energies

MDPI

Article

Physics-Informed Neural Networks for Low Reynolds Number Flows over Cylinder

Elijah Hao Wei Ang, Guangjian Wang and Bing Feng Ng *

School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore; elij0001@e.ntu.edu.sg (E.H.W.A.); wanggj@ntu.edu.sg (G.W.)
* Correspondence: bingfeng@ntu.edu.sg

Abstract: Physics-informed neural network (PINN) architectures are recent developments that can act as surrogate models for fluid dynamics in order to reduce computational costs. PINNs make use of deep neural networks, where the Navier-Stokes equation and freestream boundary conditions are used as losses of the neural network; hence, no simulation or experimental data in the training of the PINN is required. Here, the formulation of PINN for fluid dynamics is demonstrated and critical factors influencing the PINN design are discussed through a low Reynolds number flow over a cylinder. The PINN architecture showed the greatest improvement to the accuracy of results from the increase in the number of layers, followed by the increase in the number of points in the point cloud. Increasing the number of nodes per hidden layer brings about the smallest improvement in performance. In general, PINN is much more efficient than computational fluid dynamics (CFD) in terms of memory resource usage, with PINN requiring 5–10 times less memory. The tradeoff for this advantage is that it requires longer computational time, with PINN requiring approximately 3 times more than that of CFD. In essence, this paper demonstrates the direct formulation of PINN without the need for data, alongside hyperparameter design and comparison of computational requirements.

Keywords: physics-informed neural network; low Reynolds number; fluid dynamics; surrogate modelling; Navier-Stokes equation; machine learning

Citation: Ang, E.H.W.; Wang, G.; Ng, B.F. Physics-Informed Neural Networks for Low Reynolds Number Flows over Cylinder. *Energies* **2023**, *16*, 4558. https://doi.org/10.3390/en16124558

Academic Editors: Shine Win Naung, Mohammad Rahmati and Mahdi Erfanian Nakhchi

Received: 4 May 2023
Revised: 30 May 2023
Accepted: 5 June 2023
Published: 7 June 2023

1. Introduction

Fluid flows are ubiquitous in many processes, and accurate simulation of fluid flows is crucial in many applications, such as civil, aerospace, and biomedical engineering. The investigations of low Reynolds number flows remain critical, where fluid simulation is crucial in different applications, such as the development of micro air vehicles with flapping wings [1], as well as investigations into microfluidics [2]. Conventionally, the motion of viscous fluids can be described by a set of non-linear partial differential equations (PDE) called the Navier-Stokes equation, which are solved numerically using computational fluid dynamics (CFD) through the finite volume method [3]. CFD simulations are often cumbersome processes, especially for flows with complicated geometries. In addition, the pre-processing step of CFD, which requires the generation of computational mesh, is an arduous process especially for complicated domains with moving boundaries, and poor generation of mesh could lead to inaccurate results. Hence, grid independence study becomes crucial to ensure that the CFD solution is converged, but itself is a time-consuming process. In spite of the recent rise in computational power, fluid simulations using CFD continues to be computationally demanding and is especially so for design processes where multiple iterations have to be performed, for instance, in the optimization of aircraft geometries. This has prompted researchers to look for cost-effective alternatives to overcome the computation burden surrounding conventional CFD methods.

One of the most common methods to reduce the computational cost for aerodynamics simulations is to use a surrogate model. Traditional methods such as proper orthogonal

decomposition (POD) and kriging interpolation were used to perform dimensional reduction. Such methods often rely on CFD or experimental data for the formulation of these surrogate models and are limited to specific applications [4]. Lately, due to advancements in machine learning and artificial intelligence techniques, artificial neural networks have shown to be suitable candidates to act as surrogate models for fluid dynamics. Existing surrogate models using artificial neural networks work by estimating a relationship between the input features and the desired outputs of the problem, similar to the multiple regression method. Thuerey et al. [5] used the U-Net algorithm as a nonlinear regression tool to solve for flow fields around an airfoil, whereas Zhang, Sung, and Mavris [6] developed a convolutional neural network (CNN) framework to predict the lift coefficient of an airfoil based on pixel images of the airfoil geometry. Separately, Belbute-Peres et al. [7] combined CFD solvers with graph neural networks to predict the flow around airfoils and Han et al. [8] trained a convolutional long short-term memory (ConvLSTM) network to capture the spatial-temporal features of unsteady flow, using data from high-dimensional unsteady CFD results. By exploiting the neural network prediction speed, these methods are effective in aerodynamic predictions. However, the limitation is that the training of the neural networks requires huge datasets, often generated using CFD or experiments, which remains computationally expensive. Training the network using CFD or experimental data also limits the application of the model to very specific problems, and requires a whole new set of data to train a model for other problems. Additionally, these methods are considered as "Black Box" methods, which do not take into consideration the physical definitions of the model [9]. By compromising on physical definitions, the model could produce non-physical results, which have the tendency to affect the accuracy of the model.

In recent years, a novel concept called physics-informed neural networks (PINN) was introduced by Raissi et al. [10]. PINN works by solving the governing equations through the reduction of residuals and errors in the neural network training steps [11]. This new neural network architecture aims to overcome the shortcomings of surrogate modeling methods of the past which lack physical definitions. Raissi et al. [10] demonstrated the use of PINN to solve simple partial differential equations, namely the Schrodinger equation, Allen-Cahn equation, and Burgers' equation. Raissi et al. [12] then extended the concept to compute the lift and drag forces on a bluff body undergoing vortex-induced vibrations. Developments were made for efficient solutions of fluid dynamic problems using PINN, which, however, are limited to very simple cases. Sun et al. [13] made use of PINN to solve the Navier-Stokes equations directly for stenotic flow in blood vessels without the use of any experimental or simulation data. Mao et al. [14] used PINNs to approximate the Euler equations to model high-speed aerodynamics involving shockwaves. In recent years, Cai et al. [15] used PINNs to simulate fluid flows for 3D unsteady cases, turbulent flows, as well as biomedical flows, which are augmented with data obtained from CFD and experimental measurements. Arzani et al. [16] augmented PINN with sparse measurement data to model near-wall blood flow applied to stenosis and aneurysm. Eivazi et al. [17] resolved turbulent boundary layer flow by incorporating the Reynolds-averaged Navier-Stokes (RANS) equation to the PINN with large eddy simulation data for the boundaries of the computation domain. Jin et al. [18] used the velocity–vorticity formulation of the Navier-Stokes equation in PINN to solve for various fluid flow problems while augmenting the network with experimental data on the computation boundaries. It was observed that the velocity–vorticity formulation was less accurate than the primitive variables (velocity–pressure) formulation, which is caused by the definition of experimental data in primitive variables. There have also been attempts to model melt pool dynamics from metal additive manufacturing using PINN, where a simple case of solidification of metal is performed without training data, whereas more complex simulations require the use of data obtained from high-fidelity simulations [19].

PINN had also been used for the prediction of fluid problems which are not solved using the Navier-Stokes equations. For instance, Almajid and Abu-Al-Saud [20] developed a PINN to model the Buckley-Leverett problem for two-phase flows through a porous medium. Their results showed that the PINN performed better when data is being aug-

mented into the framework, and results without data augmentation were sub-optimal. Bararnia and Esmaeilpour [21] modeled flow problems involving viscous and thermal boundary layers by solving the Blasius-Polhausen, Falkner-Skan, and free convection equations. The PINN prediction of the solution to the ordinary differential equations (ODE) agrees with numerical solutions. Sufficient neurons per layer are required to predict solutions with acceptable accuracy.

Currently, PINNs have only been used to study simple cases, and more complex flow problems require augmentation with data obtained from external sources [22–24]. Moreover, the effects of the various hyperparameters in the neural network on the accuracy of the results have not been reported in detail. Limited studies have also been conducted on the computational efficiency of PINN as compared to other methods.

As such, the objective of this study is to demonstrate the use of PINN to solve for low Reynolds number flows, as well as to investigate the critical factors that influence the design of the PINN architecture. These include varying the number of neural network layers, nodes, and size of the point cloud. Here, our PINN will incorporate the Navier-Stokes equation into the training process to predict flow fields across a cylinder without any use of simulation or experimental data. Freestream conditions are prescribed for the boundaries of the computation domain and will be used for the training of the PINN. In Section 2, the methodology involving PINN theory, constraints, setup, and validation will be discussed. Following this, in Section 3, factors influencing PINN in flow field predictions will be presented. Discussions on maximum error and resource utilization are subsequently discussed in Section 4, followed by Conclusions in Section 5.

2. Methodology

2.1. Physics-Informed Neural Network

The PINN is built from fully connected layers where the weights and biases are adjusted according to physical governing equations and predefined boundary conditions. For a fully connected neural network, every node in each layer is linked to every node in the subsequent layer. The first layer is called the input layer, where the number of nodes is determined by the number of selected features that are to be fed into the network. For fluid flows, the features are the coordinates and time. The last layer in the neural network is the output layer, which gives the predicted values for the given input features. The number of nodes in the output layer is the same as the number of predictions needed. The layers between the input layer and output layer are known as hidden layers, and each layer can have any number of nodes.

Figure 1 shows the schematic for the computation of the output value of a single node. Every connection holds a weight, and each node is assigned a bias, which is utilized in calculating the output value of each node. Every node in the PINN holds an activation function, which determines the nodal output based on its input values. The output value of each node is computed as:

$$y = \varphi(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b) \tag{1}$$

$$y = \varphi(\boldsymbol{w} \cdot \boldsymbol{x} + b) \tag{2}$$

where y denotes the nodal output, φ represents the activation function, $\boldsymbol{x} \in \mathbb{R}^n$ gives the inputs into the node, $\boldsymbol{w} \in \mathbb{R}^n$ represents the vector of connection weights, and b represents the nodal bias. Some of the activation functions that are commonly used in deep neural networks are the sigmoid function, rectified linear unit (ReLU), and the hyperbolic tangent function [25]. The activation functions add nonlinearity to the network, which is critical as many problems in physics exhibit nonlinear characteristics. The activation functions also allow for the automatic differentiation algorithm to be used in order to compute the spatial or temporal derivatives found in the governing equations [26]. Once the output of the node is computed, it will be used as the input into the node of the next layer. This process repeats all the way to the output node.

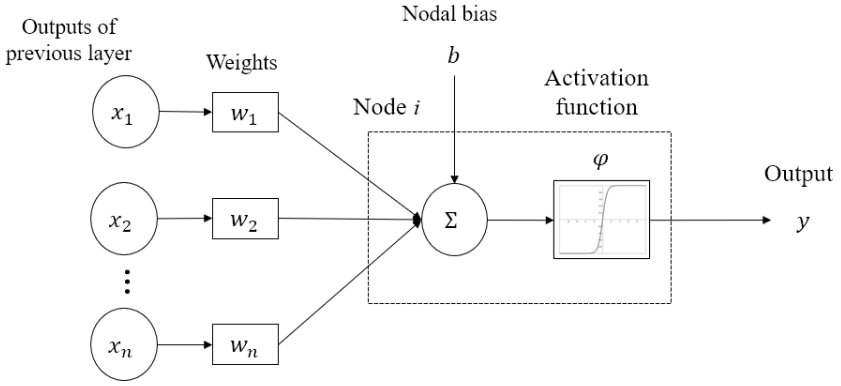

Figure 1. Schematic for computation of a single node.

During the training phase, the neural network loss is computed based on the output values generated by the network. This requires a loss function to be formulated specifically for the problem at hand. The neural network initially learns by updating the weights and biases through the backpropagation algorithm in order to minimize the loss until it reaches a specified tolerance level [27]. Subsequently, the weights and nodal biases are corrected by a factor that scales with the gradient of the loss function. A complete iteration through the entire set of training points is referred to as an epoch, and the backpropagation process is repeated for multiple epochs until the loss falls below the tolerance level.

2.2. Physics Constraints for Fluid Flows

The physics of fluid flows is governed by the Navier-Stokes equations. In the case of incompressible flows, the continuity and momentum equations can be expressed as follows:

$$\nabla \cdot \boldsymbol{u} = 0,\ \boldsymbol{x}, t \in \Omega_f \times [0, T] \tag{3}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u},\ \boldsymbol{x}, t \in \Omega_f \times [0, T] \tag{4}$$

where t and $\boldsymbol{x}$ are the temporal and spatial coordinates, respectively, $\boldsymbol{u}$ is the velocity vector, and p is pressure, ρ and ν are the fluid density and kinematic viscosity, respectively, and $\Omega_f \subset \mathbb{R}^3$ denotes the fluid domain [3]. Additionally, the initial and boundary conditions need to be obeyed and are given as:

$$\boldsymbol{u} = I_u(\boldsymbol{x}),\ \boldsymbol{x} \in \Omega_f, t = 0 \tag{5}$$

$$p = I_p(\boldsymbol{x}),\ \boldsymbol{x} \in \Omega_f, t = 0 \tag{6}$$

$$\boldsymbol{u} = B_u(\boldsymbol{x}),\ \boldsymbol{x}, t \in \partial\Omega_f \times [0, T] \tag{7}$$

$$p = B_p(\boldsymbol{x}),\ \boldsymbol{x}, t \in \partial\Omega_f \times [0, T] \tag{8}$$

where $\partial\Omega_f$ represents the boundaries of the computational domain. The governing equations and the initial and boundary conditions have to be completely satisfied within the domain, and any imbalance in the equations indicates errors in the flow fields. Therefore, Equations (3) and (4) must be integrated into the loss function for fluid flow, given by:

$$L = L_I + L_B + L_{phy} \tag{9}$$

$$L_I = \left\| \widetilde{\boldsymbol{u}}(\boldsymbol{x}, 0) - I_u(\boldsymbol{x}) \right\|_{\Omega_f} + \left\| \widetilde{p}(\boldsymbol{x}, 0) - I_p(\boldsymbol{x}) \right\|_{\Omega_f} \tag{10}$$

$$L_B = \left\| \widetilde{\boldsymbol{u}}(\boldsymbol{x},t) - B_u(\boldsymbol{x}) \right\|_{\partial\Omega_f \times [0,T]} + \left\| \widetilde{p}(\boldsymbol{x},t) - B_p(\boldsymbol{x}) \right\|_{\partial\Omega_f \times [0,T]} \tag{11}$$

$$L_{phy} = \left\| \nabla \cdot \widetilde{\boldsymbol{u}} \right\|_{\Omega_f \times [0,T]} + \left\| \frac{\partial \widetilde{\boldsymbol{u}}}{\partial t} + \left(\widetilde{\boldsymbol{u}} \cdot \nabla \right) \widetilde{\boldsymbol{u}} + \frac{1}{\rho} \nabla \widetilde{p} - \nu \nabla^2 \widetilde{\boldsymbol{u}} \right\|_{\Omega_f \times [0,T]} \tag{12}$$

Variables denoted with a tilde represents velocities and pressure predicted by the PINN. L_I represents losses due to initial conditions and L_B represents losses in the boundary condition, and they are given by the magnitude of the difference between the PINN predicted values and the prescribed values. L_{phy} represents the loss in the physics, where the first and second terms are the error for the continuity equation and the momentum equation of the Navier-Stokes equations, respectively. The errors for both equations must be eliminated to satisfy the Navier-Stokes equations. Thus, any nonzero error values can be regarded as loss to the PINN. The spatial derivatives for the Navier-Stokes equations are computed using automatic differentiation [26]. Automatic differentiation is preferred over numerical differentiation in this case as the latter causes truncation errors from the choice of finite difference schemes, whereas automatic differentiation provides the analytical derivative based on the PINN as a continuous function [13]. Since the objective of the PINN is to solve the governing equations by minimizing the errors across the fluid domain, no experimental or simulation data is necessary for the training and solution step. Figure 2 summarizes the PINN framework used to predict flow fields using a point cloud as the input data.

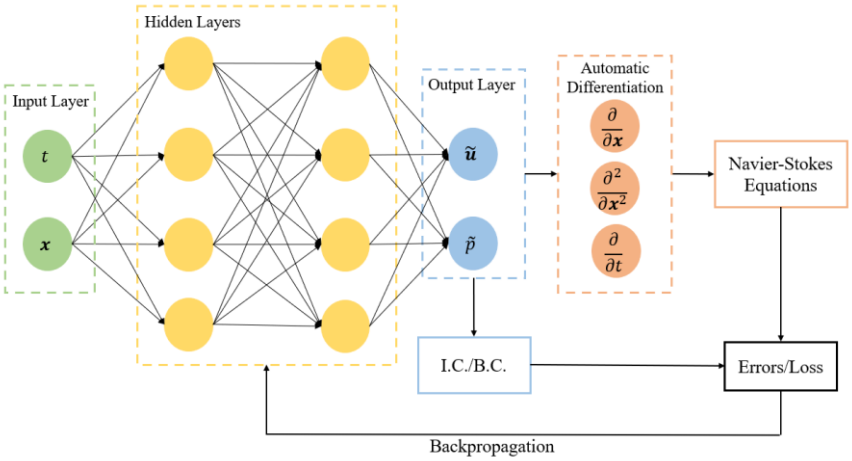

Figure 2. Overall framework for PINN.

2.3. Implementation of PINN

Figure 3 shows a flowchart of the training and implementation of a PINN for fluid dynamics. The implementation of the PINNs is done in PyTorch v 1.5.1 [28], which is an open-source machine learning framework which allows for rapid prototyping and development of various neural network architectures. First, a fully connected neural network is set up using PyTorch. The number of hidden layers, nodes, and activation function are defined in the setting up of the neural network. Secondly, the point cloud for the fluid domain and boundaries are generated using Latin hypercube sampling (LHS). In a 2D steady case, each point in the point cloud of the fluid domain contains x and y coordinates, which acts as the input features for training of the fully connected network. The training process operates in iterations, or epochs, starting with the forward propagation of the input features to compute the predicted output values of velocities and pressure at each point. The output values are used to compute the spatial derivatives needed by the Navier-Stokes equations given in Equations (3) and (4). The derivatives are computed using automatic differentiation, which is a built-in function in PyTorch which takes the

velocities and pressure at a point, and its corresponding spatial coordinates to compute the derivatives. With these derivatives, a custom physics-based loss function based on Equations (9) to (12) is computed. In each epoch, an optimizer aims to reduce the loss function, and hence reduce the residuals and errors from the Navier-Stokes equation to solve the flow problem with the given boundary conditions.

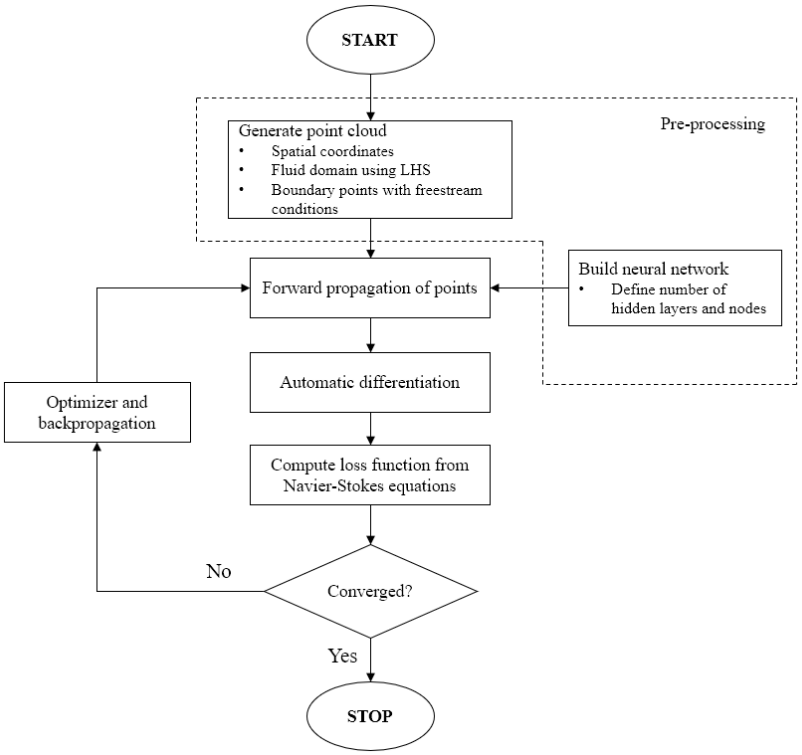

Figure 3. Flowchart for implementation and training of PINN.

2.4. PINN Setup

A flow over cylinder case will be used to demonstrate PINN and investigate the effects of the density of points, number of layers, and nodes on its accuracy. The input data used to train the PINN is represented by a point cloud, as shown in Figure 4, which includes freestream boundaries, cylinder boundaries, and the collocation points within the fluid domain that are used for the computation of errors and imbalance in the Navier-Stokes equations. The coordinates for the point cloud are randomly generated using LHS. Unlike in typical CFD meshes, LHS produces a random but uniform point cloud throughout the fluid domain rather than having greater density of points around the cylinder. Instead of using several neighboring points to compute the derivatives, PINN uses automatic differentiation to enforce the governing equation at each single point. Hence, a sufficiently dense point cloud is required across the entire fluid domain in order to capture the flow fields accurately. 180 points are placed along the cylinder surface and 600 points for the freestream boundary conditions. Since this case is used to study the effect of various critical factors in the PINN, the number of points, layers, and nodes varies according to Table 1, and 3000 epochs are used to reach a root-mean-squared error (RMSE) of below 10^{-3} [29] as there is no observable further reduction in the loss as the number of epochs increases beyond that.

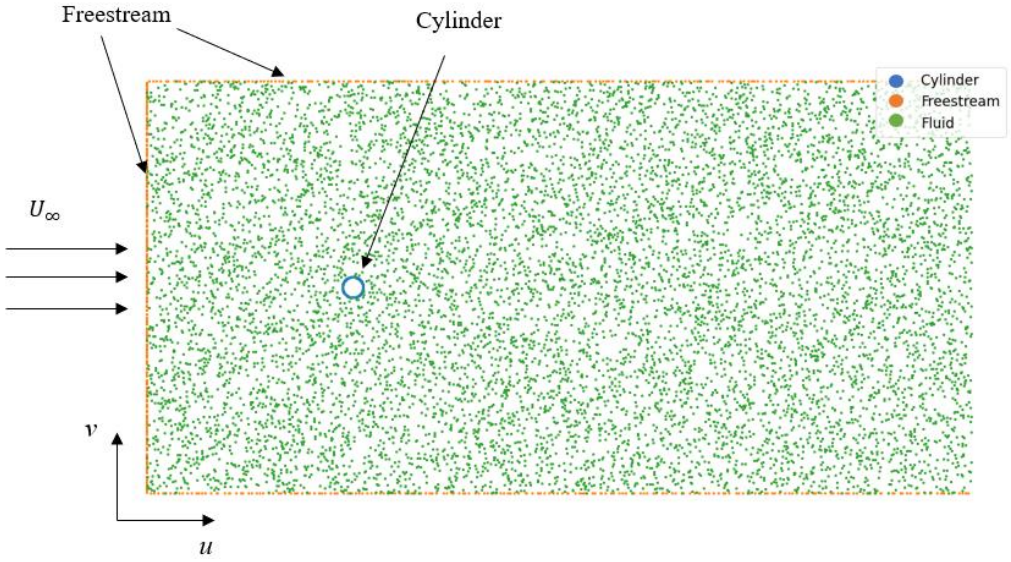

Figure 4. Schematic of point cloud for the cylinder case used to train the PINN. The orange points enforce the freestream boundary condition of 1 m/s, the blue enforce the no-slip boundary condition on the cylinder surface, and the green points are the LHS sampled collocation points where errors due to the governing equations are computed and reduced.

Table 1. Parameters for parametric study on flow over cylinder.

Study	Case 1	Case 2	Case 3	Case 4
Number of Points	2000	5000	10,000	15,000
Number of Nodes	30	40	50	60
Number of Layers	10	15	20	25

Unless otherwise stated, the PINN is developed using the machine learning framework, PyTorch [28], and trained using the Adam optimizer, which is a gradient descend algorithm to find the point of minimal loss, on an RTX 2060 GPU.

2.5. Validation Setup with Computational Fluid Dynamics

In order to validate the results obtained from PINN, CFD simulations are performed for flow over cylinder. The flow over cylinder case will be used as a simple test case to perform parametric study on PINN, and to evaluate PINN's ability to simulate flow over bluff bodies. Steady-state incompressible Navier-Stokes equations are solved using the finite volume method. For 2D incompressible steady-state problems, the governing equations reduces to,

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{13}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{14}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{15}$$

The open-source software OpenFOAM [30] is used to simulate the following cases on an Intel Xeon processor.

The schematics of the CFD computational domain and a close-up view of the mesh for the flow over a cylinder are shown in Figure 5. Here, a computational domain of 40 D by 20 D is defined, where $D = 0.1$ m is the diameter of the cylinder. The dotted arrow extends from the upper surface of the cylinder to the top edge of the computational domain and

indicates the line in which the various flow fields will be plotted for comparison of results between CFD and PINN. The CFD mesh is comprised of hexahedra cells, and the near wall cell thickness are refined to ensure that the boundary layer is well defined. An inlet velocity of 1 m/s with kinematic viscosity $\nu = 2 \times 10^{-2}$ m^2/s is prescribed 10 D upstream from the center of the cylinder and to the bottom and top boundaries of the computation domain. The resulting Reynolds number for this case is 5. A low Reynolds number is selected to ensure that the flow remains steady, and no vortex shedding happens. The right boundary of the domain is prescribed as a pressure outlet, and its placed 30 D from the center of the cylinder. A no-slip boundary condition is applied to the surface of the cylinder.

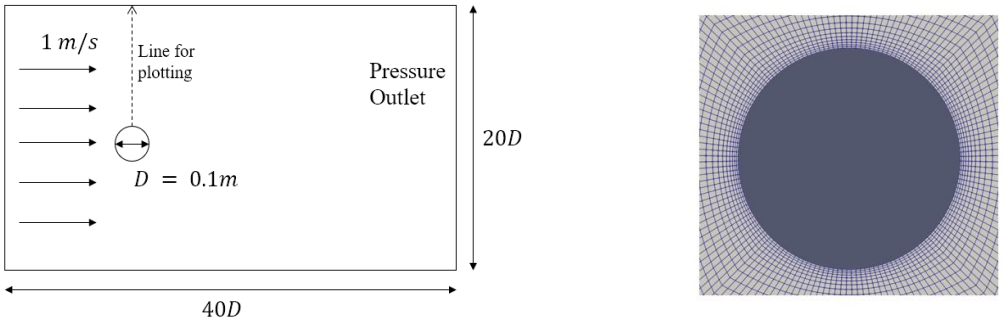

Figure 5. (**a**) Diagram illustrating the computational domain for cylinder; (**b**) CFD mesh near the cylinder wall.

The Navier-Stokes equations are solved numerically using simpleFoam, which uses the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithm to update the velocity and pressure iteratively. Spatial discretization is performed using second order upwind schemes.

Table 2 shows the grid independence study for the flow over cylinder. Based on this study, mesh Cylinder-2 is used as the drag difference between Cylinder-2 and Cylinder-1 is only 0.0045%. Table 3 compares the drag coefficient obtained from the CFD simulations with past experiments and numerical studies. The current result of the drag coefficient agrees well with experimental results from Tritton [31] and DNS simulation from Posdzeich and Grundmann [32] for Reynolds number of 5.

Table 2. Grid independence study for cylinder case.

Meshes	Number of Cells	C_D	Difference
Cylinder-1	20,392	4.5028	-
Cylinder-2	41,772	4.5030	0.0045%

Table 3. Comparison of current CFD results with experiments and DNS.

	Method	C_D
Tritton [31]	Experiment	4.467
Posdziech et al. [32]	DNS	4.1904
Current	CFD	4.503

3. Results

The effect of number of point cloud, hidden layers, and nodes per hidden layer on the accuracy of PINN is first investigated. First, the point cloud has to be dense enough to resolve flow fields near the cylinder. Insufficient points in the point cloud could lead to poorly resolved boundary layer, which leads to unsatisfactory results throughout the entire domain. The number of hidden layers and number of nodes per hidden layer

together affects the expressive power of the PINN. As demonstrated by Cybenko [33] and Hornik [34], if the number of hidden layers is insufficient, PINN will require large amounts of nodes to obtain an accurate result that leads to high computational overheads in the training process. Having an insufficient number of nodes will have to be compensated by large number of hidden layers. It is, therefore, a delicate process to balance the number of hidden layers and number of nodes per hidden layer to achieve desired accuracy of results and minimal computational costs. Table 4 shows the number of points, number of hidden layers, and number of nodes per hidden layer used in this parametric study.

Table 4. Test cases for the effect of number of points in point cloud, number of hidden layers, and number of nodes in hidden layer on the accuracy of PINN.

Case	No. of Points	No. of Layers	No. of Nodes per Layer
Effect of number of points in point cloud	2000, 5000, 10,000, 15,000	20	50
Effects of number of hidden layers	5000	10, 15, 20, 25	50
Effects of number of nodes in hidden layer	5000	20	30, 40, 50, 60

3.1. Effect of Number of Points

The density of the point cloud affects the resolution of the solution. Insufficiently dense point clouds can lead to the failure in properly resolving near wall flow conditions. Point clouds that are too dense can encounter difficulty in neural network training. In order to isolate the effects of number of points, the number of layers in the network and number of nodes per layer were kept as constants. PINN consists of 20 hidden layers, with 50 nodes per hidden layer. The accuracy of PINN was investigated for 2000 points, 5000 points, 10,000 points, and 15,000 points.

The u velocity component, v velocity component, as well as pressure fields for 2000 points in the point cloud from PINN and CFD are shown in Figure 6(a1–c1) and Figure 6(a2–c2), respectively. The differences between PINN and CFD are shown in Figure 6(a3–c3). When using just 2000 points in the fluid domain, it is clear that PINN was unable to achieve comparable results to CFD. PINN was unable to capture the correct leading-edge velocity profiles, as well as the correct pressure field and magnitude. This suggests that 2000 points created a fluid domain is too sparse to accurately capture the correct velocity and pressure fields for this case.

Increasing from 2000 to 5000 points, the PINN starts to capture the velocity and pressure fields with greater accuracy. While the velocity profile is similar between PINN and CFD, the PINN captured lower velocity magnitudes, and smaller area of flow acceleration.

The velocity and pressure fields obtained from both PINN and CFD for 10,000 points on the point cloud are shown in Figure 7. After increasing the number of points from 5000 to 10,000, no improvements could be observed for the velocity components. The u velocity components captured the wake profile, but the velocity magnitudes around the leading-edge, and the accelerated flow region are diminished. Similarly, for the v velocity profile, the trends are well captured, but regions of higher velocity magnitudes are also diminished. However, when looking at the pressure profile, the profile captured by PINN begins to look more symmetrical. Same as before, the pressure magnitudes captured by PINN is lower than the pressure magnitudes captured by CFD.

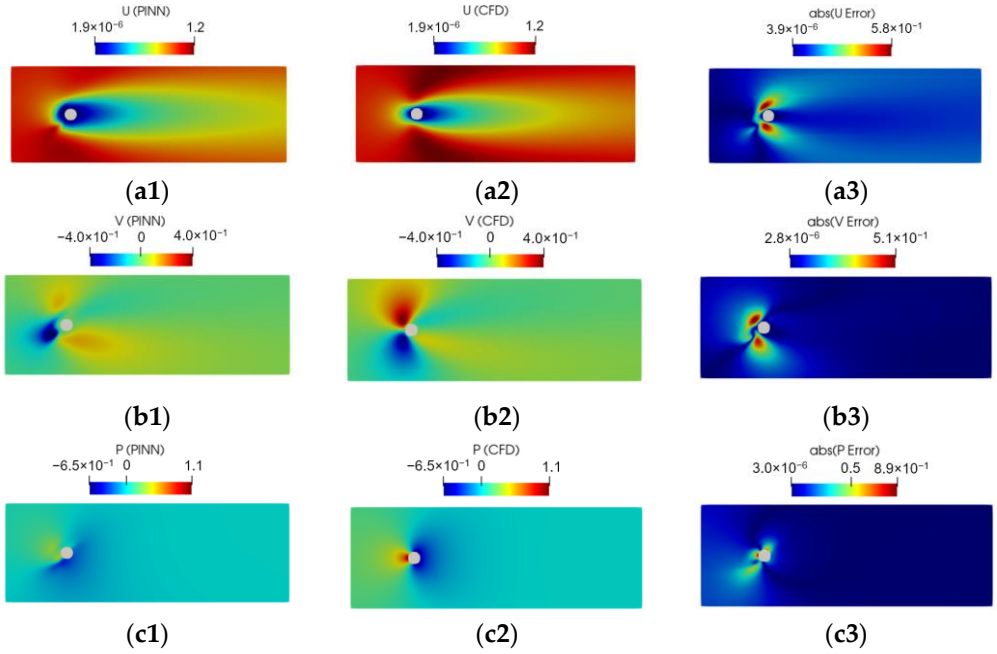

Figure 6. Flow fields near a cylinder for 2000 points from PINN (**left**), CFD (**middle**) and error between PINN and CFD (**right**). First row is for u velocity, second row is for v velocity, and third row is for P pressure. (**a1**) u velocity field predicted by PINN, (**a2**) u velocity field computed by CFD, (**a3**) error magnitude for u velocity, (**b1**) v velocity field predicted by PINN, (**b2**) v velocity field computed by CFD, (**b3**) error magnitude for v velocity, (**c1**) pressure field predicted by PINN, (**c2**) pressure field computed by CFD, (**c3**) error magnitude for pressure.

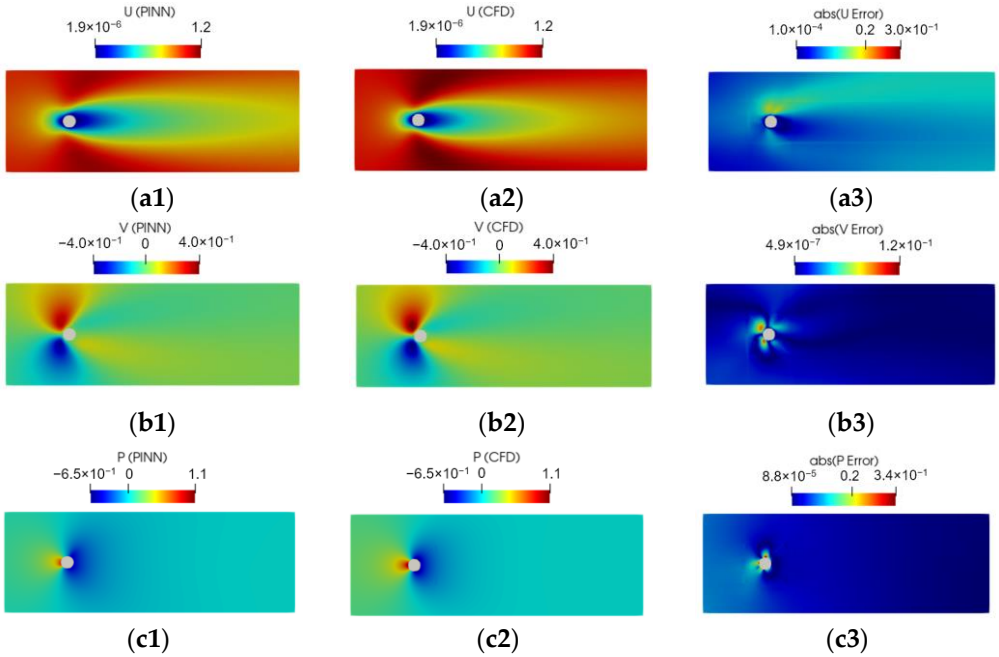

Figure 7. Flow fields near a cylinder for 10,000 points from PINN (**left**), CFD (**middle**) and error between PINN and CFD (**right**). First row is for u velocity, second row is for v velocity, and third row is for P pressure. (**a1**) u velocity field predicted by PINN, (**a2**) u velocity field computed by CFD, (**a3**) error magnitude for u velocity, (**b1**) v velocity field predicted by PINN, (**b2**) v velocity field computed by CFD, (**b3**) error magnitude for v velocity, (**c1**) pressure field predicted by PINN, (**c2**) pressure field computed by CFD, (**c3**) error magnitude for pressure.

For 15,000 points in the point cloud, increasing the number of points from 10,000 to 15,000 provided no improvement to the accuracy of the velocity and pressure profiles from PINN. The general trend of the wake and profiles are well captured; however the magnitudes captured by PINN are generally lower than the magnitudes obtained from CFD.

Figure 8 shows a plot of the u and v velocity components as well as the pressure of the flow field obtained from the top of the cylinder surface to the top edge of the computational domain for the different number of points in the point cloud. Looking at the plots, it is evident that 2000 points is insufficient, as the velocity and pressure values deviate from the CFD results to a large extent. For 5000 to 15,000 points, the near field velocities and pressure agrees with CFD. In the far field, the u velocity components from PINN match closely to CFD results; however, significant deviation is observed for the v velocity component and pressure values. This could be attributed to the relatively small magnitudes when compared to the u velocities, which results in difficulty in the reduction of error for the v velocity and pressure. The accuracy of PINN can potentially be improved by adding a scaling factor to the errors related to v velocity and pressure to facilitate training.

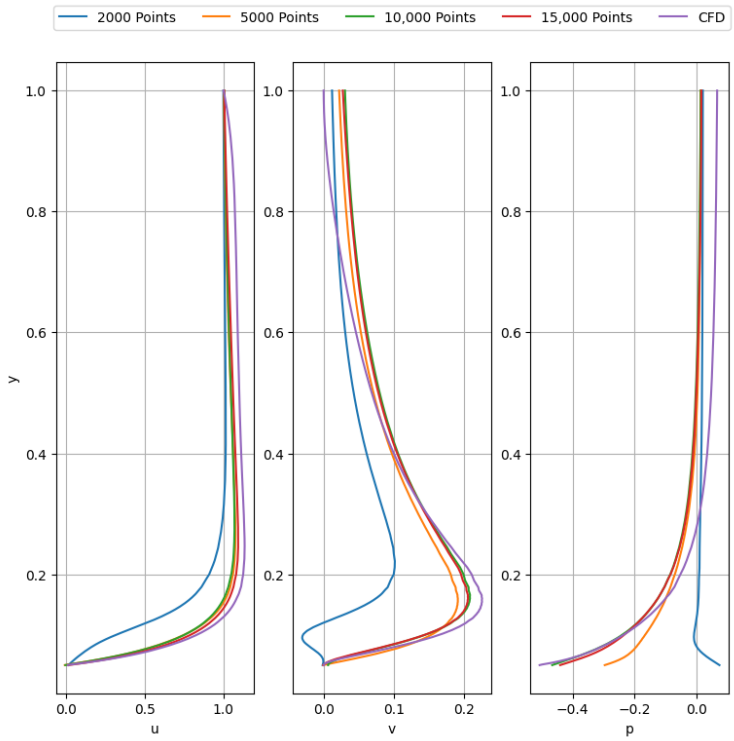

Figure 8. Line plots of u velocity, v velocity, and pressure field from top surface of cylinder to edge of computational domain for varying number of points in the point cloud.

3.2. Effect of Number of Hidden Layers

Another important parameter in the investigation of PINN is to define the number of hidden layers on the accuracy of the results. Hidden layers essentially create composite functions. By having a larger number of hidden layers, PINN is able to capture more complex problems such as those with large gradients or discontinuities. Having an insufficient number of hidden layers will result in simple, smooth solutions, which may compromise on accuracy. To study the effects, the number of points in the point cloud and number of nodes per layer are kept at 5000 points and 50 nodes per hidden layer, respectively. The accuracy of PINN is then investigated for 10, 15, 20, and 25 hidden layers.

First, for 10 hidden layers, the u velocity flow field around the cylinder is shown in Figure 9 where PINN is unable to produce accurate results. The results for 15 hidden layers are similar to the result for 10 hidden layers, where PINN is unable to capture the velocity and pressure fields around the cylinder accurately.

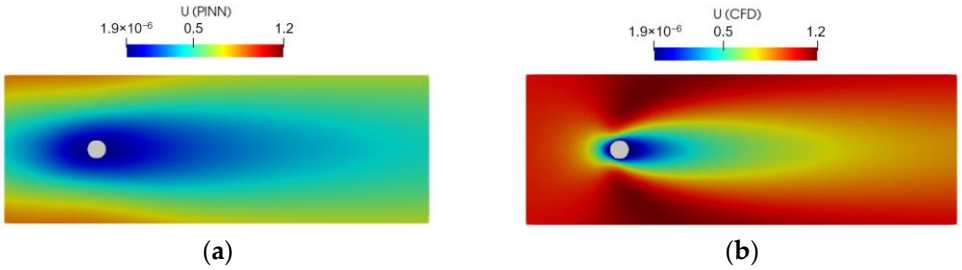

Figure 9. Velocity flow fields (u) near a cylinder for 10 hidden layers from PINN (**a**) and CFD (**b**).

For 20 hidden layers, PINN started to capture the general trends of the flow field, although the magnitudes obtained from PINN for the velocity and pressure fields are slightly diminished when compared to the CFD results.

With 25 hidden layers in Figure 10 and looking at the u velocity component, the leading-edge and wake profile captured by PINN largely agrees with CFD. However, there is a reduction in the magnitude and area of accelerated flow when compared to CFD. Additionally, the wake velocity magnitudes are also lower for PINN. The PINN is able to capture the v velocity profile accurately, and the region of negative velocity below the cylinder is also well captured. The PINN also managed to predict the small spot of high velocity in the region above the cylinder. With 25 hidden layers, the stagnation point is found on the middle of the leading-edge of the cylinder. Regions of high and low pressure are accurately predicted, with the values predicted by PINN being marginally lower than the results computed by CFD.

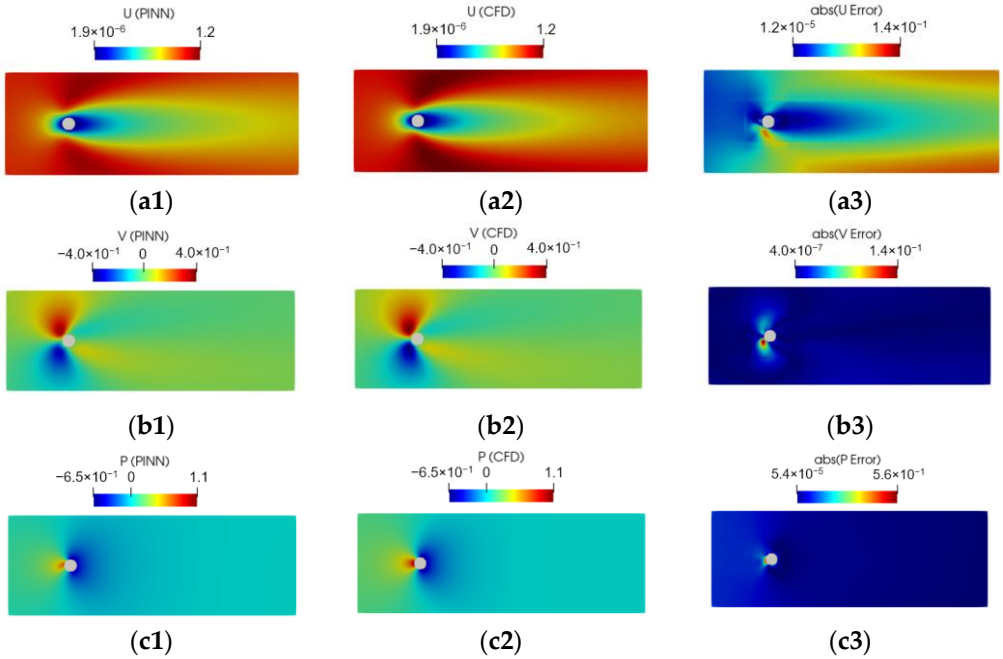

Figure 10. Flow fields near a cylinder for 25 layers from PINN (**left**), CFD (**middle**) and error between PINN and CFD (**right**). First row is for u velocity, second row is for v velocity, and third row is for P pressure. (**a1**) u velocity field predicted by PINN, (**a2**) u velocity field computed by CFD, (**a3**) error magnitude for u velocity, (**b1**) v velocity field predicted by PINN, (**b2**) v velocity field computed by CFD, (**b3**) error magnitude for v velocity, (**c1**) pressure field predicted by PINN, (**c2**) pressure field computed by CFD, (**c3**) error magnitude for pressure.

The line plots of the velocity components and pressure for varying number of hidden layers is shown in Figure 11. The PINN with 10 and 15 layers failed to follow similar trends with CFD, suggesting that more hidden layers are needed to better represent the flow field. This observation is consistent with the contour plots shown in Figure 9. The PINN with

20 and 25 layers managed to capture the trends in the near field region; however, slight deviations are observed in the far field for v velocity and pressure. Similar to the previous case, the small magnitudes of the v velocity and pressure resulted in greater difficulty in the training of the PINN.

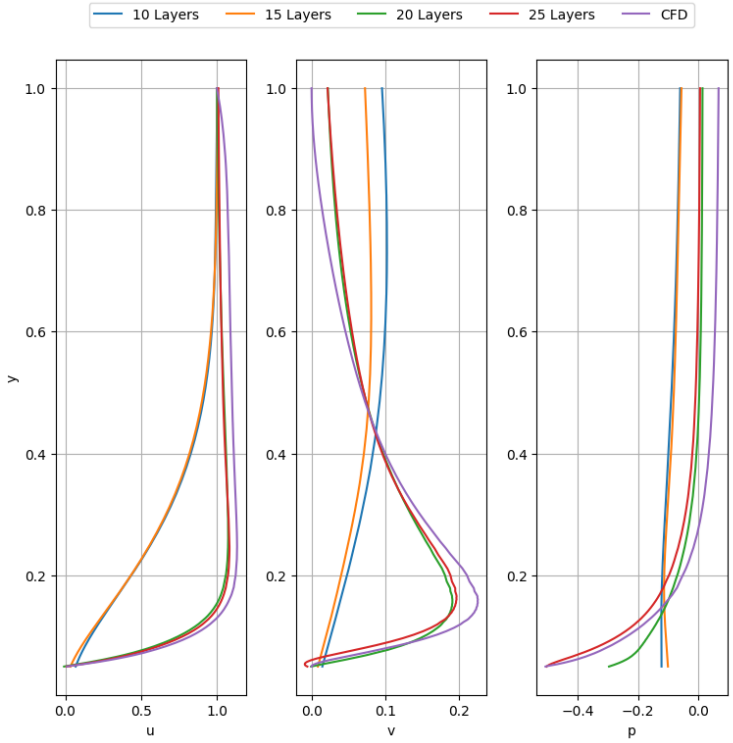

Figure 11. Line plots of u velocity, v velocity, and pressure field from top surface of cylinder to edge of computational domain for varying number of hidden layers.

3.3. Effect of Number of Nodes per Hidden Layer

To evaluate the effect of number of nodes per hidden layer, the number of points in the point cloud and number of hidden layers are kept as constants. The number of nodes affects the expressive power of PINN to fully resolve all scales of the neural network. Similar to how the Fourier series works, the more layers in the network, the more refined the small-scale solutions. The PINN made use of a point cloud consisting of 5000 points, with 20 hidden layers. The accuracy of PINN is investigated for 30, 40, 50, and 60 nodes per hidden layers.

First, a neural network utilizing only 30 nodes fails to accurately predict the flow fields for u velocity component, v velocity component, and pressure. Significant improvement in the accuracy can be seen when the number of nodes per hidden layer is increased from 30 to 40. However, some discrepancies between the results from PINN and CFD still prevails. For instance, for u velocity component calculated by PINN, the leading-edge stagnation point is displaced slightly upwards. The wake and pressure fields showed good agreement between PINN and CFD, but the magnitude of u and v velocities are lower for PINN.

For 50 nodes, there is some improvement in the accuracy of the results produced by PINN. The wake and leading-edge u velocity components are in good agreement with each other, apart from the lower velocity magnitudes around the flow acceleration region captured by PINN. Similarly, the v velocity components calculated by PINN follows a similar trend when compared the CFD result. The pressure field computed by PINN are in good agreement with the CFD results. The regions of high pressure upwind of the cylinder, and regions of low pressure downwind of the cylinder are well captured, the exception of the stagnation pressure being of a lower magnitude when compared with the CFD result.

When 60 nodes are used for the cylinder in Figure 12(a1,a2), the wake region of the u velocity component computed by PINN is longer than the wake computed by CFD. The leading-edge region computed by PINN also had a cone shaped extension when compared with the CFD result. The flow acceleration region is not well captured by PINN as well. Looking at Figure 12(b1,b2), PINN is able to capture the leading-edge profile and the wake. However, the velocity magnitudes in the core of the leading-edge profile are smaller, and of lower magnitudes than CFD. Some discrepancies are observed in the pressure profile captured by PINN in Figure 12(c1) when compared to the CFD results in Figure 12(c2). The PINN failed to adequately capture the high-pressure stagnation point on the leading-edge of the cylinder. The downwind low-pressure region is also not symmetrical about the x-axis. In general, increasing from 50 to 60 nodes did not improve the accuracy of PINN.

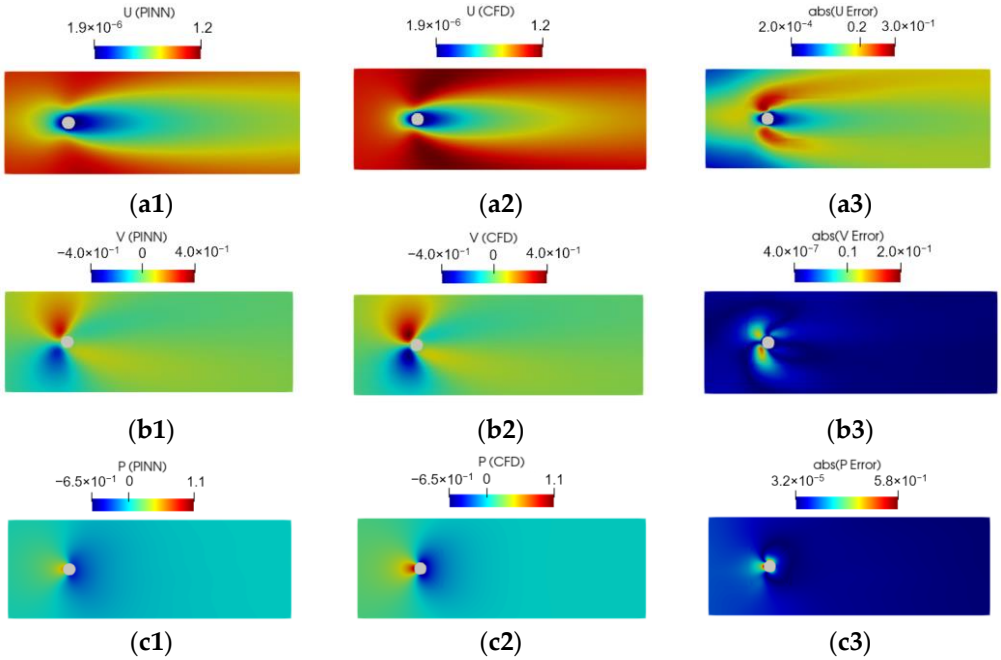

Figure 12. Flow fields near a cylinder for 60 nodes per layer from PINN (**left**), CFD (**middle**) and error between PINN and CFD (**right**). First row is for u velocity, second row is for v velocity, and third row is for P pressure. (**a1**) u velocity field predicted by PINN, (**a2**) u velocity field computed by CFD, (**a3**) error magnitude for u velocity, (**b1**) v velocity field predicted by PINN, (**b2**) v velocity field computed by CFD, (**b3**) error magnitude for v velocity, (**c1**) pressure field predicted by PINN, (**c2**) pressure field computed by CFD, (**c3**) error magnitude for pressure.

Figure 13 plots the u and v velocity components and pressure to for both CFD and PINN with varying number of nodes per hidden layer. For the u velocity component, the PINN using 30 nodes had significant deviations from the results obtained from CFD. The PINN with 40 and 50 nodes performed much better, with the results matching closely with CFD. However, increasing the number of nodes to 60 resulted in a drop of performance, consistent with the flow fields observed from Figure 12. The PINN with 50 and 60 nodes are able to capture the near field v velocity with much accuracy, but similar to the previous cases, slight deviations can be observed in the far field.

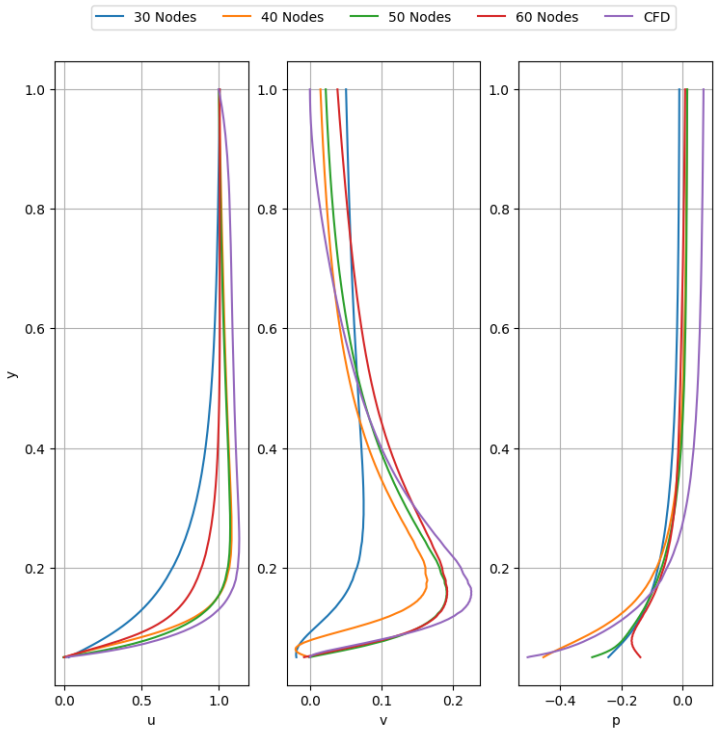

Figure 13. Line plots of u velocity, v velocity, and pressure field from top surface of cylinder to edge of computational domain for varying number of nodes per hidden layer.

4. Discussions

In general, the magnitudes computed by the PINN are smaller than the CFD results. This slight discrepancy could be due to the difficulty in the computation of pressure. Pressure is not explicitly defined in the momentum equations, and is only indirectly enforced by the continuity equation. One possible way to improve the performance of PINN is to construct an equation for pressure, for example, the pressure Poisson equation, and incorporate it as part of the loss to the neural network. However, the validity and performance of this method has yet to be investigated.

4.1. Maximum Error between PINN and CFD

The absolute error between PINN and CFD is defined as

$$\epsilon_i(\boldsymbol{x}) = \left| \widetilde{y}_i(\boldsymbol{x}) - y_i(\boldsymbol{x}) \right| \tag{16}$$

where $\widetilde{y}_i$ represents the pointwise flow variables predicted by PINN, y_i represents the pointwise flow variables computed by CFD, and $i = u, v, p$ are the index which represents u velocity, v velocity and pressure, respectively. The maximum absolute error between PINN and CFD is shown in Figure 14 for u and v velocity components for varying number of points, hidden layers, and nodes per hidden layer. Looking at the results for number of points, the accuracy stagnates after reaching 5000 points in the point cloud. This is an indication that once the minimum number of points required has been used, an increase in the number of points does not lead to a more accurate velocity field. For the pressure field error in Figure 14, the case with 10,000 points achieved the lowest maximum absolute error and increasing the number of points resulted in amplification of the error. This is due to difficulty in training the pressure field as pressure is implicitly defined in the Navier-Stokes equation. Increasing the number of points led to an increase in difficulty in training of the pressure field.

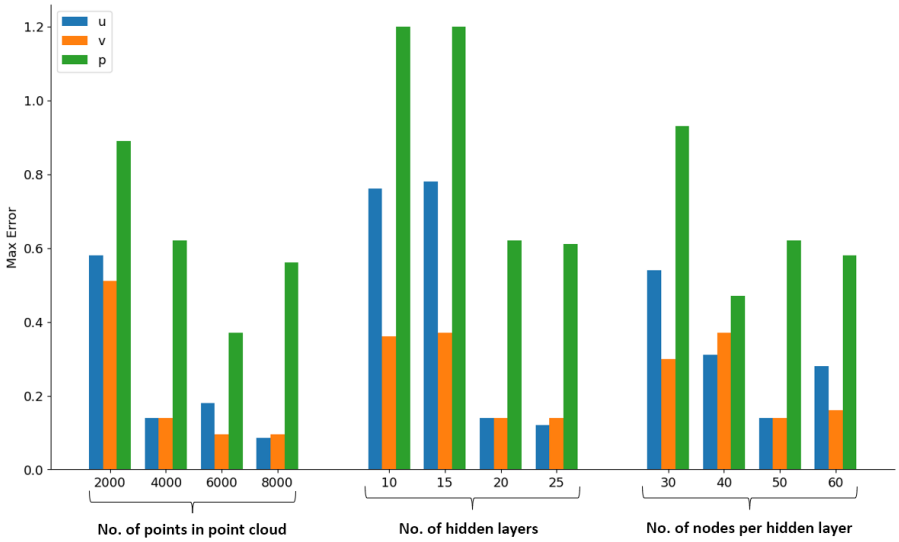

Figure 14. Maximum absolute error of PINN against number of points in point cloud, hidden layers and nodes per hidden layer for u velocity component (blue), v velocity component (orange), and pressure field (green).

For the investigations of the effect of number of hidden layers on the accuracy of the PINN, the maximum error for all flow fields is high for 10 and 15 hidden layers, which suddenly drops when it reaches 20 hidden layers. Having a low number of hidden layers prevented PINN from generalizing the solution for the entire domain. There are no significant improvements in accuracy when the number of hidden layers increased from 20 to 25 layers. This suggests that the accuracy for these cases reaches a limit at 20 hidden layers, and any additional layers will not improve the results any further. While the maximum error did not decrease significantly when increasing the number of hidden layers from 20 to 25, the accuracy of the wake produced by PINN decreased significantly. PINN is unable to obtain accurate wake velocity magnitudes for the network using 25 hidden layers. This could be largely due to an increase in the difficulty of training when the network size gets larger, leading to a less accurate result.

For different number of nodes per hidden layers in the network, insufficient nodes render the PINN unable to fully generalize the solution throughout the entire spatial scale. This is akin to the Fourier series, where a larger number of terms are required to capture the small-scale changes in the function. It is interesting to observe that as the number of nodes per hidden layer increases from 50 to 60, the accuracy of PINN results decreases. In other words, when the number of nodes per layer reaches a certain threshold, the performance of PINN remains stagnant, or in this case, worsens. Increasing the number of nodes per hidden layer increases the dimensions of the training parameters, which makes training of the PINN harder, resulting in a less accurate result.

From the study, increasing the number of nodes per hidden layer provided the least improvement in performance for the PINN. As observed, there is a saturation point in which increasing the number of nodes per layer led to no significant increase in accuracy. Going beyond this saturation point only brings about unnecessary waste of memory resources, and potential increase in computational time. Since the simulations are performed on a GPU, increment in the parameter values did not increase the computation time significantly. However, as GPU memory is limited, any increase in the number of points, number of hidden layers, or number of nodes per hidden layer results in poor optimization of GPU memory resource. Hence, finding the optimal network parameters is key to the performance of the neural network. Observing the error reduction as the parameter changes, increasing the number of hidden layers brings about the largest reduction of error. Hence, when optimizing the PINN, adjustment to the number of hidden layers should be the priority.

The next best option is to increase the number of points in the point cloud. Increasing the number of nodes per hidden layer brings about the smallest improvement in performance.

4.2. Comparison of Resource Utilzation

The comparison of computational resource usage by PINN and CFD against number of points in the point cloud is shown in Figure 15. As observed, the computation time for PINN shows a non-linear time complexity, with respect to the number of points in the point cloud, where the time taken for PINN to solve for flow over cylinder varies quadratically with the number of points. Thus, having a larger number of points will lead to a drastic increase in the computation time. Looking at the memory usage, PINN memory usage increases linearly with the number of points. When comparing the time taken for PINN with CFD, PINN takes almost 3 times longer to solve when using 5000 points. However, when comparing the memory usage between PINN and CFD, PINN uses more than 10 times less memory. While PINN takes more time to solve the problem, it is more efficient in the computational resource's aspect.

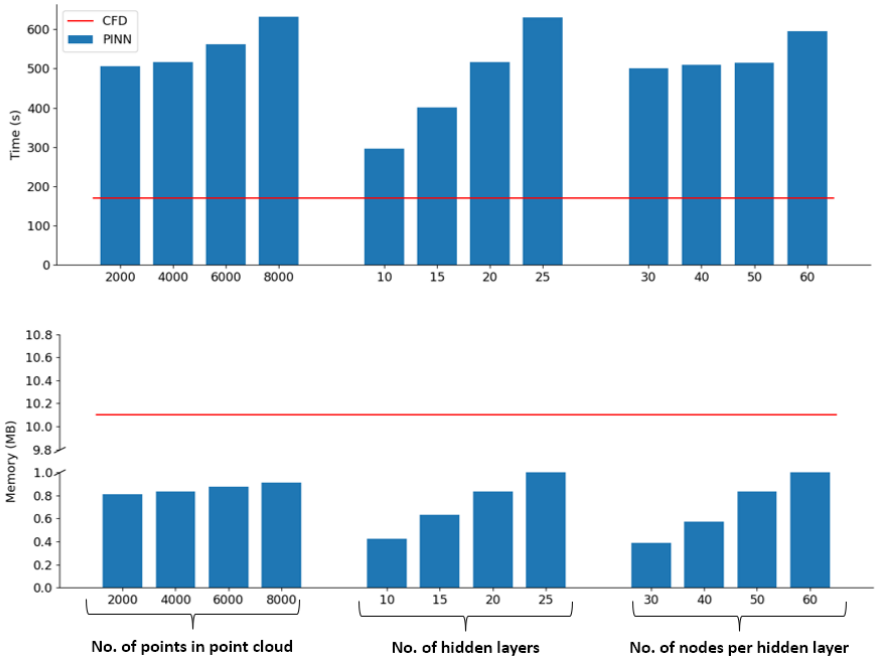

Figure 15. Computational resources used by PINN and CFD for varying number of points, hidden layers, and nodes per hidden layer. (**Top**) Comparison of computational time between PINN (blue) and converged CFD (red). (**Bottom**) Comparison of memory usage between PINN (blue) and converged CFD (red).

PINN has a linear time complexity with respect to the number of layers. Comparing the time taken by PINN with CFD, PINN took almost 3 times the amount of time to solve for flow around cylinder using 20 hidden layers. In terms of computation time, CFD is still more efficient. However, PINN is able to make up for that when it comes to memory usage. The PINN memory usage varies linearly as the number of hidden layers increases. PINN is more than 5 times more efficient in memory usage than CFD, making it less demanding on the hardware requirement for solving the problem using PINN.

For different number of nodes per hidden layer, the time complexity for PINN is linear for 30 to 50 nodes; however, there is a larger jump in computational time when the number of nodes per hidden layer increases from 50 to 60. When comparing the computation time with CFD, CFD is more computationally efficient, taking almost 3 times less time to solve for the flow around the cylinder. The memory usage of PINN is more than 5 times less than the memory usage of CFD, making PINN more efficient on the hardware and resources.

In general, the computational time for PINN is at least 3 times longer than CFD. The automatic differentiation process is the main cause for the long computation time. Automatic differentiation requires an additional backpropagation step, where partial derivatives of every parameter (nodal weights) have to be computed, and the derivatives with respect to the spatial coordinates can be obtained via the chain rule. This additional backpropagation step scales with the number of layers and number of nodes, and, hence, resulted in higher computational cost for large PINN architectures. Another reason CFD is more efficient in this case is that the CFD mesh is simple, with relatively low number of cells. With more cells in the mesh, CFD could potentially be less efficient in computation time. This can be highlighted when compared to a similar study using PINN to solve for low Reynolds number flow over an airfoil [35]. Table 5 compared the computational time used by CFD and PINN for two cases with varying number of cells for CFD. It can be seen that for cases requiring finer CFD mesh, the PINN is 4.5 times more computationally efficient than CFD. For the airfoil case, a larger number of cells is required, which resulted in a huge increase in computational time for CFD. However, the computational time for PINN remains relatively constant. Hence, in more complex cases with finer mesh, PINN can potentially use significantly less computation time.

However, what PINN lack in time efficiency, it made up in terms of the space efficiency. In all cases, PINN require about 10 times less memory when compared to CFD. The data format for PINN are simple tensors which only includes the spatial coordinates, and weights and bias of the neural network. Unlike PINN, the data which defines the CFD mesh is more complicated, including the cell vertices, cell faces and cell neighborhood, which results in larger memory usage for the CFD.

Table 5. Comparison of computational time between CFD and PINN.

Study	Number of Cells	CFD Time	PINN Time
Cylinder	41,772	170	~500
Airfoil [35]	100,172	1687	~500

5. Conclusions

Physics-informed neural network (PINN) architecture for fluid flows is demonstrated for a low Reynolds number flow around a cylinder without the use of experimental or simulation data. The PINN is able to capture the general trend of the flow field. Velocity fields produced by PINN are comparable with the results captured by CFD. However, PINN struggled slightly with the pressure fields as the pressure term is not explicitly defined in the loss function. Critical factors of designing the PINN architecture have been investigated and discussed, and showed that increasing the number of layers leads to the greatest improvement to the accuracy of results, followed by increasing number of points in the point cloud. Increasing the number of nodes per hidden layer brings about the smallest improvement in performance.

In terms of computational demands, PINN is more efficient in memory usage than CFD but not necessary for computation time. In simple cases where the number of cells in the CFD mesh is low, PINN takes longer than CFD to compute the results. However, in more complex cases that involves higher number of cells in the CFD mesh, PINN has the potential to be more time efficient.

Author Contributions: Conceptualization, E.H.W.A. and B.F.N.; formal analysis, E.H.W.A., G.W. and B.F.N.; investigation, E.H.W.A.; writing—original draft preparation, E.H.W.A.; writing—review and editing, G.W. and B.F.N.; supervision, B.F.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Nanyang Technological University, Singapore under the Nanyang President's Graduate Scholarship, and the Singapore Centre for 3D Printing, which is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Medium-Sized Centre funding Scheme.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shyy, W.; Lian, Y.; Tang, J.; Liu, H.; Trizila, P.; Stanford, B.; Bernal, L.; Cesnik, C.; Friedmann, P.; Ifju, P. Computational aerodynamics of low Reynolds number plunging, pitching and flexible wings for MAV applications. *Acta Mech. Sin.* **2008**, *24*, 351–373. [CrossRef]
2. Shanko, E.S.; van de Burgt, Y.; Anderson, P.D.; den Toonder, J.M. Microfluidic magnetic mixing at low Reynolds number and in stagnant fluids. *Micromachines* **2019**, *10*, 731. [CrossRef]
3. Moukalled, F.; Mangani, L.; Darwish, M. *The Finite Volume Method in Computational Fluid Dynamics*; Springer: Cham, Switzerland, 2016.
4. Lindhorst, K.; Haupt, M.C.; Horst, P. Reduced-order modelling of non-linear, transient aerodynamics of the HIRENASD wing. *Aeronaut. J.* **2013**, *120*, 601–626. [CrossRef]
5. Thuerey, N.; Weißenow, K.; Prantl, L.; Hu, X. Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows. *AIAA J.* **2020**, *58*, 25–36. [CrossRef]
6. Zhang, Y.; Sung, W.; Mavris, D. Application of Convolutional Neural Network to Predict Airfoil Lift Coefficient. In Proceedings of the 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, USA, 8–12 January 2018.
7. Belbute-Peres, F.D.A.; Economon, T.; Kolter, Z. Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 13–18 July 2020.
8. Han, R.; Wang, Y.; Zhang, Y.; Chen, G. A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network. *Phys. Fluids* **2019**, *31*, 127101.
9. Samek, W.; Montavon, G.; Vedaldi, A.; Hansen, L.K.; Muller, K.R. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*; Springer: Cham, Switzerland, 2019.
10. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2018**, *378*, 686–707. [CrossRef]
11. Chauvin, Y.; Rumelhart, D.E. *Backpropagation: Theory, Architectures, and Applications*; Lawrence Erlbaum Associates Publishers: Hillsdale, NJ, USA, 1995.
12. Raissi, M.; Wang, Z.; Triantafyllou, M.S.; Karniadakis, G.E. Deep learning of vortex-induced vibrations. *J. Fluid Mech.* **2019**, *861*, 119–137. [CrossRef]
13. Sun, L.; Gao, H.; Pan, S.; Wang, J.X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comput. Methods Appl. Mech. Eng.* **2020**, *361*, 112732. [CrossRef]
14. Mao, Z.; Jagtap, A.D.; Karniadakis, G.E. Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.* **2019**, *360*, 112789. [CrossRef]
15. Cai, S.; Mao, Z.; Wang, Z.; Yin, M.; Karniadakis, G.R. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mech. Sin.* **2021**, *37*, 1727–1738. [CrossRef]
16. Arzani, A.; Wang, J.X.; D'Souza, R.M. Uncovering near-wall blood flow from sparse data with physics-informed neural networks. *Phys. Fluids* **2021**, *33*, 071905. [CrossRef]
17. Eivazi, H.; Tahani, M.; Schlatter, P.; Vinuesa, R. Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations. *Phys. Fluids* **2021**, *34*, 075117. [CrossRef]
18. Jin, X.; Cai, S.; Li, H.; Karniadakis, G.E. NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *J. Comput. Phys.* **2021**, *426*, 109951. [CrossRef]
19. Zhu, Q.; Liu, Z.; Yan, J. Machine learning for metal additive manufacturing: Predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Comput. Mech.* **2021**, *67*, 619–635. [CrossRef]
20. Almajid, M.M.; Abu-Al-Saud, M.O. Prediction of porous media fluid flow using physics informed neural networks. *J. Pet. Sci. Eng.* **2022**, *208*, 109205. [CrossRef]
21. Bararnia, H.; Esmaeilpour, M. On the application of physics informed neural networks (PINN) to solve boundary layer thermal-fluid problems. *Int. Commun. Heat Mass Transf.* **2022**, *132*, 105890. [CrossRef]
22. Aliakbari, M.; Mahmoudi, M.; Vadasz, P.; Arzani, A. Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport solvers using physics-informed neural networks. *Int. J. Heat Fluid Flow* **2022**, *96*, 109002. [CrossRef]
23. Yang, X.I.A.; Zafar, S.; Wang, J.-X.; Xiao, H. Predictive large-eddy-simulation wall modeling via physics-informed neural networks. *Phys. Rev. Fluids* **2019**, *4*, 034602. [CrossRef]
24. Lucor, D.; Agrawal, A.; Sergent. Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection. *J. Comput. Phys.* **2022**, *456*, 111022. [CrossRef]
25. Haykin, S. *Neural Networks: A Comprehensive Foundation*; Pearson Education: London, UK, 1999.
26. Baydin, A.G.; Pearlmutter, B.A.; Radul, A.A.; Siskind, J.M. Automatic Differentiation in Machine Learning: A Survey. *J. Mach. Learn. Res.* **2018**, *18*, 1–43.

27. Caudill, M.; Butler, C. *Naturally Intelligent Systems*; Massachusetts Institute of Technology: Cambridge, MA, USA, 1992.
28. PyTorch. PyTorch Documentation. 2019. Available online: https://pytorch.org/docs/stable/index.html (accessed on 2 June 2021).
29. Kingma, D.P.; Ba, J.L. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
30. OpenFOAM. OpenFOAM Documentation. 22 December 2020. Available online: https://www.openfoam.com/documentation/overview (accessed on 31 May 2021).
31. Tritton, D.J. Experiments on the flow past a circular cylinder at low Reynolds numbers. *J. Fluid Mech.* **1959**, *6*, 547–567. [CrossRef]
32. Posdziech, O.; Grundmann, R. A systematic approach to the numerical calculation of fundamental quantities of the two-dimensional flow over a circular cylinder. *J. Fluids Struct.* **2007**, *23*, 479–499. [CrossRef]
33. Cybenko, G. Approximation by Superpositions of a Sigmoidal Function. *Math. Control Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
34. Hornik, K. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Netw.* **1991**, *4*, 251–257. [CrossRef]
35. Ang, E.; Ng, B.F. Physics-Informed Neural Networks for Flow Around Airfoil. In Proceedings of the AIAA SCITECH 2022 Forum, San Diego, CA, USA, Virtual, 3–7 January 2022. [CrossRef]