aim of this study is to develop a PINN model that can predict the viscous Magnus flow field. A hyperparameter tuning is first conducted to significantly improve the performance of the model by tweaking the hyperparameters. The tuning of the hyperparameters is done for a non-rotating cylinder at Re_D of 100,000 and the tuned hyperparameters are used to train the PINN for an entire dataset with varying α and fixed Re_D. The performance of the model is further improved by varying the number of training data points used.

2. METHODOLOGY

2.1. Computational Flow Modelling

Viscous Magnus flow is computed numerically using the incompressible and viscous Navier-Stokes equations, the non-dimensional integral form of which is expressed as follows:

Continuity Equation: $$\iiint_{\Omega}\left(\vec{\nabla}\bullet\vec{U}\right)d\Omega=\iint_{S}U\bullet\vec{n}dS=0 \tag{1a}$$

Momentum Equations: $$S_t\frac{\partial}{\partial t}\iiint_{\Omega}\vec{U}d\Omega+\iint_{S}\left(\vec{U}\vec{U}+p\right)\bullet\vec{n}dS=\frac{1}{Re_D}\iint_{S}\left(\vec{\nabla}\vec{U}\right)\bullet\vec{n}dS \tag{1b}$$

where $\vec{U}$ is the velocity and p is the pressure. The flow field depends on the Reynolds number of the flow, Re_D based on the diameter D of the cylinder i.e. $Re_D=\frac{DU_\infty}{\nu}$ where ν is the kinematic viscosity of the fluid and U_∞ is the freestream inflow velocity of the flow. The non-dimensional spin ratio of the cylinder, α, defined as $\alpha=\frac{D\omega}{2U_\infty}$ where ω refers to the angular velocity of the rotating cylinder. In Equation (1b), $S_t=\frac{fD}{U_\infty}$ is the Strouhal number based on the length scale D, which is the characteristic length. U_∞ is the characteristic velocity and the frequency $f=\frac{1}{T}$ where T is the characteristic time.

The non-dimensional differential form of Equations 1(a)-(b) for the two-dimensional viscous Magnus Flow problem are as follows:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2a}$$

$$St\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re_D}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{2b}$$

$$St\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re_D}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{2c}$$

where u and v are the non-dimensional *x*- and *y*- components of velocity in a Cartesian coordinate system. Equations 2(a)-(c) will be used to develop the PINN surrogate for predicting viscous Magnus flow.

The computational domain for the viscous Magnus flow simulation is shown in Fig 1.

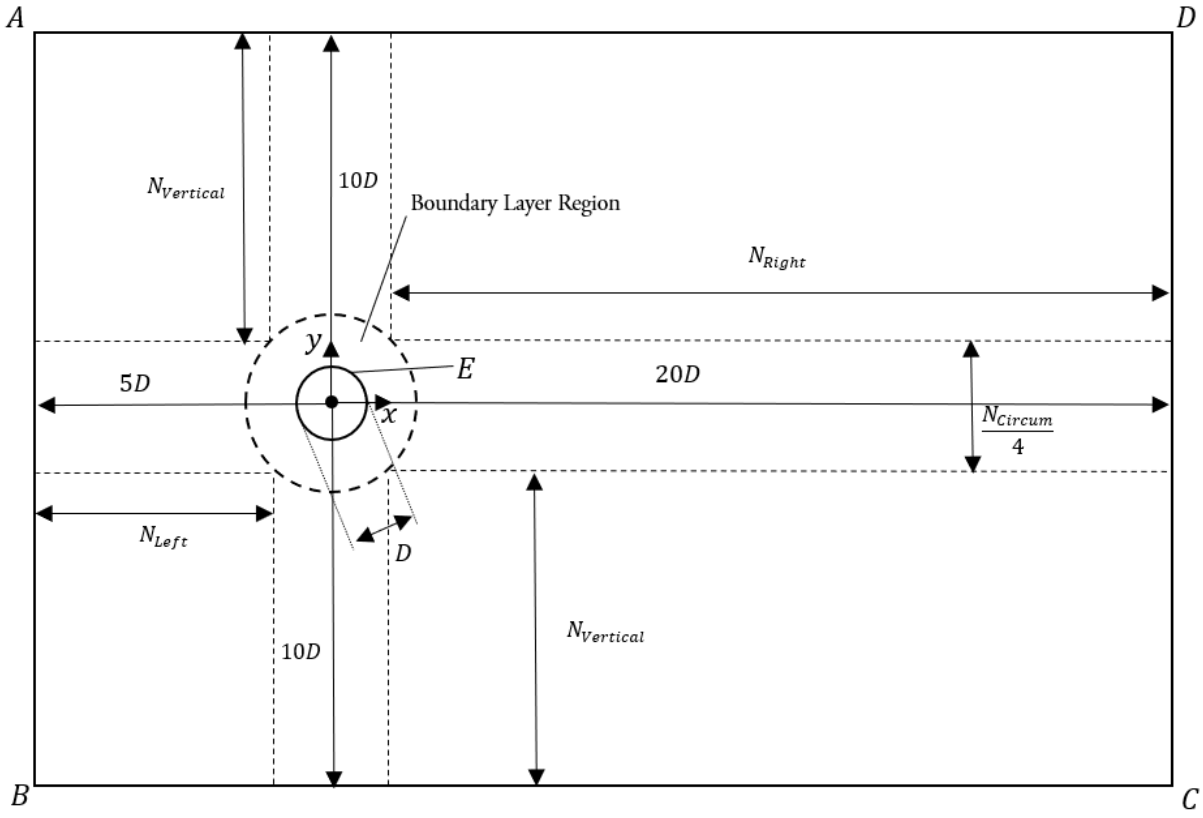
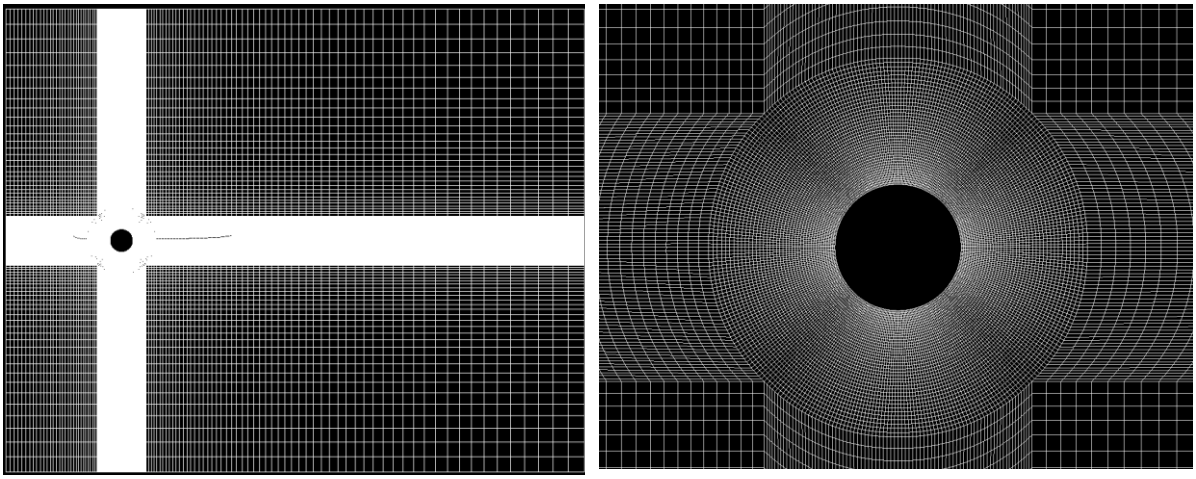


Fig 1: The computational domain of the Magnus effect simulation with nodal dimensions. A, B, C & D represent the corner vertices of the domain while E is the cylinder circular boundary.

The mesh for the flow simulation is developed using the *blockMeshDict* function in *OpenFOAM*. The mesh can be split into two main regions. The first region is labelled as the boundary layer region as shown in Fig 1. This region (with a diameter of 3*D*) is located at the circumferential region outside the cylinder where the boundary layers are captured. An O-grid is used and mesh nearest to the cylinder is the finest with an expansion of 2 from the finest cell to the coarsest cell. Fig 1 also illustrates the mesh domain with the respective defined nodal dimensions of N_{Left}, N_{Right}, $N_{Vertical}$, N_{Radial} & N_{Circum}. The number of radial nodes is defined as N_{Radial} and the number of circumferential nodes is defined as N_{Circum}. These former three dimensions are described by a rectangular structured grid. The wake (downstream) region, upstream region, top and bottom of cylinder follows the O-grid mesh of the boundary layer region with a width of $\frac{N_{Circum}}{4}$ points in each region. The mesh cell size in the region outside the boundary layer region also expands vertically and horizontally away from the cylinder. To

determine the grid to use, a grid convergence is conducted with the drag coefficient, C_D as the convergence parameter. The flow parameters are α=0 and Re_D = 140,000. In the first 8 meshes, N_{Radial} and N_{Circum} is increased from 5 to 40 in intervals of 5. This increased the number of cells from 4,600 to 25,600. Afterwards, with 6 more meshes, the number of cells in the second mesh region is increased from $N_{Vertical} = 30$, $N_{Left} = 30$ & $N_{Right} = 60$ to $N_{Vertical} = 60$, $N_{Left} = 60$ & $N_{Right} = 180$ to the number of cells to 70,400. All details are tabulated in Table 1 in Appendix A. Fig 2(a) shows the structure of the mesh used for computation and Fig 2(b) shows a close-up view of the mesh in the vicinity of the cylinder. It is visible that the O-grid around the cylinder is relatively much finer than the region outside of it. The mesh along the cylinder's wake and the inflow projected onto the cylinder along with the top and bottom of the cylinder is considerably finer than the rest of the domain.



(a) (b)

Fig 2(a): Mesh of the domain (Full view) (b): Mesh of domain (Close-up of Cylinder)

Fig 3 shows the variation of the drag coefficient C_D with $\frac{1}{N}$ for the meshes defined in Table 1 in Appendix A. The C_D reduces rapidly from 1.12 for the coarsest mesh and converges to about 0.30 as the mesh is refined. To optimise computational resources and simulation accuracy, mesh 9 from Table 1 in Appendix A with 30,200 cells ($\frac{1}{N} = 3.31e-05$) has been deemed adequate for this study.

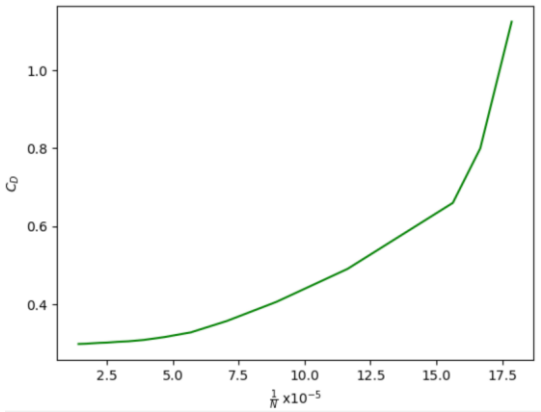


Fig 3: Grid Convergence of C_D with $\frac{1}{N}$ x (10^{-5}) where N refers to the number of cells

All flow simulations are executed in *OpenFOAM version 5* [8] by solving Equations 1(a)-(b) using the in-built *pimpleFoam* solver. As the flow is expected to in the turbulent Re_D regime for large Reynolds Numbers, the k-kL-ω transitional RANS model outlined in Walters & Cokljat [9] is used.

The upstream region is on the left with the fluid inflow towards the positive x-direction. The origin of the domain is located at the centre of the cylinder. From Fig 1, the boundary represented at AB is the 'Inflow' and CD is the 'outflow'. The AD and BC boundaries are the 'top' and 'bottom' respectively while E is the cylinder 'wall'. All boundary conditions can be represented mathematically, defined by either a pressure or velocity constraint. In the present case, $\overrightarrow{U_\infty}$ is along the x-axis, $\vec{n}$ refers to the unit normal vector of the boundary and $\vec{r}$ is the radial vector of the cylinder. The boundary conditions are defined as follows:

Inflow boundary condition at inlet (AB): $\vec{U} = \overrightarrow{U_\infty}$

Far field velocity boundary condition at top (AD): $\vec{U} = \overrightarrow{U_\infty}$

Far field velocity boundary condition at bottom (BC) $\vec{U} = \overrightarrow{U_\infty}$

Outflow zero pressure gradient boundary condition at outlet (CD): $\vec{\nabla} p \cdot \vec{n} = 0$

No slip boundary condition at the cylinder wall (E): $\vec{U} = \vec{\omega} \times \vec{r}$

The no slip boundary condition at the wall is implemented using the *rotatingWallVelocity* boundary function in *OpenFOAM*. By stating the centre coordinate of rotation, the desired ω and defining the cylinder wall patch, the abovementioned mathematical relation is implemented. Hence the fluid velocity at the wall is equal to the tangential velocity of the cylinder wall along all points of the wall. To generate numerical data for the PINN, the simulations are run from Re_D = 100,000 to Re_D = 200,000 in intervals of 10,000. For each Re_D, there are 11 cases with α=0 to α=1 in intervals of 0.1. For every Re_D, the flow simulations for α >0 uses the steady flow field from the α =0 case as the initial conditions. The rotation of the cylinder at the specific value of angular rotation is not applied abruptly, but rather ramped linearly from initial value which is zero up until the desired speed is reached. The ramp function is defined as $\omega = \frac{2t\alpha U_\infty}{t_0 D}$ when $t \leq t_0$ and $\omega = \frac{2\alpha U_\infty}{D}$ when $t > t_0$, where t is the simulation time and t_0 is a user specified time taken for the cylinder to ramp up to the desired in rotational speed. Hence with increasing time steps, the ω defined using the *rotatingWallVelocity* boundary function is incrementally increased. For the present study t_0 is set to 5s.

As *OpenFOAM* is used for the simulations, the numerical scheme used is the finite volume method. The time and divergence schemes are Euler and Gauss upwind respectively. The Laplacian scheme employs a Gauss discretisation scheme and a linear interpolation of the diffusion term in the momentum equation. The interpolation scheme for cell centres to face centres is linear. After the completion of all simulations, a python code is developed and used to cleanly extract the pressure and velocity data from all 121 cases into a *.mat* file meant for data storage.

2.2. PINN Setup

The PINN in the present study follows the structure of a feed-forward deep neural network that relies on multiple neurons per layer and multiple layers per network. The goal is to create a functional relationship between the desired inputs and outputs that can accurately predict all sets of input and outputs of the viscous Magnus flow flow field. All the neurons in each layer are fully connected to the neurons in the subsequent layer in the present study. This network makes use of weights, biases & activation functions at every neuron in creating the functional relationship.

2.2.1. PINN Formulation

Fig 4 shows the general structure of a PINN, where a simple architecture of 2 hidden layers with 5 neurons is shown. Each training data fed into the PINN consists the u, v and p values of a particular (x, y, t) point. The input layer consists 3 neurons that hold the values of x, y and t respectively. The neurons in the hidden layers contain a value known as the activation.

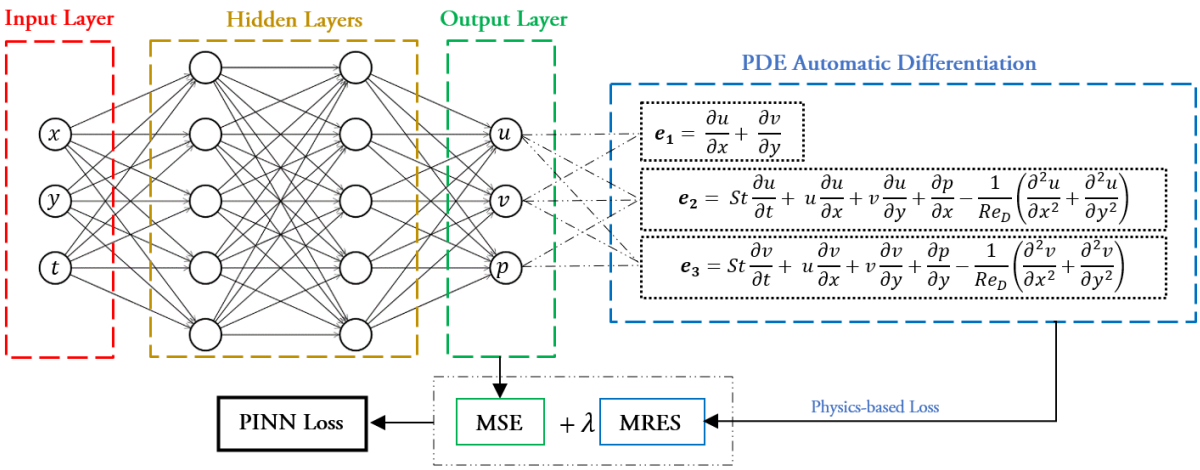


Fig 4: The structure of a [3, 5, 5, 3] PINN with the calculation of PINN Loss

The activation, a, is defined in Equation (3) as: -

$$a_i^{[l]} = \chi\left(\sum_{j=1}^{N}\left[w_{i,j}^{[l]} a_j^{[l-1]}\right] + b_i^{[l]}\right) \tag{3}$$

The i and $[l]$ indexes refer to the ith neuron of the lth layer in consideration. The j index refers to the jth neuron of the previous ($[l-1]$) layer. Each neuron of the previous layer is multiplied by its respective weight, $w_{i,j}^{[l]}$, and summed. The bias term, $b_i^{[l]}$ is added to the summed value. The χ function refers to an **activation function**, which can be considered as a hyperparameter. In the present study, all the neurons with the relation in Equation (3) use the same activation function to reduce the complexity of the PINN formulation. In the same manner as Equation (3), the neurons in the output layer obtain their activations. In the present study, the output layer consists 3 neurons that store the predicted values of u, v and p respectively. This entire process is known as forward propagation. Afterwards, the predicted values and ground true values are compared in a loss function. A commonly used loss function that is adopted in the present study is the Means Squared Error (MSE) loss, as described in Equation (4a): -

$$MSE = \sum_j^3 \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \left(y_{j,i}^{actual} - y_{j,i}^{predicted}\right)^2 \tag{4a}$$

N_{train} is the training data size and is meant to represent the size of the available numerical data from CFD. The index j in Equation (4a) refers to one of the three output values (u, v, p) and the i index refers to the ith training point out of N_{train} points. $y_{j,i}^{actual}$ is in accordance to the ground truth as given by the viscous Magnus flow simulation data and $y_{j,i}^{predicted}$ is in accordance to the prediction from the output layer neuron. The squared error for u, v and p are summed and divided by N_{train} to get the MSE loss.

However, as shown in Fig 4, there is an additional step to the loss function calculation which is the key step which distinguishes a PINN from a standard NN. The PINN uses automatic differentiation [2], a technique that takes advantage of the chain rule of the functional relationships in the network, to calculate spatio-temporal values of partial derivatives.

The advantage of this technique is that the residuals of the incompressible Navier-Stokes equations at any node for any given x, y and t can be computed from Equation (2) as follows for each data point:

$$e_1 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$$

$$e_2 = St\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \frac{1}{Re_D}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{4b}$$

$$e_3 = St\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \frac{1}{Re_D}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right).$$

The residuals computed with the output values from the NN portion of Fig 4 will not be zero. For capturing the correct physics these residuals should be zero. For this stage, there is an additional data size introduced as the collocation size (N_{col}), with $N_{total} = N_{train} + N_{col}$. N_{col} represents the points where numerical data may not be available, but still represent valid points on the spatio-temporal domain where the governing equations are applicable. This representation is used to simulate the scenario where numerical data may be scarce. This is necessary to show that PINNs can provide solutions as accurate as NNs with a lower N_{train}. Note that all N_{train} & N_{col} data points are **randomly extracted** from the spatio-temporal domain without replacement. The loss from the automatic differentiation step is shown by the calculation of the Mean Residual Error Squared (MRES). The three residuals are squared, summed over the entire N_{total} dataset, and then averaged.

$$MRES = \frac{1}{N_{total}}\sum_{i=1}^{N_{total}}[(e_1)^2 + (e_2)^2 + (e_3)^2]_i \tag{4c}$$

This implies that the residuals are not only determined at the randomly selected collocation points, but also at the randomly selected training points. This is also referred to as the physics-based loss, as illustrated in Fig 4. The total loss function for PINN is then defined as follows:

$$\text{PINN Loss} = MSE + \lambda(MRES) \tag{4d}$$

In the present study, the λ factor in Equation (4d) is set at either 0 or 1, where 1 refers to the activation of a PINN and 0 refers to a deactivation to a NN. Using the chain rule relation through the network, the partial differential of the loss function with respect to all weights and biases is formulated. This step is known as back propagation. The calculated partial differentials are multiplied by the **learning rate**, another hyperparameter, which determines an 'update quantity' for each weight and bias. This process is repeated through all N_{total} data points (formally known as an epoch). The average of each 'update quantity' of the N_{total} dataset is taken, and all weights and biases are updated by subtracting their respective 'update quantities'. The aim of the update is to shift the weights and biases in a direction that reduces the calculated PINN loss and hence improves predictive performance. This manner of reducing the loss is known as gradient descent. The manner of gradient descent can also be influenced by any **optimisation algorithm** used. The optimisation algorithm can be considered as another hyperparameter. In the present study, the Adam optimisation algorithm developed by Kingma and Ba [10] is initially adopted. The entire training process is also affected by the number of hidden layers and number of neurons in each of these hidden layers. Therefore, the **network architecture** is another hyperparameter of interest.

2.2.2. Hyperparameter Tuning Process

The hyperparameters introduced in Section 2.2.1 cannot be simply set randomly. The hyperparameters must be carefully tweaked to extract further performance from any PINN surrogate. Hence, the first stage of the PINN requires a hyperparameter tuning process to ensure that the model hyperparameters are optimized to have the loss in the model. In the present study, the hyperparameter turning is

conducted on one case of Re_D=100,000 and α=0 using the data from the viscous Magnus flow CFD. The data is extracted from the stored *.mat* file mentioned in Section 2.2.1. 11 consecutive time steps are extracted from the numerical data, forming a mother dataset of size 11 × 30,200 = 332,200. The model weights and biases are initialised by the method proposed by Xavier [11]. In this process, all models are trained at 20,000 epochs using the *TensorFlow* platform of Abadi et al. [12]. The training of the model follows the formulation laid out in Section 2.2.1. The values of N_{train} and N_{col} are set to 1000 and 200 respectively, a mere 0.36% of the mother dataset. As all the points are randomly chosen, none of the data points have any relation with each other. For every training instance, the test MSE and test MRES are calculated using the full mother dataset to determine the performance of the model. The test MSE is taken as the key metric used to evaluate the model overall performance while the MRES is taken as the measure of the extent to which the model predictions comply with the physics of the problem.

The present study considered four hyperparameters for tuning in the following hierarchical order (i) Activation Function (ii) Learning Rate (iii) Optimizer Combination and Network Architecture. At every step in the hierarchy, the hyperparameter associated with the best performing model is used. After all the hyperparameters are selected, the performance of the PINN is compared with the performance of NN using the same tuned hyperparameters to show the benefit of using PINN surrogates. Subsequently, the N_{train} and N_{col} are varied and tested to evaluate its influence on the model performance and potentially extract further predictive performance.

2.3. Magnus PINN

The final aim of the present study is to extend the predictive capabilities of the PINN developed in the hyperparameter tuning process and extend it to a range of Re_D and α cases. This PINN surrogated is termed as Magnus PINN in the present study. The present study has a Re_D range of 100,000 to 200,000 and a α range of 0 to 1, in accordance with the numerical data generated by the viscous Magnus flow simulations. The goal of the Magnus PINN is to predict the flow field at any α (in the mentioned range) for a given Re_D. The main difference between the Magnus PINN and the methodology in Fig 4 is that α is an additional input neuron to the input layer.

During the training of the PINN, the assumption is made that the flow field being predicted has the same Re_D as it is used when computing the e_2 and e_3 terms of the MRES. The mother dataset for the Magnus PINN has a size of 30,200 × 11 × 11 = 3,654,200. After developing the Magnus PINN, the present study will also test its ability to interpolate and extrapolate at different α not used in the training process.

3. RESULTS AND DISCUSSION

3.1. Hyperparameter Tuning

Prior to the commencement of hyperparameter tuning, the following hyperparameters are set to default as follows:

i. Activation Function: *Sigmoid*
ii. Learning Rate: 0.001
iii. Optimizer Combination: 100% *Adam*
iv. Network Architecture: [3, 10, 20, 10, 3]

Effect of Activation Function: Four commonly used activation functions are the *sigmoid*, hyperbolic tangent (*tanh*), rectified linear unit (*Re-Lu*) and the leaky rectified linear unit (*leaky Re-Lu*). Fig 5 illustrates the relationship between the loss and epoch of the PINN training. Fig 5(c)-(d) shows that the MSE loss plateaus early at an order of magnitude lower than *sigmoid* and *tanh* in Fig 5(a)-(b). Additionally, the test MRES worsens with increasing epochs for the *Re-Lu* and *leaky Re-Lu* activation

functions. Both the *sigmoid* and *tanh* activation functions provide a relatively low test MSE loss of 3.070e-03 and 2.850e-03 respectively in comparison with *Re-Lu* and *leaky Re-Lu*.

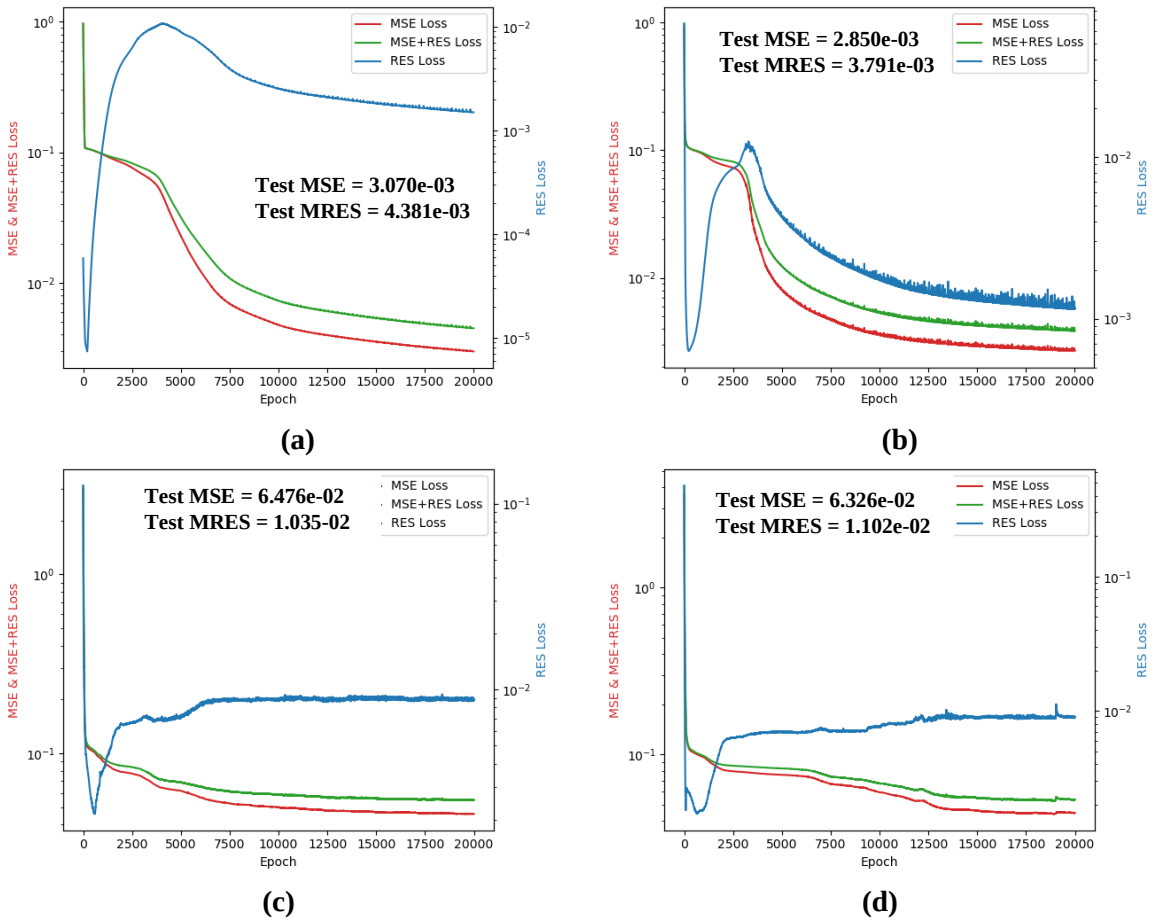


Fig 5: Loss against Epoch for (a) Sigmoid, (b) Tanh, (c) Re-Lu, (d) Leaky Re-Lu

One can notably visualise this difference in performance from the 2D plots of velocity magnitude in Fig 6. The *sigmoid* and *tanh* activation functions showed a much better match with the CFD viscous Magnus flow simulation. Nonetheless, the *tanh* activation function described the low-speed wake region more accurately than the *sigmoid* activation function. The *Re-Lu* and *leaky Re-Lu* models hardly showed a visual match with the numerical contours as they incorrectly predicted a region of low velocity upstream of the cylinder. As the *tanh* model gave the lowest test MSE loss, it is selected as the optimized activation function.

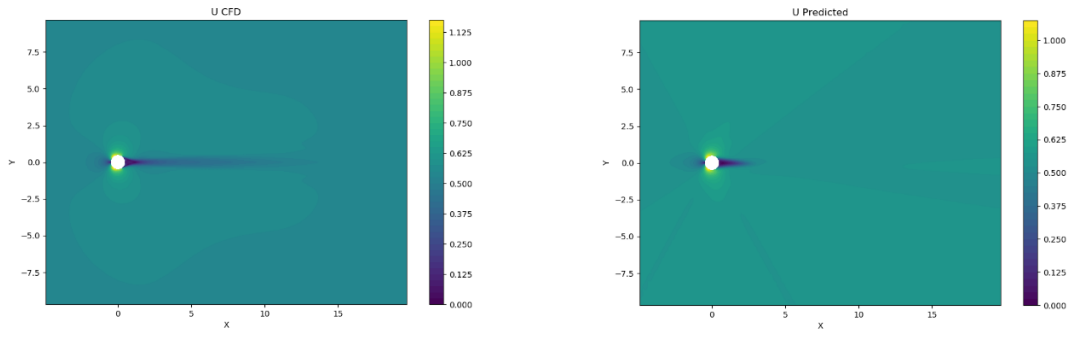
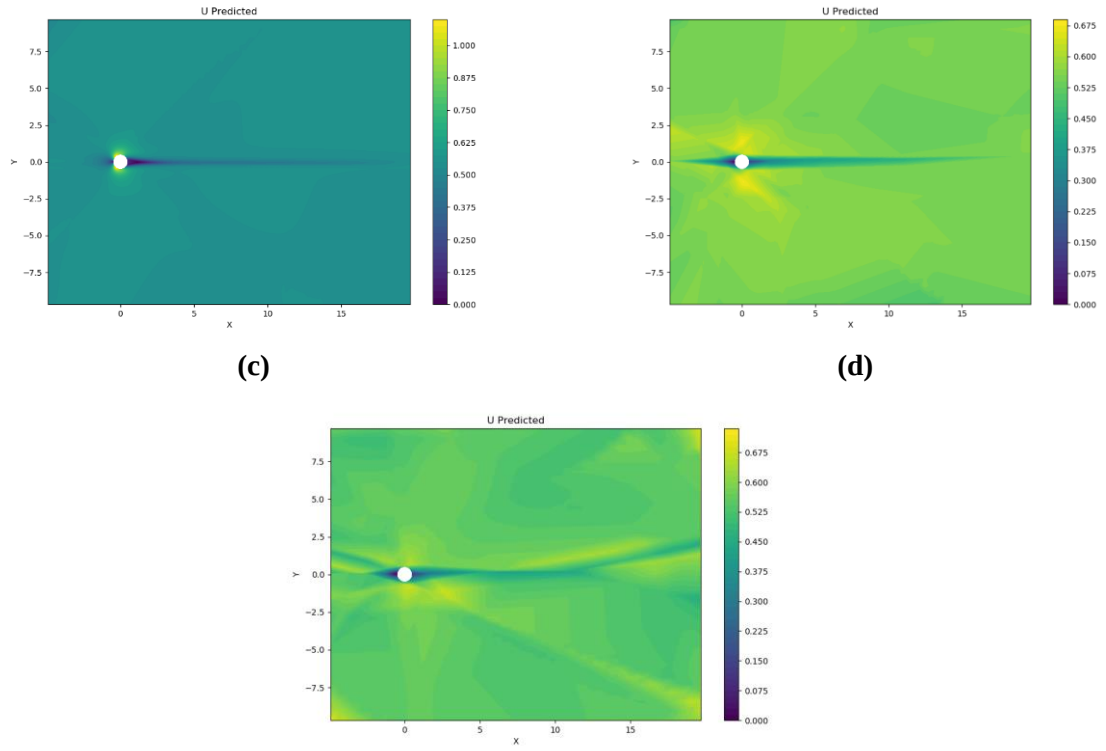


Fig 6 (continued): Velocity magnitude plots for (a) CFD (b) Sigmoid

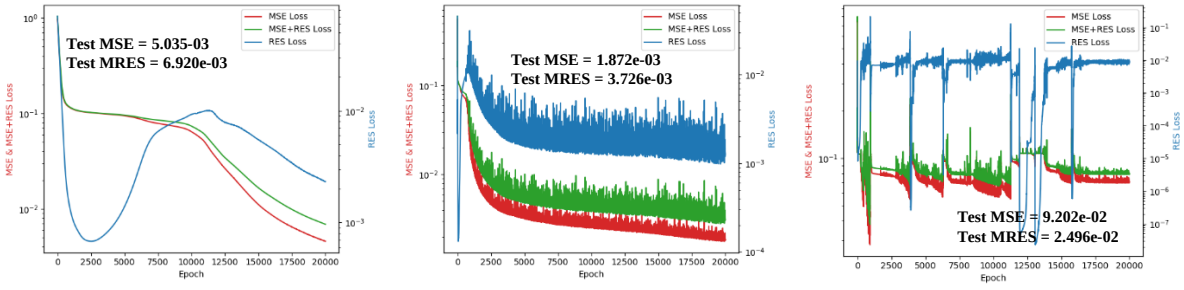


(c) **(d)**

(e)

Fig 6: Velocity magnitude plots for (c) Tanh (d) Re-Lu (e) Leaky Re-Lu

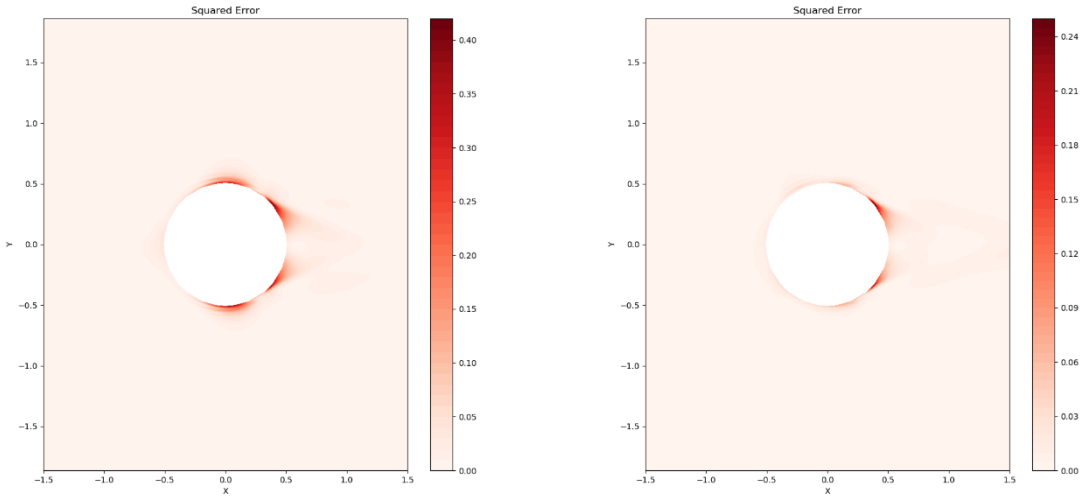
Effect of Learning Rate: The learning rate is a crucial parameter that affects the rate of update of weights and biases. A learning rate too small may lessen the risk of missing the global minima of the loss function, but the process becomes too slow and risks being caught in a bad local minimum. A learning rate too high would speed up the process, but the model may skip or miss the global minima and risk being trapped in a bad local minimum. In the first step, the learning rate is varied in the following list: (0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1), spanning 4 orders of magnitude.



(a) **(b)** **(c)**

Fig 7: Loss against Epoch for learning rates (a) 1e-04 (b) 0.01 (c) 0.05

The tests reveal that the test MSE loss improved with increasing learning rate from 1e-04 until it hit an optimum 1.872e-03 at a learning rate of 0.01, before significantly worsening in performance with further increases in learning rate. This is evident from the seemingly chaotic plot in Fig 7(c) for a learning rate of 0.05. The learning rate of 0.1 gave similar results. Fig 7(a) shows that the lowest tested learning rate of 1e-04 led to smooth changes in the loss, however the MRES only started to drop after 10,000 epochs, which is undesirable. The learning rate 0.01 has the most substantial drop in both MSE and MRES losses despite the immense fluctuations.

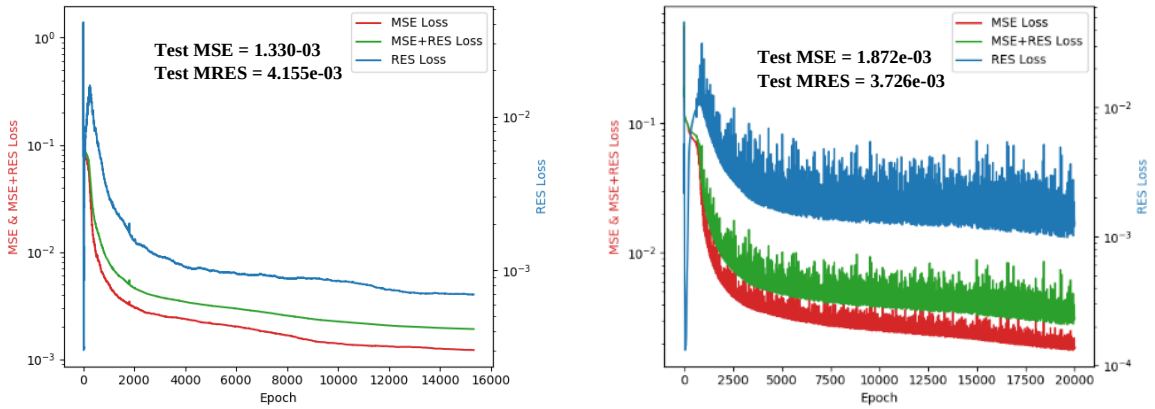


(a) **(b)**

Fig 8: Spatial Distribution of MSE (closeup) at learning rates (a) 1e-04 (b) 0.01

Analysing the spatial distribution of the MSE and MRES errors will allow for a better localising of the There is a significant error region in the boundary layer at the top and bottom of the cylinder after visibly analysing the spatial distribution of the MSE for the learning rate of 1e-04 in Fig 8(a). There is also a similar error accumulation on the wake-side of the cylinder. In contrast, the learning rate of 0.01 in Fig 8(b) only has a wake-side error region present at a lower magnitude than the former learning rate. Therefore, a learning rate of 0.01 is adopted as it optimised the minimisation of the error regions at the boundary layers.

Effect of Optimizer Combination: Optimizers are responsible for reaching the global minima of the loss function as efficiently as possible. Algorithms like *Adam* slow down the fluctuating steps in gradient descent by implementing momentum terms, allowing the model to converge to the global minima faster. In the present study, a competing optimization algorithm, *L-BFGS-B*, a box-constraint variant of the limited memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [13] is tested. As seen in Fig 9, the *L-BFGS-B* algorithm provided a smoother drop in MSE and MRES loss and a better overall test MSE loss of 1.330e-03. The algorithm in Fig 9(a) automatically converged after 15,331 epochs. The test MRES loss for both optimizers are relatively similar.



(a) **(b)**

Fig 9: Loss against Epoch plots for optimizer type (a) L-BFGS-B (b) Adam

Another key observation to be addressed from Fig 9 is that the MRES drops significantly at low epochs, regardless of its trend afterwards. Note that this trend is present in Fig 7 as well. This is expected as the Xavier initializer used meant that all initial weights and biases are low and equal in magnitude and thus the predicted values of u, v, p are the same at the start. Hence the terms with partial differentials in equations 3(a) to (c) are near to 0 in the low epoch range, leading to a low MRES loss. Eventually as the network learns the actual values of u, v, p, the MRES loss become more significant and that is when the physics-based MRES loss becomes a factor in the training of the PINN. This means that a PINN network is not effective with a relatively low epoch for training.

Afterwards, combinations of *Adam* and *L-BFGS-B* are adopted for hyperparameter tuning. Table 1 outlines the MSE and MRES loss of all the combinations. In 3 out of the 4 combinations, the test MSE improved over the 100% *L-BFGS-B*. All the combinations even had a relatively low MRES loss. Since the "25% *Adam* + 75% *L_BFGS-B*" optimizer combination had the best test MSE performance overall, it is adopted as the optimizer for the PINN.

Table 1: MSE and MRES Loss for different optimizer combinations

Optimizer Combination	Test MSE Loss	Test MRES Loss
50% *Adam* + 50% *L-BFGS-B*	1.206e-03	3.915e-03
50% *L-BFGS-B* + 50% *Adam*	2.008e-03	5.578e-03
25% *Adam* + 75% *L-BFGS-B*	1.035e-03	5.084e-03
75% *Adam* + 25% *L-BFGS-B*	1.268e-03	4.181e-03

Effect of Network Architecture: The final hyperparameter of concern is in reference to the complexity of the network. The present study trained models with shallow networks like [3, 5, 5, 3] to deep networks like [3, 30, 30, 30, 40, 40, 40, 30, 30, 30, 3] and wide networks like [3, 100, 100, 100, 3]. Table 2 list the MSE and MRES losses for each of the network architectures tested. Other than the shallow network of [3, 5, 5, 3], all networks achieved impressive low MSE losses in the low e-03 to low e-04 range. However, most of these networks failed to reach a low MRES value as the original network, with MRES in the order of e-02. The [3, 10, 25, 40, 25, 10, 3] network had the lowest relative MRES loss of 1.207e-02. Given its excellent MSE performance of 2.961e-04, it is chosen as the network for the PINN.

Table 2: MSE and MRES Loss for different PINN Architectures

PINN Architecture	Test MSE Loss	Test MRES Loss
[3, 5, 5, 3]	6.681e-02	1.433e-02
[3, 10, 25, 40, 25, 10, 3]	2.961e-04	1.207e-02
[3, 10, 20, 30, 40, 30, 20, 10, 3]	3.067e-04	2.884e-02
[3, 30, 30, 30, 40, 40, 40, 30, 30, 30, 3]	2.577e-03	8.223e-02
[3, 10, 15, 20, 25, 25, 20, 15, 10, 3]	2.988e-04	1.725e-02
[3, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 3]	1.336e-03	3.771e-02
[3, 100, 100, 100, 3]	3.159e-04	1.660e-02

With this, the very wide and deep network is adopted as the network architecture for the PINN model. In Fig 10, the predictions of velocity magnitude & pressure from the PINN model with the tuned hyperparameters are compared with the numerical representation. There is no visual difference between the predicted and CFD contours for velocity magnitude and pressure. This impressive match is substantiated by the low test MSE loss of 2.961e-04.

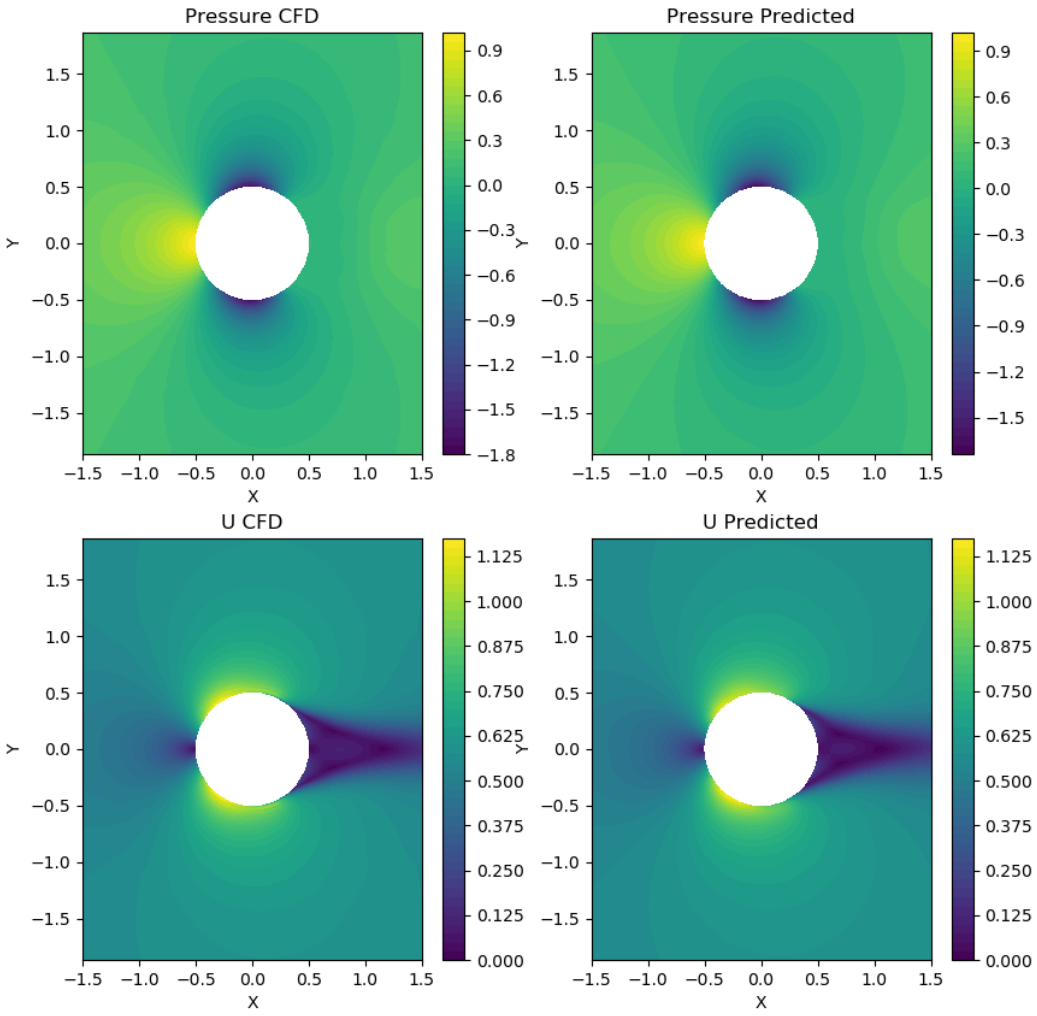


Fig 10: Numerical & Predicted plots of velocity magnitude and p of the Optimized Model of PINN

3.2. PINN vs NN

Despite the relatively low loss of the PINN, the motivation to use apply it in surrogate models can only be argued if it performs better than the traditional NN. The λ factor from Equation (4d) is set to 0 and the same optimized model hyperparameters and training procedures are used to train a NN. The MSE loss after training is 4.882e-04. This loss is not as low as compared to the 2.196e-04 from the PINN.

The predictions of the NN velocity magnitude and p are illustrated in Fig 11. Just like Fig 10, there is no discernible visual differences between the predicted and CFD contours of velocity magnitude and p. At the MSE loss order of e-04, there is very little difference between the predictions and CFD, hence both the NN and PINN are able to predict the flow very accurately. However, the PINN has a slight edge in its predictive ability with its lower MSE loss. The advantage of PINN over NN becomes much more apparent when CFD data is scarce. Instead of using $N_{train} = 1000$, the same training for both NN and PINN $N_{train} = 50$, with $N_{Col} = 0$.

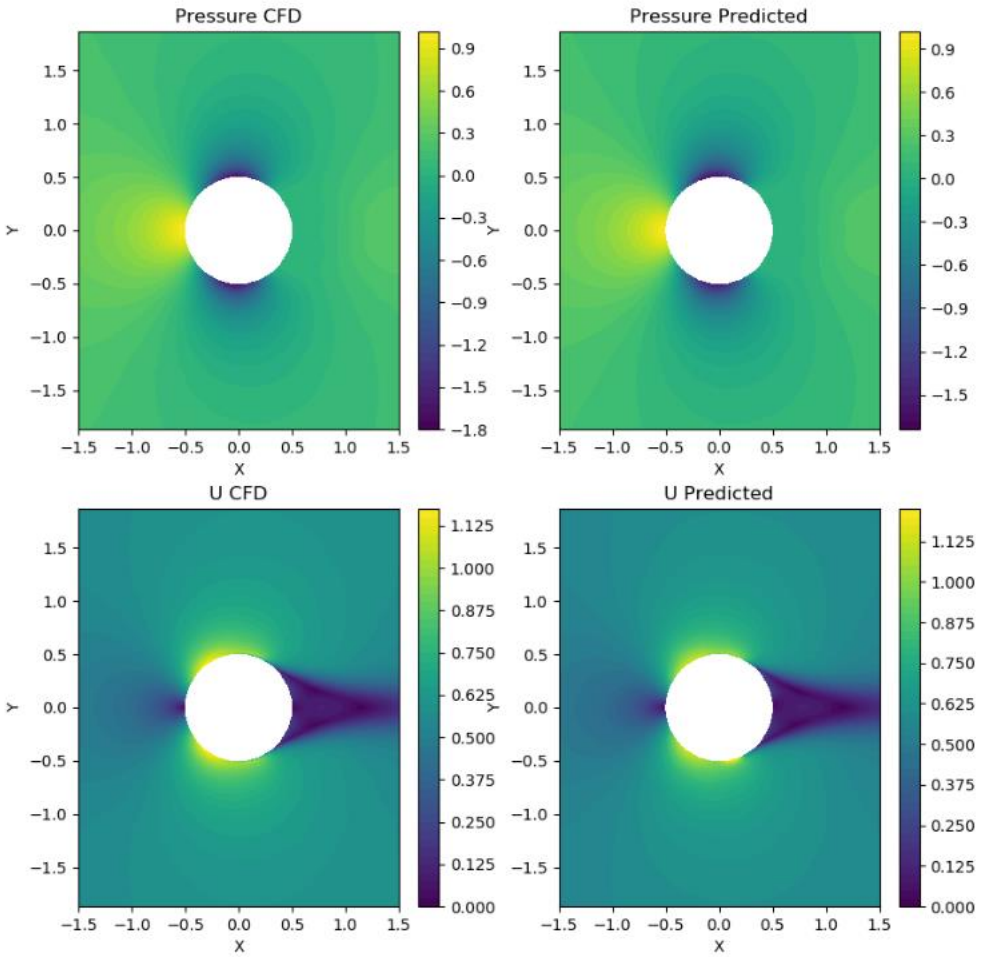


Fig 11: Numerical & Predicted plots of velocity magnitude and p of the Optimized Model of NN

The MSE loss for the PINN is 1.879e-02 while the MSE loss for the NN is higher at 3.873e-02. In this case the absolute difference in error of PINN and NN is significantly higher than the case with $N_{train} = 1000$ as the order of magnitude is higher at e-02. This difference becomes obvious when visually comparing the difference in pressure contours for PINN and NN. In Fig 12, the viscous Magnus CFD pressure contour is compared with PINN and NN. A major difference comes from the top region of the cylinder. In the CFD contour, there is a small region of negative gauge pressure, which only exists near the cylinder wall. In the PINN contour, the same region of negative gauge pressure exists near the wall too, however it incorrectly predicts a region of relatively high pressure diagonally up the domain (depicted by the diagonal yellow region). The prediction in NN, appears to show a drastically large region of negative gauge pressure that extends diagonally up, which appears significantly different from the CFD contour. Other than the MSE loss, it is visually clear that PINN prediction is superior to NN prediction with scarce CFD data points.

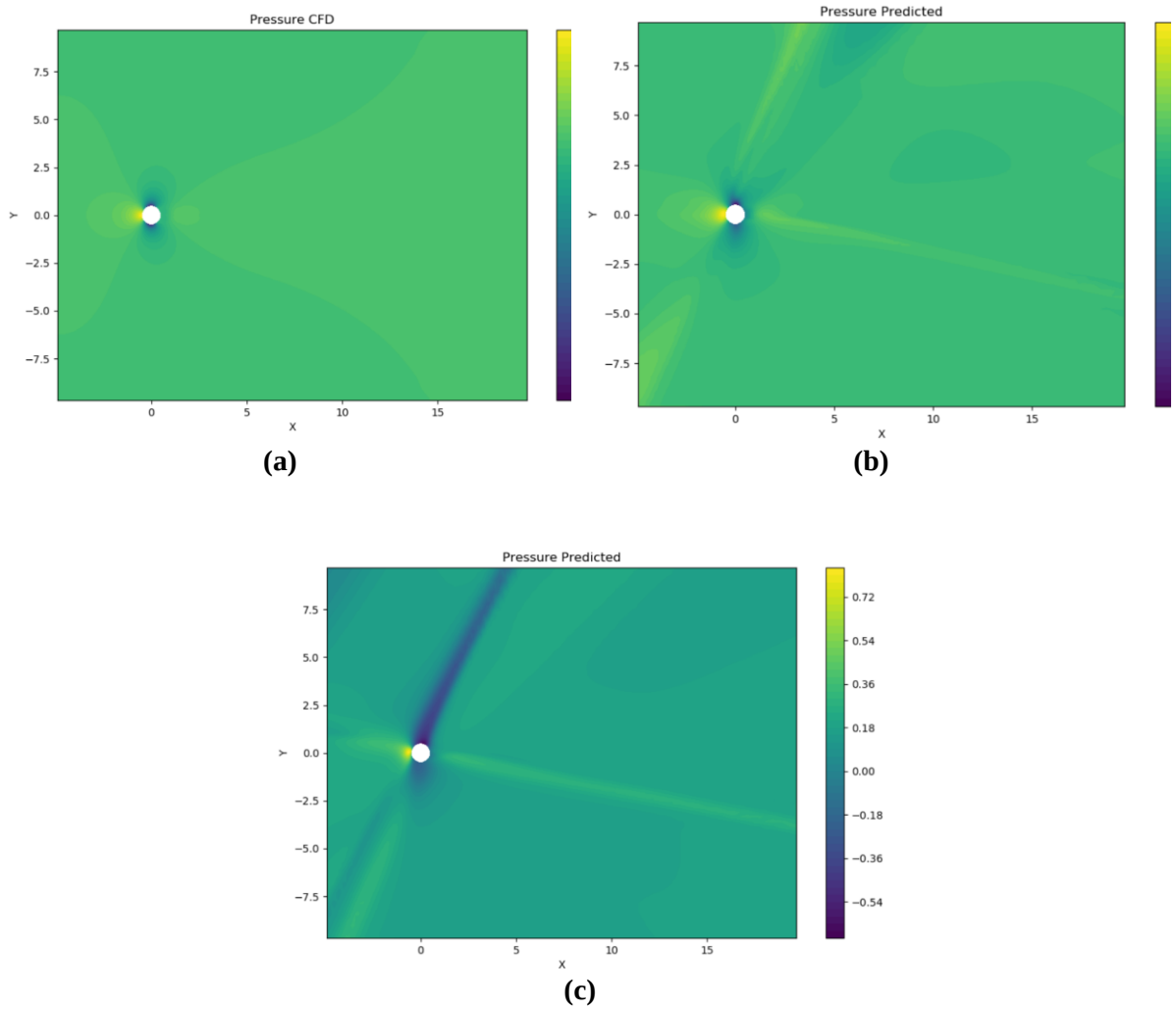


Fig 12: Pressure contours of (a) CFD (b) PINN (c) NN with $N_{train} = 50$

3.3. Effect of Collocation and Training Data Size

The parameter N_{train} is varied from 100 to 2000 in intervals of 100 to examine how it affects the MSE loss with N_{col} kept at 200. The results are shown in Fig 13. The test MSE loss appears to fluctuate and decrease with increasing N_{train}. As there are only 20 discrete points in the plot, a smooth line is shown, estimating the smooth change in test MSE loss. The smooth line shows that the test MSE loss improves by one order of magnitude, from e-03 to e-04. The plot in Fig 13 shows that when N_{train}>1000, the test MSE loss stays within a rough range of 2e-04 to 4e-04. This shows that the test MSE does improve in the range 100<N_{train}<1000, however no significant improvement is seen afterwards.

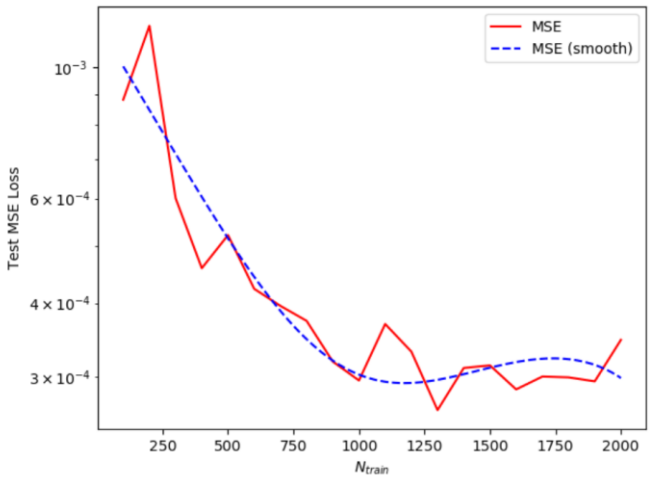


Fig 13: Test MSE Loss with increasing N_{train}

The fluctuations also show that simply increasing the N_{train} just slightly, from 1000 to 1100 for instance, does not necessarily improve performance. Nonetheless, the global trend does show that the test MSE will improve with a large enough N_{train}. That threshold in the present study is established at roughly 1000. The best performance is delivered at N_{train}=1300 with a test MSE loss of 2.639e-04.

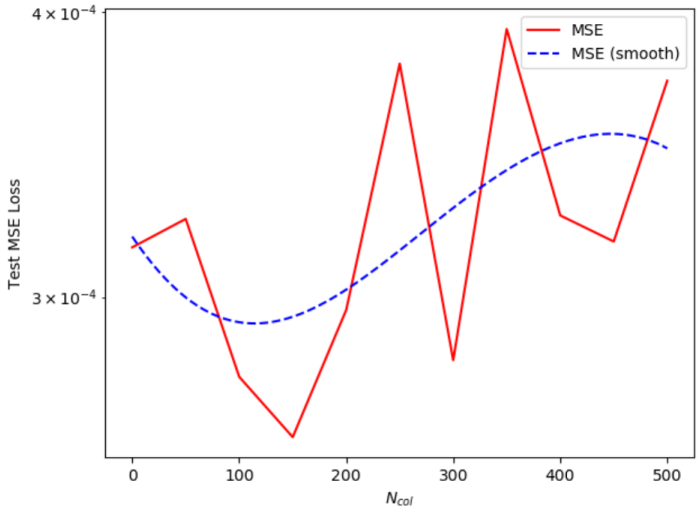


Fig 14: Test MSE Loss with increasing N_{col}

Similarly, N_{col} is varied from 0 to 500 in intervals of 50 to investigate its effect on the MSE loss with N_{train} kept at 1000. The plot is illustrated in in Fig 14. Unlike the case with N_{train}, there is no clear global trend in the data. The test MSE loss fluctuates rapidly between 2e-04 and 4e-04. This shows that the N_{train} is much more dominant in improving the overall performance of the PINN. The best performance in Fig 14 is achieved at N_{col}=150 with a test MSE loss of 2.768e-04.

3.4. Magnus PINN

The goal of the Magnus PINN is to train a model that can predict the velocity magnitude and pressure flow fields for any α for a given Re_D. The same hyperparameters in the optimized model from the hyperparameter tuning process are used to train the Magnus PINN. As the mother dataset is significantly larger (3,654,200) that the data set of the PINN for stationary cylinder (332,000), any optimized N_{train}

and N_{col} values are not yet adopted. They are kept initially at 1000 and 200 respectively. As α is a new additional input, the PINN architecture is revised to [4, 10, 25, 40, 25, 10, 3]. After training at 20,000 epochs, the test MSE loss amounted to 7.996e-04. The subsequent results shown are for **Re_D = 150,000**.

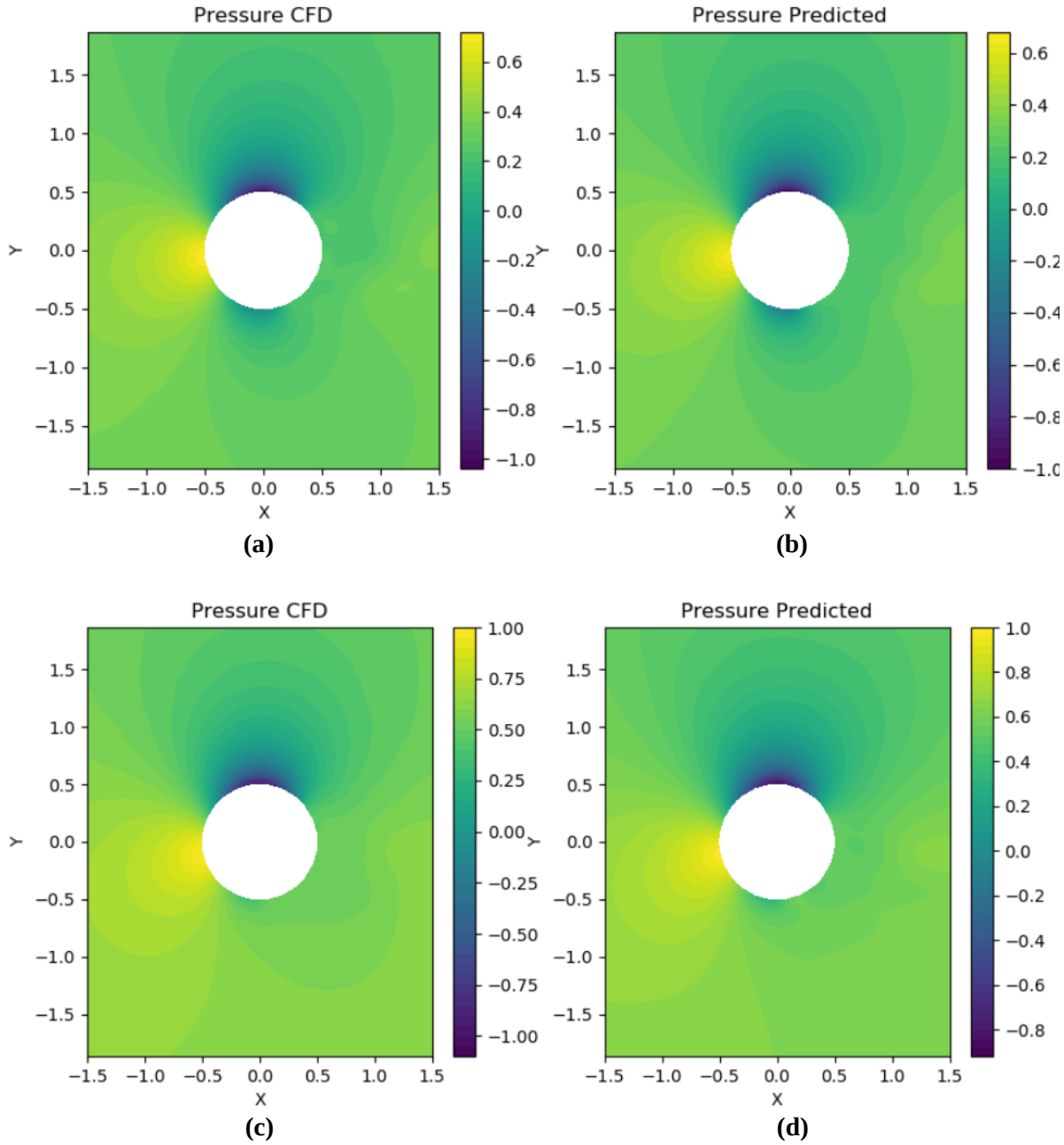
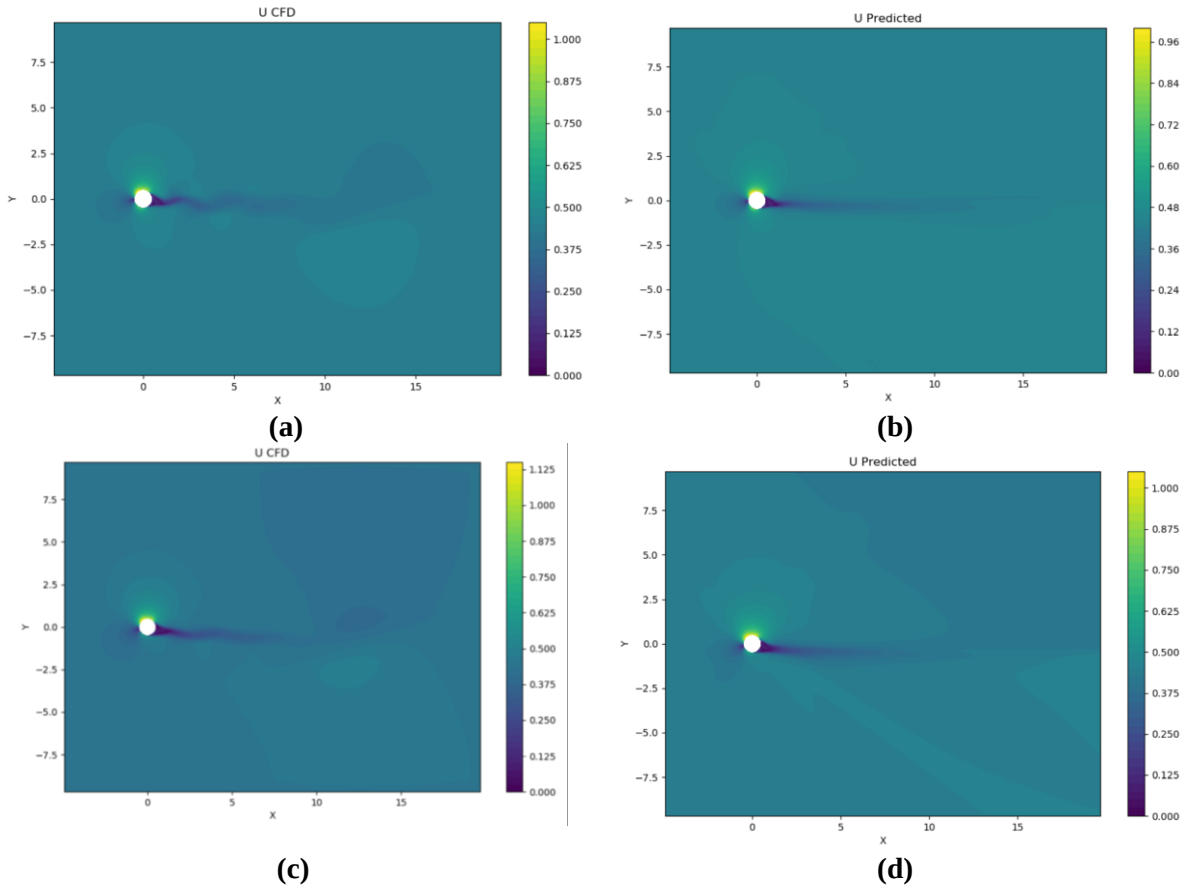


Fig 15: Pressure contours (close-up) of (a) CFD at α=0.5 (b) PINN at α=0.5 (c) CFD at α=1 (d) PINN at α=1

Fig 15 illustrates the comparison of pressure contours between the CFD and Magnus PINN at α=0.5 and α=1. There does not appear to be any discernible visual differences in the pressure contours for CFD and Magnus PINN at both α.

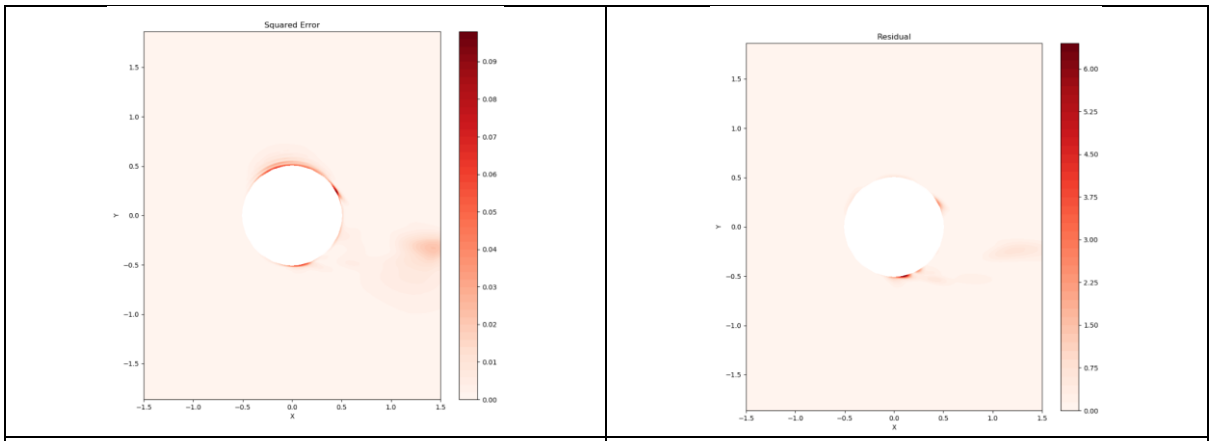
Fig 16 shows the comparison of the velocity magnitude contours between CFD and the Magnus PINN at α=0.5 and α=1. Near the wall of the cylinder, the predictions at both α seem to be visually accurate. The CFD results for both α show instances of vortex shedding, where there is a region of low momentum fluid with a characteristic oscillating flow in the wake of the cylinder. However, the PINN predictions are unable to pick up this feature in the flow field.



(a) **(b)**

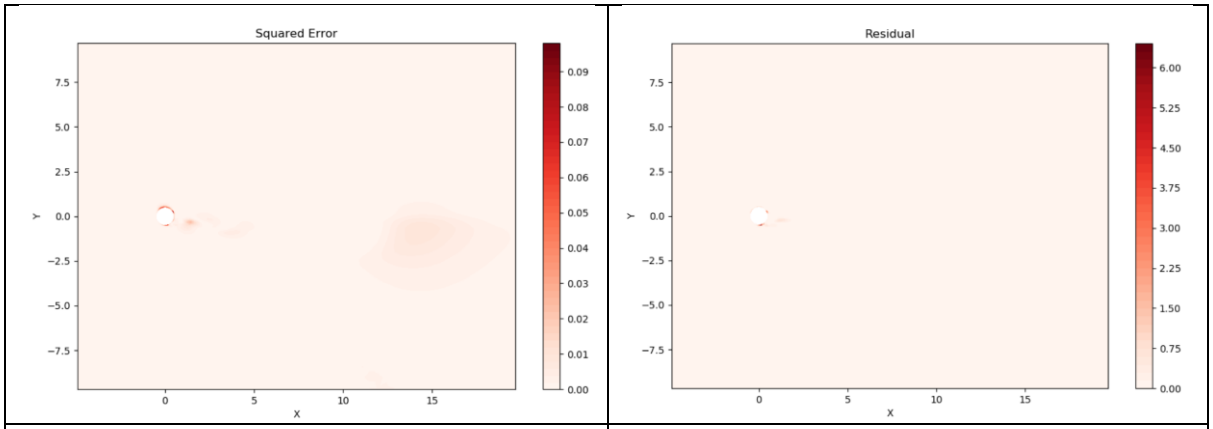
(c) **(d)**

Fig 16: Velocity magnitude contours of (a) CFD at α=0.5 (b) PINN at α=0.5 (c) CFD at α=1 (d) PINN at α=1 with N_{train}= 1000 and N_{col} = 200



(a) **(b)**

Fig 17: Spatial Distribution of (a) MSE (closeup) and (b) MRES (closeup) of Magnus PINN (continued)



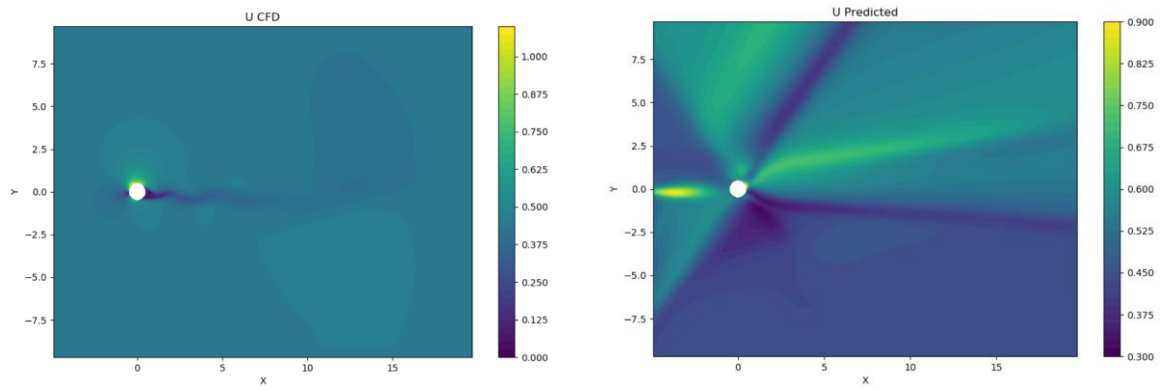
(c) **(d)**

Fig 17: Spatial Distribution of (a) MSE (full view) and (b) MRES (full view) of Magnus PINN

Fig 17 illustrates the spatial distribution of the MRES and MSE errors of the Magnus PINN. From Fig 17(a), the MSE errors mainly stem from the region immediately attached to the cylinder wall. The MRES error in Fig 17(b) is also localised to small regions adjacent to the cylinder walls. In the whole domain, the MRES error is non-existent throughout any region away from the cylinder in Fig 17(d). From Fig 17(c), the wake region behind the cylinder has a small region of significant MSE error. This can confirm the visual differences in the velocity magnitude contours in the cylinder wake in Fig 16.

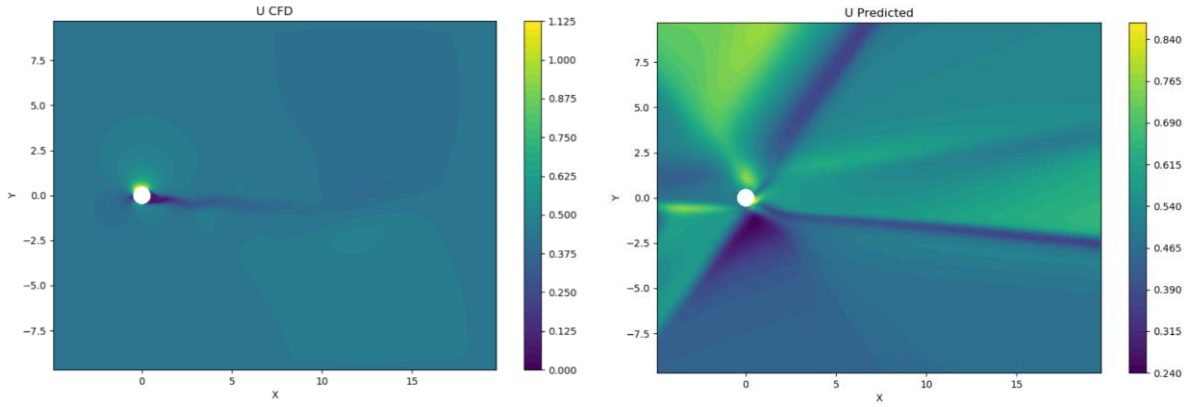
To improve this wake region prediction, N_{train} and N_{col} are increased to 5000 and 1000 respectively. However, the test MSE loss did not change much with a value of 7.813e-04. The subsequent step to improve performance is to increase the number of epochs from 20,000 to 35,000. The resulting training gave a test MSE loss of 8.330e-04, which is slightly worse in performance than 20,000 epochs. Increasing epochs and training data size did not help to improve the Magnus PINN performance significantly. One potential solution is to increase the concentration of cells used in training under N_{train} with cells in the wake region of the cylinder. This approach would therefore change the random selection method being used so far.

The next step in further probing the Magnus PINN is to test its ability to interpolate and extrapolate at different α. This is the case whereby the trained model is made to predict the flow field at α values that it did not train with. α =1.2 is used for extrapolation and α=0.65 is used for interpolation. The viscous Magnus flow CFD simulation for the aforementioned α are done in the same manner as the previously used simulations. N_{train} and N_{Col} are kept at 5000 and 1000. The CFD data at Re=200,000 is used in this case.



(a) **(b)**

Fig 18: Velocity magnitude contours of (a) CFD at α=0.65 (b) PINN at α=0.65 (continued)



(c) **(d)**

Fig 18: Velocity magnitude contours of (c) CFD at α=1.2 (d) PINN at α=1.2

Neural network models in general applications are not well known for their ability to interpolate and extrapolate. Similarly, the attempts to do as such in the present study has not shown good performance. Fig 18 illustrates the comparison of the velocity magnitude contour plots between CFD and Magnus PINN at α=0.65 and α=1.2. The Magnus PINN results have no reasonable resemblance to the CFD contour plots at both α. The test MSE losses are 7.783e-01 and 8.315e-01 for α=0.65 and α=1.2 respectively. The trained model is highly incapable of interpolating and extrapolating, even though the α values are only minimally different from the trained α values from 0 to 1 in intervals of 0.1. This is most likely an issue of overfitting. Overfitting is the scenario where the PINN is less generalised and strictly trained to only predict the data which is used as training data. Therefore, the PINN developed in the present study may be excellent at predicting the flow field for the specific α values, but very poor at any other values. To reduce the effect of overfitting, regularisation techniques such as dropout and L2-regularisation can be adopted to reduce the PINN's dependency on the data. This can aid the PINN model in extrapolating and interpolating the flow field.

4. CONCLUSION

With the development of PINN in its infancy stage, there is a whole field left to be discovered. The capability of PINN is certainly yet to be unlocked. The usefulness of PINNs to behave as surrogate models will be of great use, considering the restrictions of CFD data and the speed of numerical simulations. In this study, PINN is introduced in the context of the viscous Magnus flow, where the cylinder is either stationary or rotating. An extensive hyperparameter tuning process optimized 4 key hyperparameters to improve the performance of the model for the case of Re=100,000 and α=0. Tracking the loss with epoch showed that the performance of PINN is poor in the low epoch range. The benefit of using PINN over NN as a surrogate model is shown in the present study, with the PINN giving better performance than NN, especially with scarce training data. Increasing the training data size does improves the performance of the PINN, however this effect does not carry on after a certain data size threshold ($N_{train} = 1000$ in the present study). Varying the collocation data size did not significantly alter the performance of the PINN. Finally, a Magnus PINN that predicts the flow field for a range of α for a given Re_D is developed. Efforts to use the Magnus PINN to extrapolate and interpolate at a different α did not work well. Further studies are needed to explore the effectiveness of regularisation in PINNs to reduce the effect of overfitting. The present study may have answered some questions, but it has been made clear that further studies are exceedingly necessary to better understand the workings of PINNs at a deeper level.

REFERENCES

[1] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019

[2] P. Hoffmann, (2015), A Hitchhiker's Guide to Automatic Differentiation, *Numer Algor (2016),***72**:775–811

[3] Magnus, G., "Ueber die Abweichung der Geschosse, und: Ueber eine auffallende Erscheinung bei rotirenden Körpern," Annalen der physik, Vol. 164, No. 1, 1853, pp. 1–29.

[4] J. Seifert, "A review of the Magnus effect in aeronautics," *Progress in Aerospace Sciences*, vol. 55, pp. 17–45, 2012.

[5] Swanson, W., "The Magnus effect: A summary of investigations to date," Journal of Basic Engineering, Vol. 83, No. 3, 1961, pp. 461–470.

[6] S. Mittal and B. Kumar, "Flow past a rotating cylinder," *Journal of Fluid Mechanics*, vol. 476, pp. 303–334, Oct. 2003.

[7] J. C. Padrino and D. D. Joseph, "Numerical study of the steady-state uniform flow past a rotating cylinder," *Journal of Fluid Mechanics*, vol. 557, p. 191, 2006.

[8] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, "A tensorial approach to computational continuum mechanics using object-oriented techniques," *Computers in Physics*, vol. 12, no. 6, p. 620, 1998.

[9] D. K. Walters and D. Cokljat, "A Three-Equation Eddy-Viscosity Model for Reynolds-Averaged NavierStokes Simulations of Transitional Flow," Journal of Fluids Engineering, vol. 130, no. 12, p. 121401, 2008

[10] D. Kingma and J. Ba, (2015), ADAM: A Method for Stochastic Optimization, in *Proceedings of the 3rd International Conference on Learning Representations* (ICLR 2015), May 7-9, San Diego, USA.

[11] X. Glorot, Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, Mar. 2010.

[12] M. Abadi et al., (2016), *Tensorflow:* A System for Large-Scale Machine Learning, in Proceedings of *12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16)*, Nov 02-04, pp. 265-283, Savannah, GA, USA; ISBN 978-1-931971-33-1

[13] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, Jan. 1997.

Appendix A

Table 1: Mesh details for the grid convergence

Mesh	N_{Radial}	N_{Circum}	$N_{Vertical}$	N_{Left}	N_{Right}	No. of Cells (N)
1	5	5	30	30	60	4,600
2	10	10	30	30	60	6,400
3	15	15	30	30	60	8,600
4	20	20	30	30	60	11,200
5	25	25	30	30	60	14,200
6	30	30	30	30	60	17,600
7	35	35	30	30	60	21,400
8	40	40	30	30	60	25,600
9	40	40	40	40	60	30,200
10	40	40	40	40	90	34,400
11	40	40	50	50	90	40,000
12	40	40	50	50	120	44,800
13	40	40	60	60	120	58,400
14	40	40	60	60	180	70,400