Mathematical and Computational Applications

Article

Evaluation of Physics-Informed Neural Network Solution Accuracy and Efficiency for Modeling Aortic Transvalvular Blood Flow

Jacques Francois Du Toit and Ryno Laubscher *

Department of Mechanical and Mechatronic Engineering, Stellenbosch University, Private Bag X1, Matieland, Stellenbosch 7602, South Africa; jacf.dt@gmail.com
* Correspondence: rlaubscher@sun.ac.za

Abstract: Physics-Informed Neural Networks (PINNs) are a new class of machine learning algorithms that are capable of accurately solving complex partial differential equations (PDEs) without training data. By introducing a new methodology for fluid simulation, PINNs provide the opportunity to address challenges that were previously intractable, such as PDE problems that are ill-posed. PINNs can also solve parameterized problems in a parallel manner, which results in favorable scaling of the associated computational cost. The full potential of the application of PINNs to solving fluid dynamics problems is still unknown, as the method is still in early development: many issues remain to be addressed, such as the numerical stiffness of the training dynamics, the shortage of methods for simulating turbulent flows and the uncertainty surrounding what model hyperparameters perform best. In this paper, we investigated the accuracy and efficiency of PINNs for modeling aortic transvalvular blood flow in the laminar and turbulent regimes, using various techniques from the literature to improve the simulation accuracy of PINNs. Almost no work has been published, to date, on solving turbulent flows using PINNs without training data, as this regime has proved difficult. This paper aims to address this gap in the literature, by providing an illustrative example of such an application. The simulation results are discussed, and compared to results from the Finite Volume Method (FVM). It is shown that PINNs can closely match the FVM solution for laminar flow, with normalized maximum velocity and normalized maximum pressure errors as low as 5.74% and 9.29%, respectively. The simulation of turbulent flow is shown to be a greater challenge, with normalized maximum velocity and normalized maximum pressure errors only as low as 41.8% and 113%, respectively.

Keywords: Physics-Informed Neural Networks; Computational Fluid Dynamics; aortic valve; valvular stenosis; turbulent flow

check for updates

Citation: Du Toit, J.F.; Laubscher, R. Evaluation of Physics-Informed Neural Network Solution Accuracy and Efficiency for Modeling Aortic Transvalvular Blood Flow. *Math. Comput. Appl.* **2023**, *28*, 62. https://doi.org/10.3390/mca28020062

Academic Editors: Hans Beushausen and Sebastian Skatulla

Received: 15 February 2023
Revised: 10 March 2023
Accepted: 30 March 2023
Published: 14 April 2023

1. Introduction

Computational Fluid Dynamics (CFD) methods have become an essential tool in many engineering industries, from aerospace to biomedical. While traditional CFD methods, such as the Finite Volume Method (FVM), are proven and versatile tools, they have limitations, such as being ill-suited to solving ill-posed problems where the solution space is not fully confined by the boundary conditions alone. In heat transfer and biomedical applications, it is often difficult or impossible to determine exact boundary conditions for simulations; however, sparsely distributed measurement data are often readily available: the incorporation of these measurements could theoretically be used to constrain the solution, and to allow for the solution of the unknown boundary conditions. Unfortunately, this is a prohibitively expensive task to perform using traditional methods.

A further limitation of traditional CFD methods is the high computational expense of many-query-analysis type problems, such as simulation-based design, real-time monitoring using virtual sensors and model-predictive control: in these applications, the solution of a partial differential equation (PDE) is required for a large number of different boundary conditions or material properties. The computational expense can be partially alleviated by using surrogate models [1], but the construction of a data-fit surrogate model still requires a large dataset to be collected before the model can be constructed.

The physics informed neural network (PINN), first introduced by Raissi et al. [2], is an alternative fluid simulation and modeling technique that could potentially overcome the limitations mentioned above, and thus be a valuable tool in fluid dynamics research and modeling efforts. The PINN is a versatile, deep-learning-based modeling technique that allows for the solving of PDEs [3], the construction of surrogate models [4] and the solving of ill-posed problems [5].

With a PINN, a neural network is used as a general function approximator, and is trained to approximate the solution of a PDE. Typically, the training of a neural network requires a large amount of labeled training data: this requirement is lifted, by assembling a loss function that enforces known physics. Given that the problem is well-posed, the PINN method can be used to solve PDEs without any labeled data [3,4,6–9].

Currently, PINNs are more computationally expensive than traditional CFD techniques, when simulating a single, well-posed, unparameterized problem. However, when the dimensionality of the problem is increased (for example if we want to characterize the performance of an airfoil over a range of geometric parameters), the computational expense of PINNs scales favorably, due to the fact that PINNs can be trained on the entire parameter space simultaneously. Once a PINN has been trained, it can be used to rapidly infer solutions to the problem for any value of the parameter inside the training range. This makes PINNs promising candidates for use in system-level models, design optimization, real-time monitoring and control applications.

It has been shown by Sun et al. [4] that PINNs can be used to solve the Navier–Stokes Equations on a parameterized domain without labeled data: here, PINNs were presented as a method of constructing surrogate models in a data-sparse setting. The authors argued that in many scientific applications, large training datasets are not available, and that traditional data-fit surrogate models fail when the physical system exhibits strong nonlinearity. The authors constructed surrogate models for pipe flows with variable viscosity and laminar flow through a simplified artery with a variable degree of stenosis or aneurysm. The degree of stenosis or aneurysm was treated as a continuous parameter. They showed that PINNs can be used to solve the parameterized problem accurately without any labeled training data, and that the resulting surrogate model can be used to rapidly infer the flow field for any value of the geometry parameter inside the training range.

PINNs are also capable of solving multi-physics problems that have a greater number of transport equations that must be solved simultaneously. It was shown by Laubscher [9] that PINNs can accurately solve an incompressible, viscous flow problem with heat transfer and species diffusion. A dry air humidification problem was simulated, by using a mixed variable approach [10], which avoided higher-order derivatives in the governing equations. The author also introduced a segregated-network architecture that used three separate neural networks to approximate the energy equation, the species diffusion equation and the Navier–Stokes equations, respectively: this architecture was shown to be more accurate than the standard, single-network PINN architecture. A hyperparameter search was performed, to investigate the importance of the number of hidden layers, the number of neurons per layer and the number of sampling points in the domain. A parameterized version of the problem was also solved, to illustrate the potential of PINNs for surrogate modeling. The PINN performance was evaluated by comparing the results to a reference, finite volume method (FVM) solution.

In this paper, we attempted to use PINNs as a PDE integrator, meaning that we used PINNs to solve a well-posed problem without the use of any labeled training data: this is currently a challenging task for PINNs; there are, however, many examples in the literature where PINNs are used in novel ways that include the use of sparse training data [5,11–14].

In the field of biomedical engineering, patient-specific modeling is becoming an attainable and desirable practice; however, uncertainty in the boundary conditions of flow problems makes this a difficult task to perform using traditional CFD methods. Arzani et al. [14] showed how PINNs can be used to recover high-resolution wall shear stress estimates in aneurysms and stenoses, using sparse, noisy velocity measurements and unknown boundary conditions. Wall shear stress is a difficult quantity to measure in vivo, but it has been shown to play an important role in cardiovascular disease [15]. Arzani et al. used synthetic data, but the method could be applied to real patient-specific modeling, by using 4D flow magnetic resonance imaging (MRI) data to get noisy, low-resolution velocity measurements, and then using PINNs to recover high-resolution wall shear stresses.

Another area where boundary conditions are often unknown is heat transfer. Heat transfer problems are plagued by the difficulty of determining exact thermal boundary conditions. In most engineering applications, idealized boundary conditions are used for simulations, and in industry, engineers rely on experience to effectively operate equipment: PINNs offer a way to bridge this gap between real-world conditions and mathematical models. Cai et al. [5] used PINNs to simulate various heat transfer problems with complex, unknown boundary conditions, by using sparse, synthetic measurement data to constrain the solution. They illustrated the simulation of single-phase, multiphase, moving-boundary, steady, and unsteady flow problems. They also presented a method for using PINNs to inform sensor placement, to minimize the number of measurements required to accurately solve the problem.

The simulation of turbulent flow using PINNs for forward mode problems has proved to be a challenging task: different approaches have been used, including solving Reynolds-Averaged Navier-Stokes (RANS) equations with [16] and without [13], a turbulence closure model, and direct numerical simulation [17]. The Reynolds decomposition allows the simulation of turbulent flows at a lower computational cost, compared to direct numerical simulation; however, RANS turbulence closure models introduce assumptions and simplifications that make them less generalizable [18]. Eivazi et al. [13] showed that PINNs can be used to solve the RANS equations without any specific modeling assumptions or closure models, and by only using measurement data on the problem boundaries: this includes measurements of the Reynolds stresses that are difficult to measure in practice.

The PINN method is still in its infancy, and requires much further development. In this paper, we evaluated the use of Physics-Informed Neural Networks (PINNs) as an alternative method of simulating fluid flow through a simplified 2D aortic valve geometry, with the end goal of developing surrogate models.

A variety of techniques from the literature were implemented that have improved the training and prediction performance of PINNs [7,19,20]. Flow in the laminar and turbulent regimes were simulated by solving the incompressible Navier–Stokes and RANS equations, respectively. The results are compared to those obtained by using the Finite Volume Method (FVM), and are shown to be in good agreement. The PINN method is shown to be a promising candidate for use in system-level models, design optimization, real-time monitoring, control applications and inverse modeling.

2. Case Study Setup

To demonstrate the use of PINNs in simulating fluid flow, and to compare the results to those obtained by using the FVM, a simple example problem was chosen to be simulated: *transvalvular blood flow through an aortic valve with various degrees of stenosis in the laminar and turbulent regimes.*

A simplified 2D geometry of an adult male aortic valve was used. As a result of the simlification, the solution was not necessarily an accurate approximation of real transvalvular blood flow: this was, however, not required, as the aim of the study was to discuss the application of PINNs to forward-mode fluid modeling, and to compare simulation results from the PINN method to the FVM method.

Figure 1 illustrates the computational domain used to represent a typical aortic valve. The inlet, outlet, wall and symmetry boundary conditions were imposed on the geometry, as shown in Figure 1c. The dimensions are tabulated in Table 1.

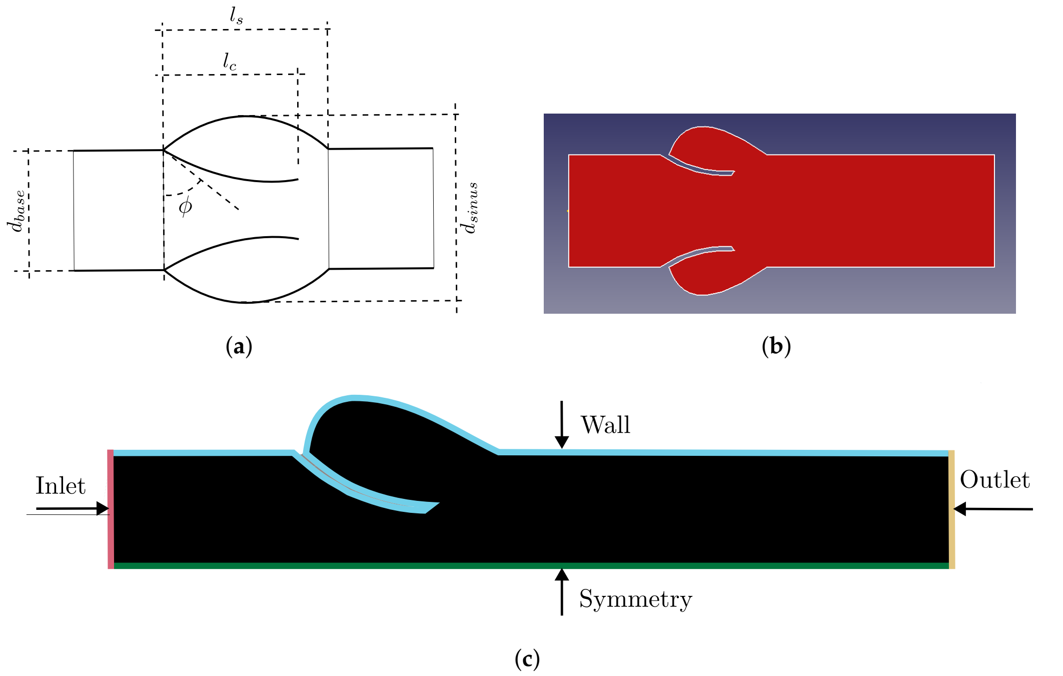


Figure 1. Simplified aortic valve geometry: (**a**) schematic representation, where d_{base} is the base diameter, ϕ is the valve angle, l_c is the length of the valve leaflet, l_s is the length of the sinus and d_{sinus} is the sinus diameter; (**b**) 2D computer aided design (CAD) model used as the simulation domain; the size of the simulation domain was reduced by using a symmetry plane along the axis, as shown in (**b**); (**c**) the computational domain, shown in black with boundary conditions, as imposed on the geometry.

Table 1. Aortic valve dimensions.

Parameter	Symbol	Value	Units	Reference
base diameter	d_{base}	24.7	[mm]	[21]
sinus diameter	d_{sinus}	36.1	[mm]	[22]
sinus length	l_s	21.5	[mm]	[22]
leaflet length	l_c	18.9	[mm]	[22]

Turbulent blood flow can be approximately described by the incompressible Reynolds-Averaged Navier–Stokes equations:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + (\mu+\mu_t)\nabla^2\mathbf{u} + \mathbf{f} \qquad \mathbf{x}, t \in \Omega_{f,t}\ ,\ \boldsymbol{\theta} \in \mathbb{R} \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \qquad \mathbf{x}, t \in \Omega_{f,t}\ ,\ \boldsymbol{\theta} \in \mathbb{R} \tag{2}$$

where $\mathbf{x}$ and t are dimensions of space and time, $\Omega_{f,t} \in \mathbb{R}$ is the problem domain in space and time, $\boldsymbol{\theta}$ is a vector of fluid properties—such as ρ: density and μ: viscosity—and μ_t is the turbulent viscosity. Given suitable boundary and initial conditions, the solution for the mean velocity field, $\mathbf{u}$, and the mean pressure field, p, is uniquely defined.

Here, we approximated the fluid to have a linear-strain-rate–stress relationship, with a dynamic viscosity of 0.004 Pa·s: this was accurate at the high Reynolds numbers that the transvalvular flow was expected to have [23]. A density of 1055 $\mathrm{kg/m^3}$ was used [24].

A turbulence closure model was required, to compute the turbulent viscosity. A mixing length model [18] was chosen for the PINN and the FVM simulations. Mixing length models have a very narrow range of applicability in modern CFD, as they are incapable of accurately representing flows that include recirculation or flow separation [18]. Modern CFD routinely uses turbulence models, such as the $k-\epsilon$ and $k-\omega$ models, which are much more appropriate for general flows; however, the current state of the art in data-less PINNs does not yet allow for the inclusion of these models, as they increase the stiffness of the system of equations to be solved.

The mixing length model was an algebraic model, which means that the turbulent viscosity was directly computed from the mean velocity field, without solving an additional set of equations. The turbulent viscosity was computed as follows:

$$\mu_t(\mathbf{x}) = \rho(l_m(\mathbf{x}))^2\sqrt{G(\mathbf{x})} \tag{3}$$

$$G(\mathbf{x}) = 2\left(\frac{\partial u}{\partial x}(\mathbf{x})\right)^2 + 2\left(\frac{\partial v}{\partial y}(\mathbf{x})\right)^2 + \left(\frac{\partial u}{\partial y}(\mathbf{x}) + \frac{\partial v}{\partial x}(\mathbf{x})\right)^2 \tag{4}$$

where l_m was the mixing length, and G was the strain-rate tensor. Various formulations can be used to compute an appropriate mixing length. The formulation used in this work is discussed later.

To simulate a laminar case, the governing equation was the same as the turbulent case, as shown in Equation (1), but the turbulent viscosity was simply set to zero.

Various degrees of stenosis were considered, by setting the valve leaflets to different angles, which restricted the flow by different degrees as shown in table 2. The area ratio was defined as the ratio of the area of the smallest restriction in the valve to the area of the base.

Table 2. Degrees of stenosis.

Valve Angle	Area Ratio	Degree of Stenosis
35	0.113	Very Severe
45	0.227	Severe
55	0.395	Moderate
75	0.808	None

This study did not include a comparison of simulation results to clinical data. The aim of this study was not to simulate a realistic case, but to compare the PINN method to the FVM method. As a simplified geometry was used, the results were not expected to be an accurate representation of real transvalvular blood flow.

3. Physics-Informed Neural Networks

With the PINN method, first proposed by Raissi et al. [2], we aimed to solve the PDE problem in a fundamentally different way than with the FVM.

Traditionally, the PDE problem has been solved by discretizing the PDE, and solving the resulting system of equations.

The PINN method aims to solve the PDE problem by constraining and training a neural network to satisfy any symmetries, conservation principles, invariances or boundary conditions that are known to hold for the system.

After training, what we are left with is a neural network that can infer the solution of the PDE at any point in the domain, as shown in Figure 2. An attractive property of this solution is that it is continuous in the domain, and that inference is very fast once the model has been trained: this makes PINNs valuable surrogate modeling tools.

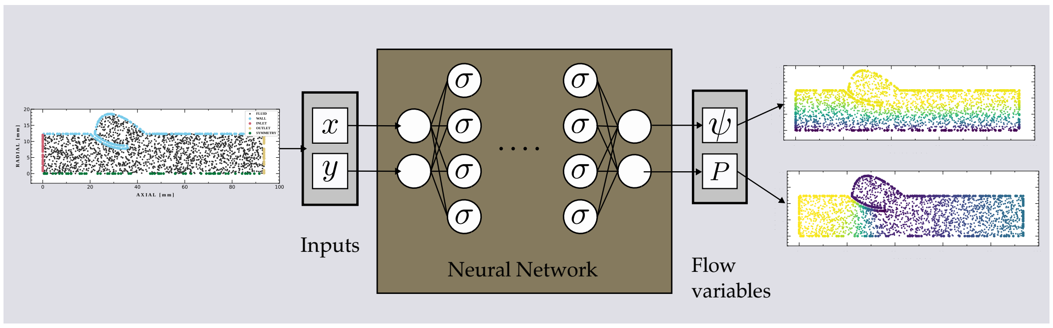


Figure 2. Neural network as a general function approximator for approximating the solution of a partial differential equation (PDE). Several input coordinates are passed to the network, and the output is the predicted flow field. Here, the flow field is expressed as the scalar stream function ψ and scalar pressure field p.

3.1. Problem Formulation

In this paper, the stream function pressure formulation was used as the preferred formulation of the Navier–Stokes equations. We defined the scalar stream function ψ, such that

$$u = \frac{\partial \psi}{\partial y} \quad \text{and} \quad v = -\frac{\partial \psi}{\partial x} \tag{5}$$

was true. Equations (5) were substituted into the Navier–Stokes Equations (1) and (2), to obtain the stream function pressure formulation.

This formulation was preferred, as it automatically satisfied mass conservation: this came at the computational cost of having to calculate third-order derivatives in diffusion terms.

For this case study, velocity results differed by several orders of magnitude from pressure results. This caused an imbalance in the loss function, which had larger terms associated with pressure, and smaller terms associated with the velocities. This imbalance was severely detrimental to the optimization process, and could cause the model training to fail outright. To remedy this, the governing equations were non-dimensionalized. The non-dimensional variables were defined as:

$$x^* = \frac{x}{l_c} \quad , \quad y^* = \frac{y}{l_c} \quad , \tag{6}$$

$$u^* = \frac{u}{u_c} \quad , \quad v^* = \frac{v}{u_c} \quad , \quad p^* = \frac{p}{\rho u_c^2} \tag{7}$$

where the characteristic length l_c was the diameter of the base, and the characteristic velocity u_c was the inlet velocity. The non-dimensionalized stream function can be written as:

$$\psi^* = \frac{\psi}{u_c l_c} \tag{8}$$

3.2. Mathematical Description of the PINN

The PINN method uses a neural network as a flexible mathematical function that can be used to represent an approximate solution to a given PDE problem; however, the PINN method differs from typical regression methods, in that the neural network is not trained in a supervised manner, to approximate a function. Instead, we treat the network as a function that must satisfy all the invariances, symmetries and boundary conditions that are known to hold for the physical system. If the invariances, symmetries and boundary conditions are indeed consistent, the solution is unique, and the loss function will have a minimum that corresponds to the solution of the PDE problem, which can be found using a nonlinear optimizer. An overview of the PINN method is shown in Figure 3.

In the subsequent section, we will elaborate on the derivation of the loss function, which constitutes a crucial distinction between Physics-Informed Neural Networks and conventional neural networks. By encoding the known physics of the system, the PINN loss function facilitates training without labeled data.

As shown in Figure 3, the PINN architecture contains a fully connected neural network (NN) that is attached to a training section. The NN is expressed mathematically as a nonlinear operator : $\mathcal{N}$

$$\zeta(\mathbf{x}) = \mathcal{N}(\mathbf{x}, \mathbf{w}) \tag{9}$$

where $\mathbf{x} \in \mathbb{R}^{n\times d}$ is a matrix of n samples in Cartesian coordinates, each with dimensionality $d = 2$; $\mathbf{w} \in \mathbb{R}^{q}$ is the vector of q model parameters; and $\zeta \in \mathbb{R}^{n\times m}$ is a matrix of $m = 2$ predicted flow variables.

$$\mathbf{x} = \begin{bmatrix} x^* \\ y^* \end{bmatrix} \qquad\qquad \zeta = \begin{bmatrix} \psi^* \\ p^* \end{bmatrix} \tag{10}$$

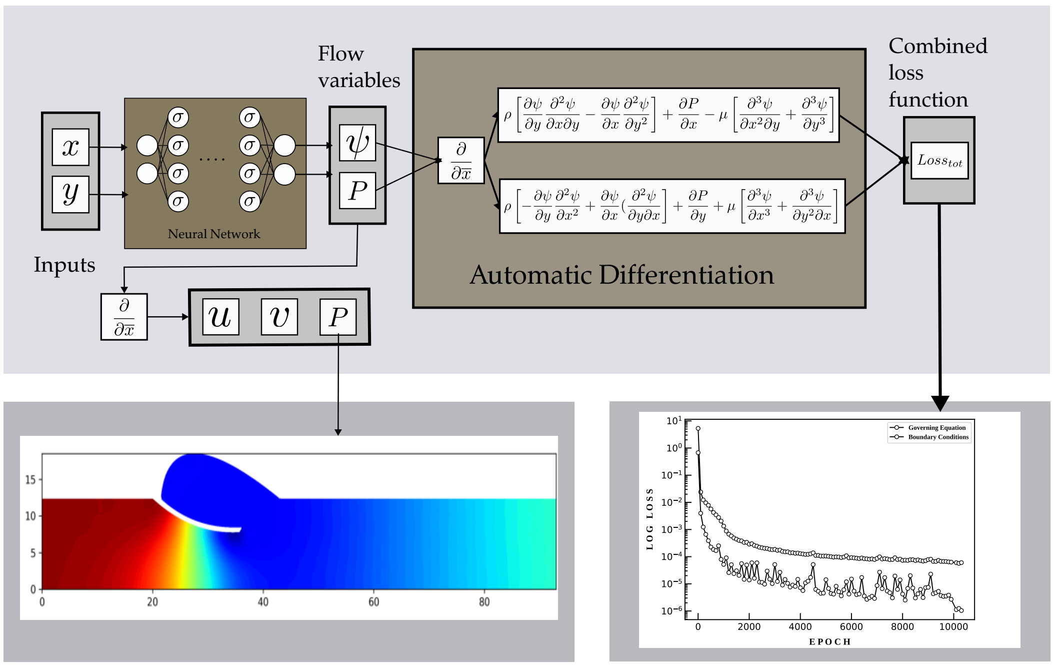

Flow variables
Combined loss function
x
y
Inputs
Neural Network
ψ
P
$\frac{\partial}{\partial x}$
$\rho\left[\frac{\partial\psi}{\partial y}\frac{\partial^2\psi}{\partial x\partial y} - \frac{\partial\psi}{\partial x}\frac{\partial^2\psi}{\partial y^2}\right] + \frac{\partial P}{\partial x} - \mu\left[\frac{\partial^3\psi}{\partial x^2\partial y} + \frac{\partial^3\psi}{\partial y^3}\right]$
$\rho\left[-\frac{\partial\psi}{\partial y}\frac{\partial^2\psi}{\partial x^2} + \frac{\partial\psi}{\partial x}(\frac{\partial^2\psi}{\partial y\partial x}\right] + \frac{\partial P}{\partial y} + \mu\left[\frac{\partial^3\psi}{\partial x^3} + \frac{\partial^3\psi}{\partial y^2\partial x}\right]$
$Loss_{tot}$
Automatic Differentiation
u
v
P
Governing Equation
Boundary Conditions
LOG LOSS
EPOCH

Figure 3. Simplified overview of the physics informed neural network (PINN) architecture.

To train the NN, a composite loss function, $\mathcal{L}$, is assembled and evaluated at several pseudo-randomly spaced locations in the domain during each training iteration. These sample locations are called *collocation points*, and are shown in Figure 4.

Collocation points are generated using the FreeCAD software package [25], which offers great flexibility via the python API, is free and is open source. Points are placed in the domain, and are pseudo-randomly distributed according to a Sobol sequence. The Sobol sequence offers a good compromise between uniformity and low discrepancy in the placement of collocation points [26].

The loss function is composed of multiple terms, with each term corresponding to a conservation law, boundary condition or other constraints, enforced using the penalty method, which penalizes deviation of the network output from the constraints in a soft manner (soft constraints are never perfectly satisfied as opposed to hard constraints that are. As the loss function is minimized, soft constraints get closer to being satisfied, but a residual always remains). The loss function is written as:

$$\mathcal{L}(\mathbf{w}) = \lambda \frac{1}{n} \sum_{i=1}^{n} \left(f_j(\mathbf{x_i})\right)^2 \tag{11}$$

where $\lambda \in \mathbb{R}^j$ is a vector of j manually tuned weights, f_j is the j'th residual function, and n_j is the number of collocation points where f_j is evaluated.

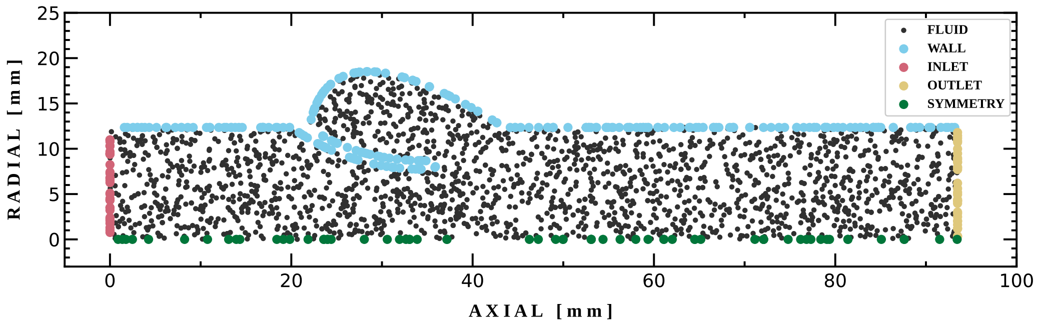


Figure 4. Collocation points where the PINN loss function is evaluated. Boundary condition residuals are only evaluated at points placed on the corresponding boundary.

The residual functions to be minimized are the governing equations: namely, the momentum conservation equations and the boundary conditions. Mathematically, they are expressed as:

Governing Equations: Momentum Conservation:

$$f_{mom;x}(\mathbf{x}) = \left[\frac{\partial \psi^*}{\partial y^*}(\mathbf{x})\frac{\partial^2 \psi^*}{\partial x^* \partial y^*}(\mathbf{x}) - \frac{\partial \psi^*}{\partial x^*}(\mathbf{x})\frac{\partial^2 \psi^*}{\partial y^{*2}}(\mathbf{x})\right] + \frac{\partial p^*}{\partial x^*}(\mathbf{x}) - \frac{(\mu + \mu_t(\mathbf{x}))}{\rho u_c l_c}\left[\frac{\partial^3 \psi^*}{\partial x^{*2}\partial y^*}(\mathbf{x}) + \frac{\partial^3 \psi^*}{\partial y^{*3}}(\mathbf{x})\right] \approx 0 \qquad \mathbf{x} \in \Gamma_{fluid} \tag{12}$$

$$f_{mom;y}(\mathbf{x}) = \left[-\frac{\partial \psi^*}{\partial y^*}(\mathbf{x})\frac{\partial^2 \psi^*}{\partial x^{*2}}(\mathbf{x}) + \frac{\partial \psi^*}{\partial x^*}(\mathbf{x})\frac{\partial^2 \psi^*}{\partial y^* \partial x^*}(\mathbf{x})\right] + \frac{\partial p^*}{\partial y^*}(\mathbf{x}) + \frac{(\mu + \mu_t(\mathbf{x}))}{\rho u_c l_c}\left[\frac{\partial^3 \psi^*}{\partial x^{*3}}(\mathbf{x}) + \frac{\partial^3 \psi^*}{\partial y^{*2}\partial x^*}(\mathbf{x})\right] \approx 0 \qquad \mathbf{x} \in \Gamma_{fluid} \tag{13}$$

Boundary Conditions: no slip wall:

$$f_{wall}(\mathbf{x}) = \overline{u^*}(\mathbf{x}) \approx 0 \qquad \mathbf{x} \in \Gamma_{wall} \tag{14}$$

Boundary Conditions: specified velocity inlet:

$$f_{inlet}(\mathbf{x}) = \overline{u^*}(\mathbf{x}) - \overline{u^*}_{inlet}(\mathbf{x}) \approx 0 \qquad \mathbf{x} \in \Gamma_{inlet} \tag{15}$$

Boundary Conditions: reference pressure outlet:

$$f_{outlet}(\mathbf{x}) = p^*(\mathbf{x}) - p^*_{outlet}(\mathbf{x}) \approx 0 \qquad \mathbf{x} \in \Gamma_{outlet} \tag{16}$$

Boundary Conditions: symmetry plane:

$$f_{sym;1}(\mathbf{x}) = \frac{\partial p^*}{\partial \overline{n}}(\mathbf{x}) \approx 0 \qquad \mathbf{x} \in \Gamma_{symmetry} \tag{17}$$

$$f_{sym;2}(\mathbf{x}) = \frac{\partial \overline{u^*}}{\partial \overline{n}}(\mathbf{x}) \approx 0 \qquad \mathbf{x} \in \Gamma_{symmetry} \tag{18}$$

$$f_{sym;3}(\mathbf{x}) = \overline{u^*}(\mathbf{x}) \cdot \overline{n} \approx 0 \qquad \mathbf{x} \in \Gamma_{symmetry} \tag{19}$$

where $\overline{u^*}$ is the dimensionless velocity vector, $\overline{u^*}_{inlet}$ is the dimensionless prescribed inlet velocity, p^* is the dimensionless pressure, p^*_{outlet} is the dimensionless reference outlet pressure, $\overline{n}$ is the boundary normal vector, and Γ_{wall}, Γ_{inlet}, Γ_{outlet}, and $\Gamma_{symmetry}$ are the wall, inlet, outlet and symmetry boundary domains, respectively.

To evaluate the loss function, derivatives of the network outputs are required, as can be seen by the derivative terms in the equations above. These derivatives are computed using automatic differentiation, which allows the computation of the gradients of $\mathcal{N}$ with respect to $\mathbf{x}$, as $\mathcal{N}$ is a computational graph. Interested readers are directed to

Güene et al. [27] for a detailed explanation of automatic differentiation, and how it differs from numerical differentiation.

3.3. Training

Once the loss function $\mathcal{L}$ is defined, a solution can be computed, by applying a stochastic gradient descent algorithm, such as ADAM [28], to find the values of the network weights that minimize $\mathcal{L}$.

At each iteration of the training algorithm, a mini-batch of collocation points is sampled from the domain, and the loss function is evaluated. ADAM is a gradient-based optimization algorithm, which means that derivatives of the loss function must also be computed: this is done efficiently, using automatic differentiation. The network weights are updated iteratively, for a fixed number of iterations.

Mini-batch subsampling is done, to enable the use of a larger set of collocation points, while still being able to take advantage of a Graphical Processing Unit (GPU) (an Nvidia GTX 1050Ti GPU was used for training), which has a limited amount of memory for training. Full batch training is done for the boundary conditions.

Convergence to a global minimum of the loss function is not guaranteed, and the algorithm will most likely converge to a local minimum. This is an inherent limitation of machine learning methods that rely on the formulation of a non-convex loss function and nonlinear optimizer. The result is some variability in the convergence speed and the final solution of each training run. In the authors' experience, however, this variability is not significant enough to impact the practical usability of the method.

While the PINN training procedure does have inherent flaws, such as convergence to local minima, it also has inherent advantages, such as allowing the flexibility to incorporate measurement data into the solution, and the ability to solve parameterized problems in parallel.

3.4. Pinn Turbulence Model Implementation

The mixing length model, proposed by Hennning et al. for use in Nvidia Symnet [16], was chosen to compute the turbulent viscosity. To the authors' knowledge, there are no publications that have used a more complex turbulence model with PINNs trained without labeled training data.

The turbulent viscosity was computed as explained in Section 2, with a mixing length [16] defined as:

$$l_m(\mathbf{x}) = min(0.419d(\mathbf{x}), 0.09d_{max}) \tag{20}$$

where wall distance d and maximum wall distance d_{max} had to be efficiently computed. Commercial CFD solvers compute the normal wall distance efficiently by solving specially formulated PDEs, such as the Eikonal equations [29], in the same computational domain. We followed a similar approach, by solving a Poisson-type equation, as proposed by Spalding [30]:

$$\nabla^2 L = C \tag{21}$$

$$d = \pm\sqrt{\sum_{i=1,2}\left(\frac{\partial L}{\partial x_i}\right)^2} + \sqrt{\sum_{i=1,2}\left(\frac{\partial L}{\partial x_i}\right)^2 + 2L} \tag{22}$$

with boundary conditions

$$\begin{aligned} L &= 0 \quad \text{on} \quad \Gamma_{wall} \\ \frac{\partial L}{\partial \overline{n}} &= 0 \quad \text{on} \quad \Gamma_{inlet}, \Gamma_{outlet}, \Gamma_{symmetry} \end{aligned}$$

where x_i was the ith spatial dimension, $C = -1$, d was the wall distance, L was the level set function, and $\overline{n}$ was the boundary normal vector.

A separate PINN was used to solve Equation (21), and to recover the level set function L. As the PINN was differentiable by using automatic differentiation, the derivative terms $\frac{\partial L}{\partial x_i}$ and wall distance d could be efficiently computed using Equation (22). The level set function and wall distance are shown in Figure 5. Once the wall distance PINN was trained, it could be efficiently used to infer the wall distance at any point in the domain. As the geometry was static for each simulation case, the wall distance PINN only needed to be trained and evaluated once.

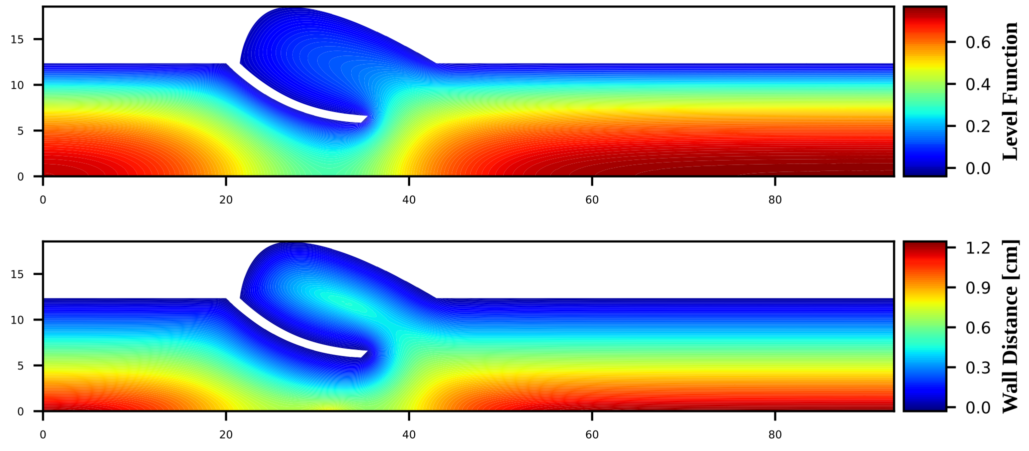


Figure 5. Level function and associated wall distance, computed using a PINN, trained to solve the Eikonal equations.

This is also a powerful approach to use when the geometry is dynamic. For example, in the case of a variable valve angle, the wall distance PINN can be trained to compute the wall distance for a continuous range of valve angles. Once the PINN is trained, it can rapidly infer the wall distance at any location, for any valve angle in that range.

The Poisson method yields a slightly less accurate wall distance, compared to the Eikonal method: this difference is, however, mostly in the region far from the wall, where it has little or no impact on the mixing length. The authors found that the Poisson method was sufficient for their purposes, and was easier to implement.

3.5. Gradient Enhanced PINN

To improve solution accuracy, we implemented a gradient-enhanced loss function, as proposed by Yu et al. [19]. This method was based on the observation that if we enforced a zero loss function across the entire domain, then the gradient of the loss function w.r.t the spatial dimensions would also need to be zero. We thus utilized the availability of gradient information, to add a loss term that penalized a non-zero gradient of the momentum residual functions, as shown in Equation (23):

$$\mathcal{L}_g(\mathbf{w}) = \lambda_g \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\partial f_j}{\partial x_d}(\mathbf{x_i}) \right)^2, \quad \text{where} \quad j \in \{\text{momentum x}, \text{momentum y}\}. \tag{23}$$

The final loss function thus contained two terms that penalized deviation from momentum conservation in the x and y directions, and two additional terms that penalized the non-zero gradients of the first two terms in the x and y directions.

3.6. Fourier Layer

PINNs exhibit stiff training dynamics, which limit their practical use as a PDE solver. Many papers have focused on alternative network architectures, to enable faster training and more accurate solutions. In this paper, the Fourier layer, as proposed by Wang et al. [20], was implemented. Wang et al. [20] showed that PINNs are biased towards training low spatial frequencies in the solution faster than high frequencies. The Fourier layer aims to alleviate this spectral bias, by mapping the input coordinates to randomized sinusoidal features, before entering the network. The notation of Wang et al. [20], and the formulation of Tancik et al. [31],

$$\gamma(\mathbf{x}) = \begin{bmatrix} \sin(\mathbf{Bx}) \\ \cos(\mathbf{Bx}) \end{bmatrix} \tag{24}$$

where $\mathbf{B} \in \mathbb{R}^{n \times d}$ is a random matrix, were used, with entries sampled from a Gaussian distribution with standard deviation σ. The standard deviation was a tunable parameter that changed the frequency of the spectral bias.

3.7. Skip Connections

A well-known phenomenon that complicates the training of deep neural networks is that of disappearing gradients. As the layer count increases, the gradients of the loss function, w.r.t model parameters, get smaller and smaller. This is also the case for gradients of the predicted flow variables, w.r.t input coordinates, which are used in the PINN loss function [7]. The decreasing strength of spatial derivative information leads to longer training times and less accurate solutions.

Shi, Liu and Huo [7] proposed a solution to this problem, by re-introducing spatial coordinate information into each layer of the network, as shown in Figure 6.

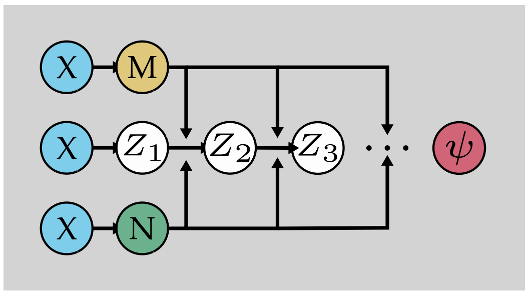


Figure 6. Fully connected deep neural network with skip connections. Spatial information is re-introduced at each layer, to strengthen the spatial gradients in the loss function.

The forward propagation rule for the network at the *i*th layer is written as:

$$\begin{aligned} Z^i &= \sigma(W^i Z^{i-1} + b^i) \cdot M + N \\ M &= \sigma(W_M X + b_M) \\ N &= \sigma(W_N X + b_N) \end{aligned} \tag{25}$$

where Z^i is the output of the *i*th layer with weights W^i and biases b^i, X is the input coordinates and σ is the activation function. W_M and W_N are the weights of the M and N layers, respectively, while b_M and b_N are the biases of the M and N layers, respectively.

This method synergizes well with the gradient-enhanced PINN (gPINN) [19]. The additional gPINN loss function term, along with the $\psi - p$ formulation, results in fourth-order spatial gradients in the loss function: this makes this formulation especially sensitive to weak spatial gradients during training.

4. Fvm Reference Solution

As a reference solution, the problem was solved using the Ansys® 2023 R1 software suite. Mesh generation was done using Ansys Mechanical® 2023 R1 [32], as shown in Figure 7: this resulted in a mixed quadrilateral and triangular element mesh with around 135,000 linear elements. Mesh inflation was used for 10 elements around the wall, to ensure good-quality elements around the wall, which were also aligned with the boundary. The meshing operation was performed using an intel i7 7700HQ processor.

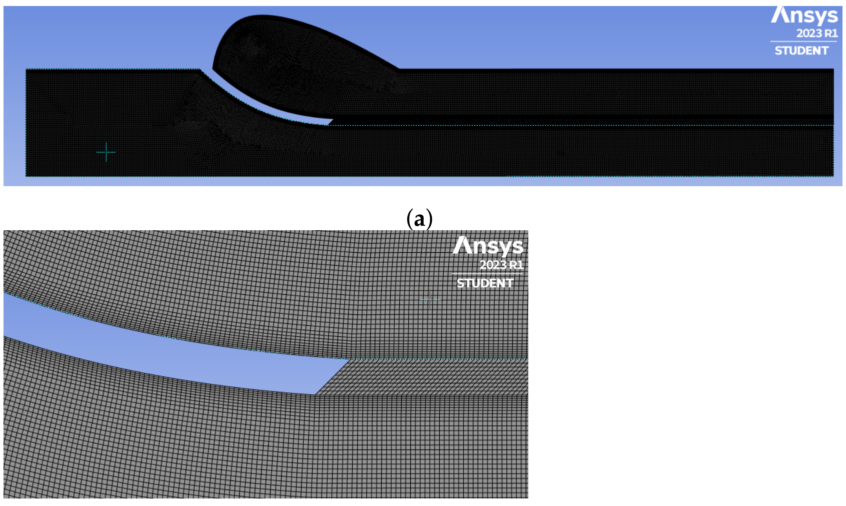


Figure 7. (**a**) Finite volume method (FVM) mesh for a valve angle of 45° with a cell count of approximately 135,000 (**b**) close-up view of the mesh near the valve leaflet tip.

The mesh was imported into Ansys Fluent® 2023 R1 [33], to solve both the laminar and the turbulent cases. The same hardware was used for the solver as was used for the meshing operation. Boundary conditions were imposed, as shown in Figure 8. A specified velocity profile inlet condition, reference pressure of 0 Pa outlet condition, no-slip wall and symmetry plane were imposed.

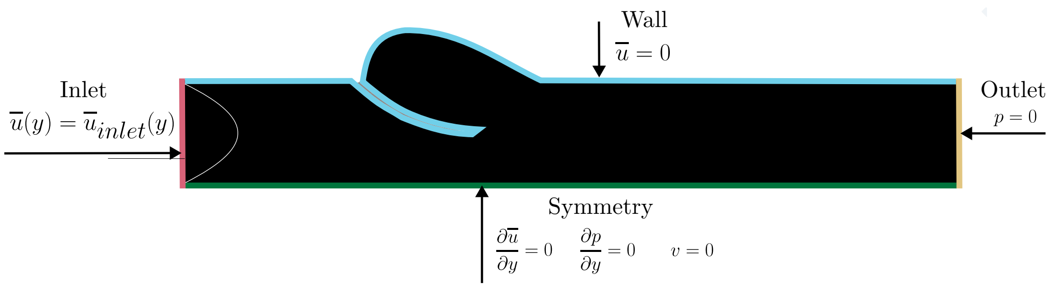

Wall
$\overline{u} = 0$
Inlet
$\overline{u}(y) = \overline{u}_{inlet}(y)$
Outlet
$p = 0$
Symmetry
$\frac{\partial \overline{u}}{\partial y} = 0$ $\frac{\partial p}{\partial y} = 0$ $v = 0$

Figure 8. Boundary conditions imposed for the laminar case and turbulent case with a mixing length turbulence model.

4.1. Turbulence Modeling

For the turbulent solution, a mixing length turbulence model [18] was used. As mentioned before, this algebraic turbulence model does not have a transport equation for any turbulence properties, and thus requires no additional boundary conditions to be imposed beyond the laminar case.

The turbulent viscosity was calculated using a wall distance l_m and shear rate G, as shown previously in Equation (3).

The official ANSYS Fluent® user guide [34] does not disclose the exact method used to calculate the mixing length, l_m; however, the authors determined empirically that the mixing length was calculated using the same equation as the one used for the PINN formulation shown in Equation (3).

4.2. Equation Discretization and Solution

Similar discretization schemes were chosen for the laminar and turbulent solutions. The gradient terms were computed using a least-squares-based method. The face pressures were computed using a second-order (or central differencing) scheme. The divergence terms in the momentum equations were computed using a second-order upwind scheme [35].

The laminar solution was computed using a coupled solver, and the turbulent solution was computed using a SIMPLE solver [35].

The solution process was initialized by first computing the inviscid solution, and using the resulting velocity field as an initial guess for the viscous solution. The solution was then iterated until convergence was reached.

The convergence criteria for the viscous solution was a maximum absolute change in the area-averaged inlet pressure of 10^{-2} Pa: it was observed that if this criterion was met, the momentum and continuity residuals were also below 10^{-4}.

5. Results

The metrics used to evaluate the accuracy of the PINN solution are expressed in Equations (26)–(29).

The first compares the peak jet velocities on the symmetry plane; the difference was normalized by the peak jet velocity of the FVM solution:

$$e_1 = \frac{(u_{max})_{PINN} - (u_{max})_{FVM}}{(u_{max})_{FVM}} \cdot 100 \qquad u_{max} = \max_i(||\mathbf{u}^i_{symmetry}||_2) \tag{26}$$

The second compares the transvalvular pressure drop: this was computed as the area-averaged pressure difference between the inlet and outlet; the difference was normalized by the transvalvular pressure drop of the FVM solution:

$$e_2 = \frac{(p_{drop})_{PINN} - (p_{drop})_{FVM}}{(p_{drop})_{FVM}} \cdot 100 \qquad p_{drop} = \frac{1}{NA}\sum_{i=0}^{N}(A^i p^i_{inlet}) - p_{outlet} \tag{27}$$

The third is the maximum absolute error of the velocity magnitude, normalized by the maximum velocity magnitude of the FVM solution:

$$e_3 = \frac{||\mathbf{u}^i_{PINN} - \mathbf{u}^i_{FVM}||^2_\infty}{(\mathbf{u}_{max})_{FVM}} \cdot 100 \tag{28}$$

The fourth is the maximum absolute error of the pressure field, normalized by the transvalvular pressure drop of the FVM solution:

$$e_4 = \max_i\left(\frac{|p^i_{PINN} - p^i_{FVM}|}{(p_{drop})_{FVM}}\right) \cdot 100 \tag{29}$$

The solutions are also plotted along two lines intersecting the geometry, as shown in Figure 9.

The pressure is plotted on the first line, which runs along the symmetry plane: this illustrates the pressure change in the flow direction, as shown in Figures 11 and 13, in black.

The second line is perpendicular to the flow direction, just downstream of the valve: this is used to compare the axial velocity profile in the jet, as shown in Figures 11 and 13, in red.

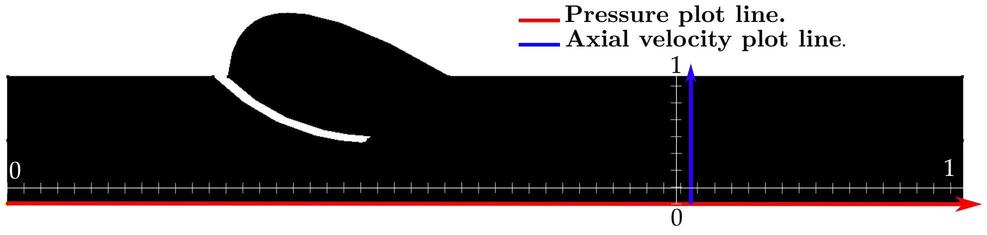


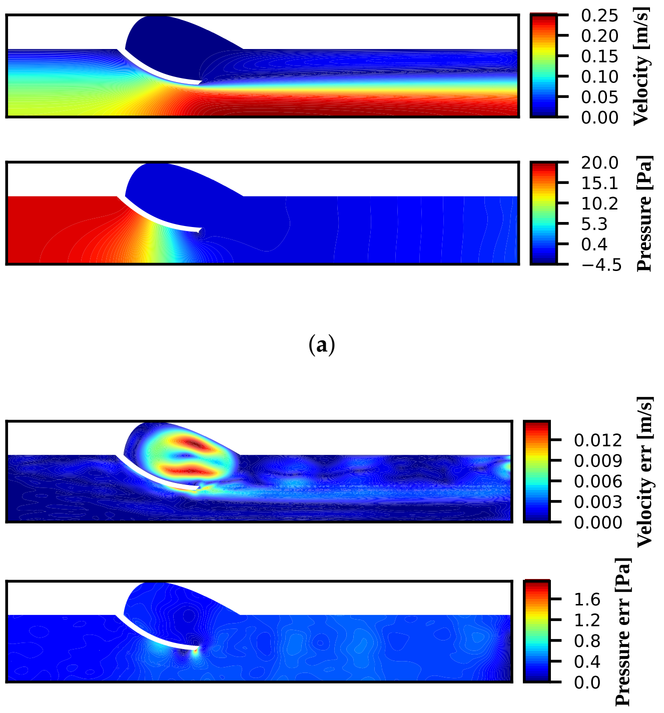
Figure 9. Intersect plot: the pressure field is plotted along the red line, and the axial velocity field is plotted along the blue line.

5.1. Laminar Flow

Flow in the laminar regime was simulated for a range of valve angles between 75° and 35°. A parabolic inlet velocity profile was used, with a volumetric flow rate of 46 mL/s: at this volume flow rate, the Reynolds number, based on the base diameter, was $Re = 651$.

A four-layer neural network was used, with 20 neurons per layer, except for the Fourier encoding layer, which had 40 neurons. The network was trained for 5000 epochs, using the ADAM optimizer with a mini-batch size of 1024 and a collocation number count of 8727 for momentum conservation and 765 on all the boundaries combined. A learning rate of 1×10^{-3} was used (interested readers are referred to the classic text by Rao [36] for more background information on general mathematical optimization and the ADAM optimizer.).

The pressure and velocity fields for a valve angle of 45° are shown in Figure 10 for the PINN and FVM, as well as the absolute difference between the two. The figures for the other valve angles are shown in Appendix A. The evaluation metrics are shown in Tables 3–5.



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

(**c**)

Figure 10. Laminar flow results for a valve angle of 45°: (**a**) FVM pressure and velocity fields; (**b**) PINN pressure and velocity fields; (**c**) absolute difference between the fields predicted by the PINN and FVM.

Table 3. Simulation Cases: FVM.

Valve Angle [deg]	Peak Jet Velocity [m/s]	Pressure Drop [Pa]	Simulation Time [s]
35	0.3386	47.47	36.452
45	0.2465	18.17	38.782
55	0.195	5.621	25.357
75	0.154	0.8754	19.637

Table 4. Simulation Cases: PINN.

Valve Angle [deg]	Peak Jet Velocity [m/s]	Pressure Drop [Pa]	Momentum Residual	Simulation Time [s]
35	0.3377	47.45	4.69×10^{-4}	5010
45	0.2475	18.4	3.11×10^{-4}	4515
55	0.1956	5.843	1.13×10^{-4}	5172
75	0.1549	0.5956	1.41×10^{-4}	5843

Table 5. Simulation Cases: error metrics.

Valve Angle [deg]	e1 [%]	e2 [%]	e3 [%]	e4 [%]
35	0.3757	0.06139	6.691	9.29
45	−0.3016	−1.295	5.736	10.59
55	−0.2521	−3.952	2.612	10.18
75	−0.6278	31.96	7.227	78.57

The results show that the PINN solution was in good agreement with the FVM solution. Both the pressure and the velocity fields were closely matched. It can be seen in Figure 10c that the PINN solution showed a circulation action in the sinus that was not present in the FVM solution. The physicality of this circulation action was difficult to verify without experimental data; however, we intuitively expected some circulating flow in the sinus, due to the shearing action of the jet: if this intuition was correct, then the PINN solution was representative of the actual flow.

Tables 3 and 5 show that both the PINN and FVM predicted an increase in the peak jet velocity and pressure drop as the flow restriction was increased. Notably, the PINN simulation time was significantly longer than that of the FVM: this was to be expected with the current state of the art in PINN research for this type of problem. Both the PINN and FVM showed slower convergence with increasing flow restriction.

Table 5 shows that the velocity field was in much better agreement between the PINN and FVM than the pressure field. The peak jet velocity errors (e_1) were all below 1%, whereas three of the pressure drop errors (e_2) were above 1%. The pressure drop error for a valve angle of 35° was substantially lower than the other errors, for both the laminar and turbulent solutions. We consider this result to be an outlier, especially as the maximum absolute error of the pressure field (e_4) was not that much smaller than the e_4 errors of the other valve angles. The pressure drop error (e_2) and maximum absolute error of the pressure field (e_4), for a valve angle of 75°, were much larger than the other errors: this was because there was very little flow restriction in this case, which resulted in a very small pressure drop. The PINN training process is biased towards resolving features that have a larger impact on the numerical values in the loss function. When there is little flow restriction, the pressure gradient term in the momentum residual function is small, and the PINN training process resolves the pressure field more slowly.

Overall, the velocity field results are in better agreement than the pressure field results: this was somewhat expected, as although the $\psi - p$ formulation used for the PINN ensured perfect mass conservation, there remained a residual in the momentum conservation equation. Momentum conservation ensured the dynamic equilibrium of forces in the system, which the pressure and velocity fields contributed to; however, it seems that the pressure field was more sensitive to errors in the momentum conservation equation than the velocity field.

The section plots shown in Figure 11 illustrate good agreement between the predicted pressure and velocity fields by the FVM and PINN methods. The velocity profile predicted by the PINN, shown in a black dashed line, is difficult to distinguish from the profile predicted by the FVM, shown in a solid black line, due to them being so closely matched.

Both methods predict an s-shaped velocity profile, with some reversed flow near the wall of the vessel.

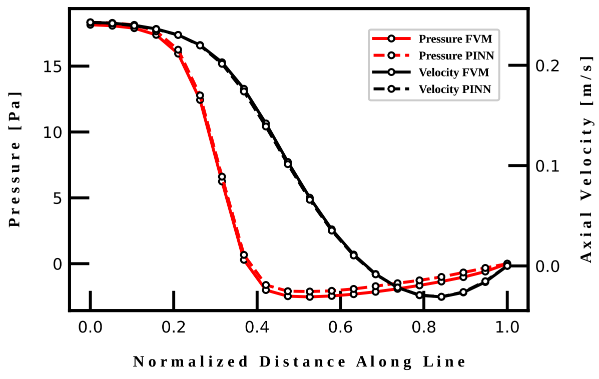


Figure 11. Section plot for the laminar solution at a valve angle of 45°; the pressure profile along the symmetry plane is in red, and the axial velocity profile downstream of the valve is in black.

The pressure profiles, shown in red, are less closely matched, but still difficult to distinguish. Both methods predict a sharp pressure drop at the valve, followed by a small amount of pressure recovery downstream of the valve, as the steady velocity profile starts to develop and the maximum jet velocity decreases.

5.2. Turbulent Flow

Flow in the turbulent regime was also simulated for a range of valve angles between 75° and 35°. A parabolic inlet velocity profile was used, at a volumetric flow rate of 400 mL/s: this flow rate corresponded to the peak systolic flow rate of a healthy adult male [37]. At this volume flow rate, the Reynolds number based on the base diameter was $Re = 5438$, which was below the critical Reynolds number for a tube. The flow did, however, transition to turbulence, due to the presence of the valve.

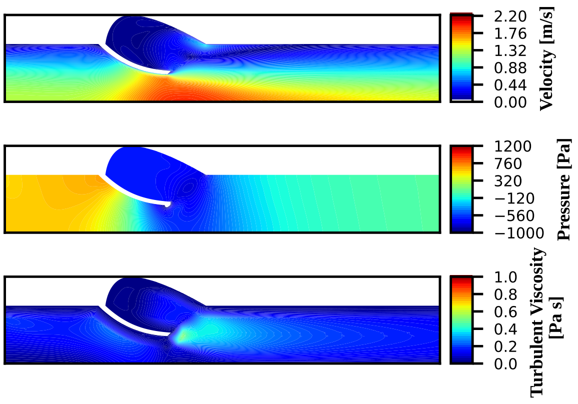
Similarly to the laminar model setup, a four-layer neural network was used, with 20 neurons in each layer, except for the Fourier encoding layer, which had 40 neurons. The network was trained for 10,000 epochs, using the ADAM optimizer with a mini-batch size of 1024 and a learning rate of 1×10^{-3}. The momentum conservation loss function and boundary loss functions were evaluated at 8727 and 765 collocation points, respectively.

The pressure, velocity and turbulent viscosity fields for a valve angle of 45° are shown in Figure 12 for the PINN and FVM, as well as the absolute difference between the two. Figures for the other valve angles are shown in Appendix A. The results are also tabulated in Tables 6–8.

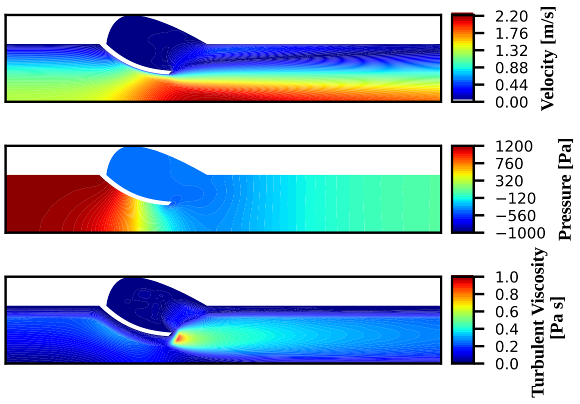
The results show that the addition of a mixing length turbulence model to the PINN enabled the simulation of steady turbulent flows; however, the PINN solution did not match the FVM solution as well as it matched the laminar case: here, the FVM solution predicted more circulation in the sinus area than did the PINN, as shown in Figure 12c. Both methods predicted a region of high turbulent viscosity near the tip of the valve leaflet, but the PINN predicted a higher viscosity than the FVM.

Looking at Tables 6 and 7, once again both models predicted an increase in the peak jet velocity and pressure drop, as the valve angle decreased. Simulation times were longer for both methods when compared to the laminar case, and the PINN simulations converged more slowly than did the FVM simulations.

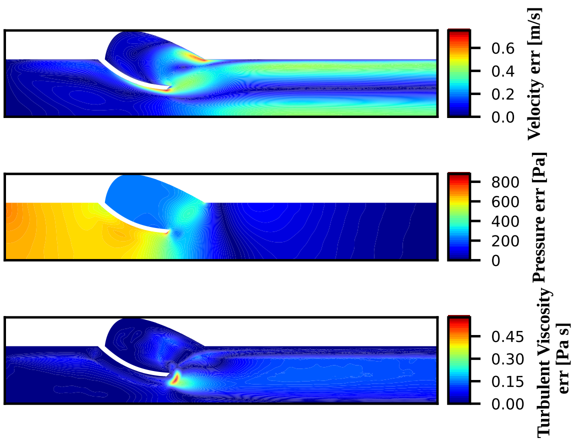
The PINN simulation, at a valve angle of 35°, terminated with a relatively large momentum residual, compared to the other laminar and turbulent simulations: it seems that the steeper gradients present in this case reduced the training performance of the neural network.



(**a**)



(**b**)



(**c**)

Figure 12. Turbulent flow results for a valve angle of 45°: (**a**) FVM pressure, velocity and turbulent viscosity fields; (**b**) PINN pressure, velocity and turbulent viscosity fields; (**c**) absolute difference between the fields predicted by the PINN and FVM.

Table 6. Simulation Cases: FVM.

Valve Angle [deg]	Peak Jet Velocity [m/s]	Pressure Drop [Pa]	Simulation Time [s]
35	2.601	1609.6	642
45	1.906	518.2	613
55	1.531	132.1	299.5
75	1.252	−39.07	278

A strange result to note in Table 6 is the negative pressure drop at a valve angle of 75°: this was a result of the parabolic inlet velocity profile that was used in place of a fully developed turbulent velocity profile. As a turbulent velocity profile develops, the average velocity remains constant, but the peak velocity decreases: this results in a decrease in the kinetic energy of the mean flow, and an increase in the static pressure, due to Bernoulli's principle. As the pressure drop due to flow restriction was so small in the 75° case, the pressure gain due to the developing velocity profile was enough to cause a negative pressure drop.

Table 8 shows higher errors across the board for the turbulent case, compared to the laminar case. The peak jet velocity error (e_1), pressure drop error (e_2), normalized maximum velocity error (e_3) and normalized maximum pressure error (e_4) had a median increase in error, by a factor of 20, 28, 5.5 and 18 higher, respectively.

Table 7. Simulation Cases: PINN.

Valve Angle [deg]	Peak Jet Velocity [m/s]	Pressure Drop [Pa]	Momentum Residual	Simulation Time [s]
35	2.744	2557.2	1.74×10^{-3}	10,002
45	2.103	1173.6	3.31×10^{-4}	12,321
55	1.667	502.1	1.36×10^{-4}	10,166
75	1.318	150.0	5.41×10^{-5}	9234

Once again, the PINN was able to infer more accurate velocities than pressures. The median difference in error between the turbulent peak velocity error e_1 and the turbulent pressure drop error e_2 was 20 times. The median difference in error between the turbulent normalized maximum velocity error e_3 and the turbulent normalized maximum pressure error e_4 was 3.7 times.

Median values are quoted instead of mean values because the results were skewed by the large error values for the 75° case. The 75° turbulent case had a larger error for the same reason as the laminar case: the small amount of flow restriction made the pressure term in the momentum equation small, which caused the PINN to resolve the pressure field much more slowly than the velocity field.

Table 8. Simulation Cases: error metrics.

Valve Angle [deg]	e1 [%]	e2 [%]	e3 [%]	e4 [%]
35	−5.529	−58.87	41.82	112.9
45	−10.34	−126.5	39.04	157.6
55	−8.93	−280.2	32.86	241.7
75	−5.312	484.0	21.58	−4168

Figure 13 provides another perspective on the differences between the turbulent FVM and PINN solutions, along the section lines as defined in Figure 9 for a valve angle of 45°.

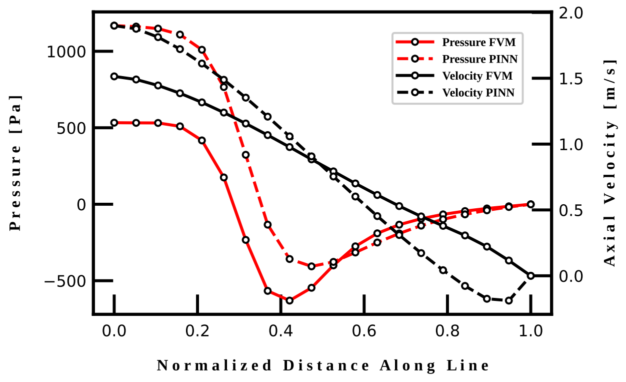


Figure 13. Section plot for the turbulent solution at a valve angle of 45°; the pressure profile along the symmetry plane is in red, and the axial velocity profile downstream of the valve is in black.

Based on the velocity profiles, shown in black, it can be seen that the PINN predicted less diffusion in the jet than the FVM. The PINN predicted some backflow near the wall, whereas the FVM did not.

The pressure curves, plotted in red, show a sharp drop in pressure at the valve, with a much faster recovery downstream than the laminar case (Figure 11): this is seen in the

PINN and FVM solutions, but the FVM solution shows a faster pressure recovery than the PINN.

6. Conclusions

In this paper, we have demonstrated the application of Physics-Informed Neural Networks as a new deep-learning-based method for solving the incompressible Reynolds-averaged Navier–Stokes equations in a biomedical context. Transvalvular blood flow was simulated in the laminar and turbulent regimes, with a varying degree of simulated valve stenosis. Various methods of improving the training performance and prediction accuracy of PINNs, from the literature, were implemented, including the Gradient-Enhanced PINN [19], the Fourier encoding layer [20] and the re-introduction of spatial information using skip connections [7]. A mixing length turbulence closure model was implemented for the turbulent case. The mixing length was calculated by solving a Poisson-type equation with a secondary PINN to calculate the normal wall distance.

The PINN method was found to be able to accurately simulate the laminar flow case, but the method seemed to be sensitive to the specific problem geometry: at valve angles of 35° to 55°, the error values ranged between 0.2521% and 6.691% for the velocity field, and 0.06139% and 10.59% for the pressure field; at a valve angle of 75°, the error values were as large as 7.227% and 78.57% for the velocity field and pressure field, respectively.

The PINN method matched the FVM solution in the turbulent regime, qualitatively, which is a promising result; however, quantitative accuracy will need to improve substantially for this methodology to be a useful scientific tool in this regime. At higher degrees of stenosis, with valve angles between 35° and 55°, the error values ranged between 5.529% and 41.82% for the velocity field, and between 58.87% and 280.2% for the pressure field. At a valve angle of 75°, the velocity and pressure errors were as large as 21.58% and 4168%, respectively.

The general trend seems to be that the PINN method was able to infer more accurate velocity fields than pressure fields: this is likely, as the $\psi - p$ formulation of the Navier–Stokes equations was used, which enforces continuity exactly, but leaves a finite residual in the momentum equation. Thus, we observed that the pressure field was more sensitive to errors in the momentum equation than was the velocity field.

Simulation times were significantly longer with the PINN method than with the FVM, with the PINN taking up to two orders of magnitude longer to converge than did the FVM code. This currently makes PINNs an unattractive option in a scenario like the one presented in this paper, where a well-posed, non-parametric problem must be solved. Much research is being done to understand and improve the training dynamics of PINNs [20,38,39], which will hopefully lead to a more efficient training process. The authors also note that in this study, accuracy was prioritized over computational efficiency, and that the custom code used for the PINN was not pre-compiled or optimized.

The PINN method becomes much more attractive when the problem is ill-posed: for example, the problem of simulating transvalvular flow, where the inlet boundary condition is unknown but some sparse velocity measurements can be taken by using 4D flow MRI. Solving such a problem with the FVM requires a post-optimization procedure that is currently impossible, due to its computational expense. With the PINN method, however, the solution time is approximately the same as, or faster than, the well-posed, data-less case.

Once a PINN is trained, inference can be performed in real time with little computational expense: this makes PINNs a good candidate for use in real-time monitoring and control systems, as a surrogate model. Another advantage of using PINNs as surrogate models is that known physics is enforced, allowing for more reliable extrapolation, if that is required. PINNs also offer favorable scaling of the computational expense of constructing surrogate models for parameterized problems, because the solution is computed across the parameter range in parallel: this means that considerable speedup can be achieved for highly parameterized problems.

Future work in PINNs research should include further improvement of the numerics of PINNs. As noted above, PINNs suffer from numerical stiffness of the training dynamics, and it has been shown that a substantial reduction in the training time, and improvement in the prediction accuracy, can be achieved by improving the numerics of the PINN methodology. Such improvements can be achieved with alternative network architectures, such as convolutional neural networks (CNNs) [40] or graph neural networks (GNNs): these architectures are expected to perform better, because they introduce a sense of locality to the network. The information from collocation points that are close together interacts differently in these networks than does the information from collocation points that are far apart: this mirrors how regions of fluid that are close together interact differently than regions of fluid that are far apart. Future work should also include further investigation of methods for performing automatic weighting of the loss function terms. Some work has been done in this area [39,41], but more work is needed, to find a method that works well for all problems. Finally, the authors believe that some fundamental research is needed to understand why PINNs perform so poorly, with the formulation used in this paper, in the turbulent regime.

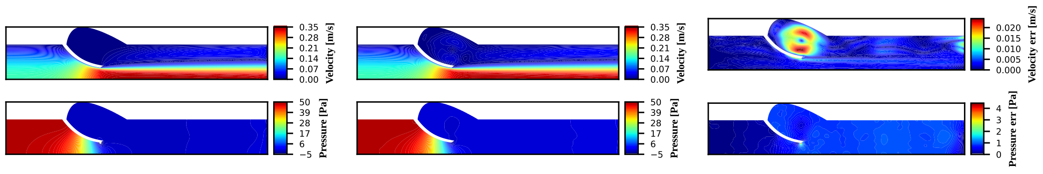
Author Contributions: The authors have contributed to this work as follows: J.F.D.T.; methodology, software, validation, formal analysis, writing—original draft preparation, writing—review and editing, visualization. R.L.: methodology, software, investigation, resources, writing—review and editing, supervision, project administration. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by the African Conference on Computational Mechanics.

Data Availability Statement: Data can be provided upon reasonable request to the authors.

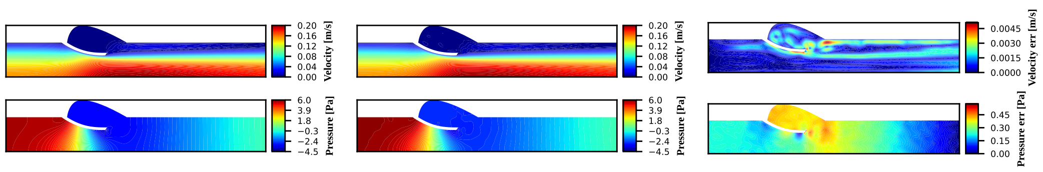
Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Simulation Results



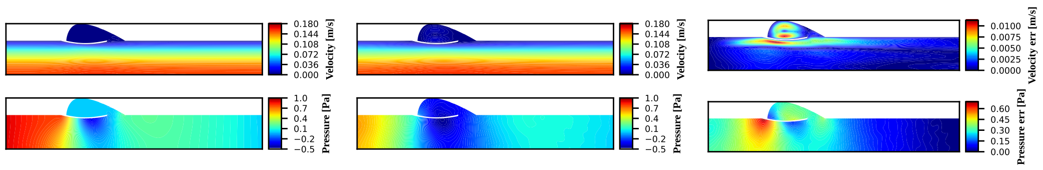
(**a**) FVM (**b**) PINN (**c**) ERR

Figure A1. Laminar flow: 35 degree valve.



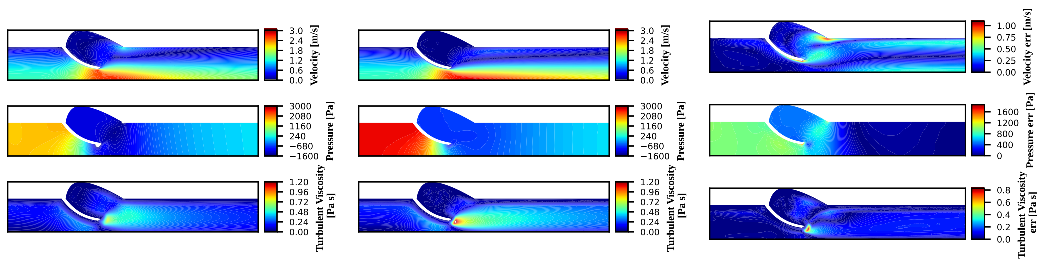
(**a**) FVM (**b**) PINN (**c**) ERR

Figure A2. Laminar flow: 55 degree valve.



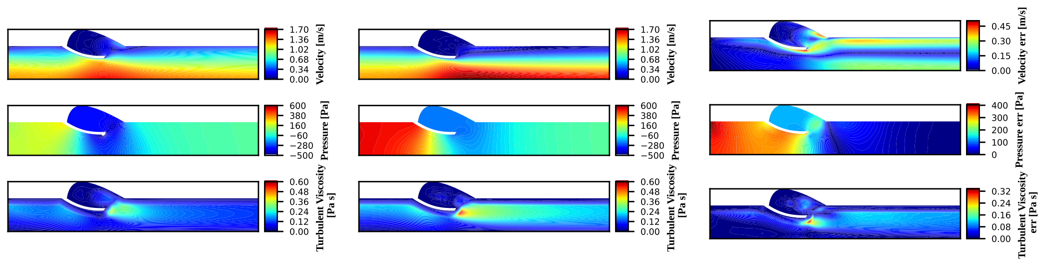
(**a**) FVM (**b**) PINN (**c**) ERR

Figure A3. Laminar flow: 75 degree valve.



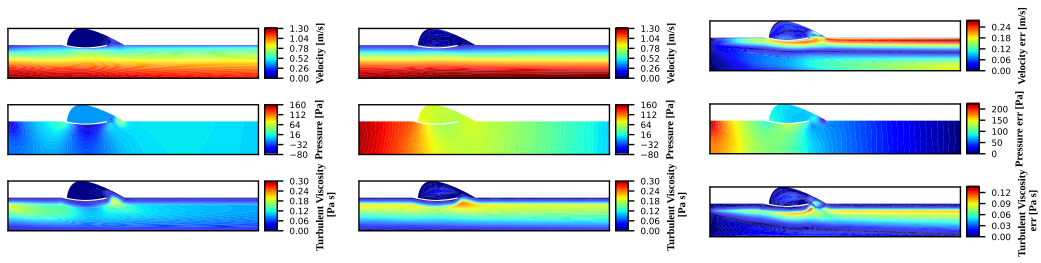
(**a**) FVM (**b**) PINN (**c**) ERR

Figure A4. Turbulent flow: 35 degree valve.



(**a**) FVM (**b**) PINN (**c**) ERR

Figure A5. Turbulent flow: 55 degree valve.



(**a**) FVM (**b**) PINN (**c**) ERR

Figure A6. Turbulent flow: 75 degree valve.

References

1. Entezari, A.; Liu, N.-C.; Zhang, Z.; Fang, J.; Wu, C.; Wan, B.; Swain, M.; Li, Q. Nondeterministic multiobjective optimization of 3d printed ceramic tissue scaffolds. *J. Mech. Behav. Biomed. Mater.* **2023**, *138*, 105580. [CrossRef]
2. 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.* **2019**, *378*, 686–707. [CrossRef]
3. Laubscher, R.; Rousseau, P. Application of a mixed variable physics-informed neural network to solve the incompressible steady-state and transient mass, momentum, and energy conservation equations for flow over in-line heated tubes. *Appl. Soft Comput.* **2022**, *114*, 108050. [CrossRef]
4. 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]
5. Cai, S.; Wang, Z.; Wang, S.; Perdikaris, P.; Karniadakis, G. Physics-informed neural networks (pinns) for heat transfer problems. *J. Heat Transf.* **2021**, *143*, 060801. [CrossRef]
6. Bihlo, A.; Popovych, R.O. Physics-informed neural networks for the shallow-water equations on the sphere. *J. Comput. Phys.* **2022**, *456*, 111024. [CrossRef]
7. Shi, S.; Liu, D.; Huo, Z. Simulation of flow field in silicon single-crystal growth using physics-informed neural network with spatial information. *Phys. Fluids* **2022**, *34*, 113610. [CrossRef]
8. Laubscher, R.; Rousseau, P.; Meyer, C. Modeling of inviscid flow shock formation in a wedge-shaped domain using a physics-informed neural network-based partial differential equation solver. In Proceedings of the ASME Turbo Expo 2022: Turbomachinery Technical Conference and Exposition, Rotterdam, The Netherlands, 13–17 June 2022.
9. Laubscher, R. Simulation of multi-species flow and heat transfer using physics-informed neural networks. *Phys. Fluids* **2021**, *33*, 087101. [CrossRef]

10. Rao, C.; Sun, H.; Liu, Y. Physics-informed deep learning for incompressible laminar flows. *Theor. Appl. Mech. Lett.* **2020**, *10*, 207–212. [CrossRef]
11. Depina, I.; Jain, S.; Valsson, S.M.; Gotovac, H. Application of physics-informed neural networks to inverse problems in unsaturated groundwater flow. *Georisk* **2022**, *16*, 21–36. [CrossRef]
12. 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]
13. Eivazi, H.; Tahani, M.; Schlatter, P.; Vinuesa, R. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Phys. Fluids* **2022**, *34*, 075117. [CrossRef]
14. 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]
15. Mahmoudi, M.; Farghadan, A.; McConnell, D.R.; Barker, A.J.; Wentzel, J.J.; Budoff, M.J.; Arzani, A. The Story of Wall Shear Stress in Coronary Artery Atherosclerosis: Biochemical Transport and Mechanotransduction. *J. Biomech. Eng.* **2020**, *143*, 041002. [CrossRef] [PubMed]
16. Hennigh, O.; Narasimhan, S.; Nabian, M.A.; Subramaniam, A.; Tangsali, K.; Fang, Z.; Rietmann, M.; Byeon, W.; Choudhry, S. *NVIDIA SimNet™: An AI-Accelerated Multi-Physics Simulation Framework*; Springer: Berlin/Heidelberg, Germany, 2021.
17. 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]
18. Versteeg, H.K.; Malalasekera, W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*; Pearson Education Limited: London, UK, 2007; Chapter 3, p. 69.
19. Yu, J.; Lu, L.; Meng, X.; Karniadakis, G.E. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Comput. Methods Appl. Mech. Eng.* **2022**, *393*, 114823. [CrossRef]
20. Wang, S.; Wang, H.; Perdikaris, P. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **2021**, *384*, 113938. [CrossRef]
21. Westaby, S.; Karp, R.B.; Blackstone, E.H.; Bishop, S.P. Adult human valve dimensions and their surgical significance. *Am. J. Cardiol.* **1984**, *53*, 552–556. [CrossRef]
22. Swanson, W.M.; Clark, R.E. Dimensions and geometric relationships of the human aortic value as a function of pressure. *Circ. Res.* **1974**, 35, 871–882. [CrossRef]
23. Perktold, K. On the paths of fluid particles in an axisymmetrical aneurysm. *J. Biomech.* **1987**, *20*, 311–317. [CrossRef]
24. Olufsen, M.S.; Peskin, C.S.; Kim, W.Y.; Pedersen, E.M.; Nadim, A.; Larsen, J. Numerical simulation and experimental validation of blood flow in arteries with structured-tree outflow conditions. *Ann. Biomed. Eng.* **2000**, 28, 1281–1299. [CrossRef] [PubMed]
25. Riegel, Y.v.J.; Mayer, W. Freecad (Version 0.19), (2001–2023). Available online: http://www.freecadweb.org (accessed on 14 February 2023).
26. Markidis, S. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? *Front. Big Data* **2021**, *4*. [CrossRef] [PubMed]
27. 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.
28. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
29. Tucker, P.G. Differential equation-based wall distance computation for des and rans. *J. Comput. Phys.* **2003**, *190*, 229–248. [CrossRef]
30. Spalding, D.B. Calculation of turbulent heat transfer in cluttered spaces. In Proceedings of the 10th International Heat Transfer Conference, Brighton, UK, 14–18 August 1994.
31. Tancik, M.; Srinivasan, P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.; Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7537–7547.
32. ANSYS Mechanical® 2023 R1. 2023. Available online: https://www.ansys.com/webinars/ansys-2023-r1-mechanical-update (accessed on 14 February 2023).
33. ANSYS Fluent® 2023 R1. 2023. Available online: https://www.ansys.com/webinars/ansys-2023-r1-ansys-fluent-whats-new (accessed on 14 February 2023).
34. ANSYS Inc. *ANSYS Fluent User Guide*; ANSYS Inc.: Canonsberg, PA, USA, 2015.
35. ANSYS Inc. *ANSYS Fluent Theory Guide*; ANSYS Inc.: Canonsberg, PA, USA, 2015.
36. Rao, S.S. *Engineering Optimization: Theory and Practice*; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2019.
37. Brooks, R.C.; Hammermeister, K.E.; Warbasse, J.R. The rate of change of left ventricular volume in man. *Circulation* **1974**, *49*, 729–738.
38. Wang, S.; Teng, Y.; Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.* **2021**, *43*, A3055–A3081. [CrossRef]
39. Wang, S.; Yu, X.; Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *J. Comput. Phys.* **2022**, *449*, 110768. [CrossRef]

40. Gao, H.; Sun, L.; Wang, J.-X. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *J. Comput. Phys.* **2021**, *428*, 110079. [CrossRef]
41. Maddu, S.; Sturm, D.; Muller, C.L.; Sbalzarini, I.F. Inverse-dirichlet weighting enables reliable training of physics informed neural networks. *Mach. Learn. Sci. Technol.* **2021**, *3*, 015026. [CrossRef]