

Fluid-Structure Interaction Analysis of Pulsatile Flow in Arterial Aneurysms with Physics-Informed Neural Networks and Computational Fluid Dynamics

Michael Ajao-Olarinoye¹

¹Center for Computational Science and Mathematical Modelling

olarinoyem@coventry.ac.uk

1 Physics-Informed Neural Networks

1.1 Overview

High-fidelity Computational Fluid Dynamics (CFD) simulations (Section 2.4) offer detailed insights into arterial hemodynamics but are computationally expensive and require significant expertise in mesh generation, solver tuning, and post-processing. In contrast, Physics-Informed Neural Networks (PINNs) [?] combine deep learning with physics-based modeling by embedding the governing partial differential equations (PDEs) directly into the network's loss function. This integration enables the network to learn realistic flow solutions while significantly reducing the need for extensive CFD re-runs when parameters or geometries change.

By incorporating PDE residuals and boundary conditions into the training process, PINNs naturally enforce physical constraints, reduce overfitting, and require fewer labeled data points. Consequently, PINNs can complement or even partially replace traditional CFD simulations, offering:

- 1) **Denser solution fields:** PINNs can be evaluated at arbitrary spatial and temporal points, yielding high-resolution predictions that surpass the original CFD mesh density.
- 2) **Enhanced generalization:** Adaptation to new boundary conditions or geometrical changes is possible with minimal retraining.
- 3) **Reduced computational cost:** Once trained, PINN inference is rapid, making real-time or iterative analyses feasible.

In this study, we integrate PINNs with CFD data to predict pressure, velocity components, and wall shear stress (WSS) in both healthy and Marfan Syndrome aortic models (Section 2.1), enabling a direct comparison of flow patterns and shear forces between non-pathological and aneurysmal vessels.

1.2 Formulation of Physics-Informed Neural Networks

Modeling pulsatile flow in arterial segments requires that the continuity and momentum conservation equations be enforced to capture the essential hemodynamic effects. In our study, the blood flow model and associated boundary conditions are described in Sections 2.2 and 2.3 using the Reynolds-averaged Navier–Stokes (RANS) equations (Equations (2.1)–(2.9)). Let $\mathbf{u} = (u, v, w)$ denote the velocity field, p the pressure, ρ the fluid density, and μ the dynamic viscosity. The continuity and momentum equations are combined into the Navier–Stokes system, which governs the evolution of the velocity field and pressure in response to external forces and viscous effects.

In standard index notation (with the Einstein summation convention), the governing equations are written as

$$\begin{aligned} \frac{\partial u_i}{\partial x_i} &= 0, \\ \rho \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) &= -\frac{\partial p}{\partial x_i} + \mu \frac{\partial^2 u_i}{\partial x_j^2}, \end{aligned} \tag{3.1}$$

where $i, j = 1, 2, 3$ correspond to the spatial coordinates. Equation (3.1) thus encapsulates both the continuity equation (first line) and the momentum (Navier–Stokes) equation (second line).

The boundary conditions are as follows:

- A no-slip condition on the arterial walls, i.e., $\mathbf{u} = \mathbf{0}$ on the wall boundary,
- Zero relative pressure at the outlet, and
- A pulsatile inlet velocity prescribed on the inlet boundary.

In our formulation, these conditions are enforced via penalty terms in the loss function.

For example, the no-slip and inlet conditions are imposed as

$$L_{\text{boundary}} = \left\| \mathbf{u} \Big|_{\partial\Omega_{\text{wall}}} - \mathbf{0} \right\|_2, \quad L_{\text{inlet}} = \left\| \mathbf{u} \Big|_{\partial\Omega_{\text{inlet}}} - \mathbf{u}_{\text{inlet}}(t) \right\|_2,$$

where $\partial\Omega_{\text{wall}}$ and $\partial\Omega_{\text{inlet}}$ denote the vessel wall and inlet boundaries, respectively.

Within the PINN framework, neural networks are used to approximate the functions

$$u(x, y, z, t), \quad v(x, y, z, t), \quad w(x, y, z, t), \quad p(x, y, z, t),$$

as well as the three wall shear stress (WSS) components

$$\tau_x(x, y, z, t), \quad \tau_y(x, y, z, t), \quad \tau_z(x, y, z, t).$$

Together, these functions capture the spatiotemporal evolution of blood flow and the shear forces acting on the vessel walls.

Let $F(\cdot)$ represent the nonlinear operator corresponding to the Navier–Stokes system. The physics residual loss is then defined as

$$L_{\text{physics}} = \left\| F(\mathbf{u}, p) \right\|_2,$$

where $\|\cdot\|_2$ denotes the Euclidean norm. Minimizing this loss term drives the network to produce solutions that are consistent with the Navier–Stokes and continuity equations, even in regions where direct measurements are not available.

Furthermore, when CFD reference data (e.g., \mathbf{u}_{CFD} , p_{CFD} , and $\boldsymbol{\tau}_{\text{CFD}}$) are available, a data-fitting loss is incorporated:

$$L_{\text{data}} = \left\| \mathbf{u}_{\text{NN}} - \mathbf{u}_{\text{CFD}} \right\|_2 + \left\| p_{\text{NN}} - p_{\text{CFD}} \right\|_2 + \left\| \boldsymbol{\tau}_{\text{NN}} - \boldsymbol{\tau}_{\text{CFD}} \right\|_2.$$

The total loss function is then expressed as

$$L_{\text{total}} = \lambda_{\text{physics}} L_{\text{physics}} + \lambda_{\text{boundary}} L_{\text{boundary}} + \lambda_{\text{inlet}} L_{\text{inlet}} + \lambda_{\text{data}} L_{\text{data}}, \quad (1)$$

where λ_{physics} , $\lambda_{\text{boundary}}$, λ_{inlet} , λ_{data} are self-adaptive weighting coefficients that are optimized during training to balance the contributions of each loss component [?]. The

details of the self-adaptive weighting mechanism are discussed in Section ??.

1.3 Formulation of Physics-Informed Neural Networks

Modeling pulsatile flow in arterial segments requires enforcing the fundamental laws of fluid mechanics—namely, the continuity and Navier–Stokes equations—to capture the essential hemodynamic effects. In our study, the blood flow in aortic geometries is modeled using the Reynolds-averaged Navier–Stokes (RANS) framework (see Sections 2.2 and 2.3). Let $\mathbf{u} = (u, v, w)$ denote the velocity field, p the pressure, ρ the fluid density, and μ the dynamic viscosity. The Navier–Stokes system, which governs the evolution of the velocity field and pressure under the influence of external forces and viscous effects, is expressed in standard index notation as

$$\begin{aligned} \frac{\partial u_i}{\partial x_i} &= 0, \\ \rho \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) &= -\frac{\partial p}{\partial x_i} + \mu \frac{\partial^2 u_i}{\partial x_j^2}, \end{aligned} \quad (3.1)$$

where the Einstein summation convention is assumed (i.e., summing over repeated indices $i, j = 1, 2, 3$). In our simulations (Sections 2.2 and 2.3), the boundary conditions are prescribed as follows: a no-slip condition on the arterial walls (i.e., $\mathbf{u} = \mathbf{0}$ on the wall boundary), zero relative pressure at the outlet, and a pulsatile inlet velocity on the inlet boundary. These conditions are incorporated into the PINN framework through penalty terms within the loss function, thereby ensuring that the network’s predictions remain consistent with the physical constraints and the transient nature of blood flow.

Within the PINN framework, the neural network approximates the functions

$$u(x, y, z, t), \quad v(x, y, z, t), \quad w(x, y, z, t), \quad p(x, y, z, t),$$

as well as the three wall shear stress (WSS) components

$$\tau_x(x, y, z, t), \quad \tau_y(x, y, z, t), \quad \tau_z(x, y, z, t).$$

Together, these variables capture the spatiotemporal evolution of blood flow and the shear forces acting on the vessel walls.

Let $F(\cdot)$ denote the nonlinear operator corresponding to the Navier–Stokes system defined in Equation (3.1). The physics residual loss is defined as

$$L_{\text{physics}} = \left\| F(\mathbf{u}, p) \right\|_2,$$

where $\|\cdot\|_2$ denotes the Euclidean norm. Minimizing this term drives the network to learn solutions that are consistent with the governing equations even in regions where direct data are sparse.

Additionally, the boundary and inlet conditions are enforced via penalty terms:

$$L_{\text{boundary}} = \left\| \mathbf{u}|_{\partial\Omega_{\text{wall}}} - \mathbf{0} \right\|_2, \quad L_{\text{inlet}} = \left\| \mathbf{u}|_{\partial\Omega_{\text{inlet}}} - \mathbf{u}_{\text{inlet}}(t) \right\|_2,$$

where $\partial\Omega_{\text{wall}}$ and $\partial\Omega_{\text{inlet}}$ denote the wall and inlet boundaries, respectively, and $\mathbf{u}_{\text{inlet}}(t)$ is the prescribed pulsatile inlet velocity profile (Section 2.3). When available, CFD reference data for pressure, velocity, and WSS—denoted by \mathbf{u}_{CFD} , p_{CFD} , and $\boldsymbol{\tau}_{\text{CFD}}$ —are used to define a data-fitting loss:

$$L_{\text{data}} = \left\| \mathbf{u}_{\text{NN}} - \mathbf{u}_{\text{CFD}} \right\|_2 + \left\| p_{\text{NN}} - p_{\text{CFD}} \right\|_2 + \left\| \boldsymbol{\tau}_{\text{NN}} - \boldsymbol{\tau}_{\text{CFD}} \right\|_2.$$

Here, the subscripts “NN” and “CFD” denote the PINN predictions and the CFD reference values, respectively.

The total loss function, which combines the physics residual, boundary, inlet, and data-fitting terms, is then given by

$$L_{\text{total}} = \lambda_{\text{physics}} L_{\text{physics}} + \lambda_{\text{boundary}} L_{\text{boundary}} + \lambda_{\text{inlet}} L_{\text{inlet}} + \lambda_{\text{data}} L_{\text{data}}, \quad (3.2)$$

where the weighting coefficients λ_{physics} , $\lambda_{\text{boundary}}$, λ_{inlet} , λ_{data} are determined via a self-adaptive scheme [?] to automatically balance the contributions of each term during training.

1.4 Formulation of PINNs

2 Physics-Informed Neural Networks (PINNs)

Although high-fidelity CFD simulations provide accurate haemodynamic predictions, they often incur a high computational cost. To address this limitation, we employ *Physics-Informed Neural Networks* (PINNs), which embed the governing fluid equations (Navier–Stokes, continuity, turbulence closure) and boundary conditions directly into a neural-network loss function. In doing so, PINNs can learn accurate velocity, pressure, and wall shear stress distributions without relying exclusively on large, labeled datasets. Instead, the PDEs themselves act as guiding constraints in the training process.

2.1 PINN Framework and Motivation

A PINN can be viewed as a neural network that takes the spatiotemporal coordinates, (x, y, z, t) , as inputs and outputs an approximation to the relevant flow fields. In this study, the network separately predicts

- Pressure, $p(x, y, z, t)$,
- Velocity components, $(u, v, w)(x, y, z, t)$,
- Wall shear stress (WSS) components, $(\tau_x, \tau_y, \tau_z)(x, y, z, t)$.

Unlike conventional machine-learning models, PINNs penalize deviations from both *physical laws* and *observed data*, creating a hybrid loss function. This approach allows the model to exploit fundamental conservation principles—such as incompressibility and momentum transport—even in regions where direct data (e.g. from simulations or experiments) may be sparse.

2.2 Constructing the PINN Loss Function

In our code, the overall PINN loss is composed of four main terms:

$$L_{\text{total}} = \lambda_{\text{physics}} L_{\text{physics}} + \lambda_{\text{boundary}} L_{\text{boundary}} + \lambda_{\text{inlet}} L_{\text{inlet}} + \lambda_{\text{data}} L_{\text{data}},$$

where $\lambda_{\text{physics}}, \lambda_{\text{boundary}}, \lambda_{\text{inlet}}, \lambda_{\text{data}}$ are *self-adaptive weights* learned during training. The sub-terms are defined as follows.

1. Physics Residual Loss (L_{physics}). This term enforces the Navier–Stokes and continuity equations, ensuring that the neural-network predictions respect conservation of mass and momentum. Formally, the code computes residuals by differentiating the neural-network outputs (u, v, w, p) with respect to (x, y, z, t) and substituting these derivatives into the PDEs. Any deviation from zero is penalized via a Mean Squared Error (MSE) objective.

2. Boundary Condition Loss (L_{boundary}). Because we impose a no-slip condition on arterial walls, points identified as “wall boundary” in the dataset must have $(u, v, w) = 0$. In the code, a small tolerance is used to detect boundary points, and any non-zero values at these locations are penalized in the loss.

3. Inlet Condition Loss (L_{inlet}). A time-varying pulsatile profile is prescribed at the inlet boundary (Section ??). The PINN enforces this condition by comparing predicted velocities at inlet points to the known sinusoidal (or piecewise) function. Any mismatch is penalized to ensure the model reproduces the correct pulsatile flow.

4. Data-Fitting Loss (L_{data}). Wherever reference data exist (from CFD or experiments), the network predictions— $(p_{\text{NN}}, \mathbf{u}_{\text{NN}}, \tau_{\text{NN}})$ —are matched to the known values, $(p_{\text{CFD}}, \mathbf{u}_{\text{CFD}}, \tau_{\text{CFD}})$. The MSE of these differences constitutes L_{data} . This ensures that the PINN conforms to any available, high-fidelity information.

2.3 Self-Adaptive Weighting

One innovative feature of this setup is that the four weighting coefficients $\lambda_{\text{physics}}, \lambda_{\text{boundary}}, \lambda_{\text{inlet}}, \lambda_{\text{data}}$ are *learnable parameters*. Rather than fixing them manually, each weight is stored in the code as $\log \lambda$, which the model updates during backpropagation. This technique, known as *self-adaptive loss balancing* [?], helps the training process converge more robustly. Early in training, the network may focus on satisfying the easier constraints (e.g. data-fitting), then gradually allocate more effort to meeting the stricter PDE requirements, or vice versa, as needed.

2.4 PINN Training Workflow

1. **Coordinate and Data Preparation:** The inputs, (x, y, z, t) , are prepared in PyTorch tensors with `requires_grad=True` so that automatic differentiation can compute partial derivatives for the PDE residuals. The velocity, pressure, and WSS data are similarly loaded for supervised matching where available.
2. **Forward Pass:** Each specialized PINN (for $p, u, v, w, \tau_x, \tau_y, \tau_z$) takes the input coordinates and returns its predicted field variable. `torch.autograd.grad` then computes spatial and temporal gradients used in the physics loss.
3. **Loss Computation:** - L_{physics} is calculated by inserting the predicted fields into the continuity and momentum equations. - L_{boundary} checks velocity at boundary points. - L_{inlet} enforces the inlet velocity profile. - L_{data} matches network outputs to known CFD data. These components are combined using the current (learned) values of $\lambda_{\text{physics}}, \dots, \lambda_{\text{data}}$.
4. **Backpropagation:** The total loss L_{total} is backpropagated to update both the PINN's network weights (which define how it approximates the fields) and the log-weights for the self-adaptive coefficients.
5. **Convergence and Early Stopping:** Training continues until a defined convergence criterion or stopping tolerance is met, often monitored via a validation set or by observing a plateau in the total loss. This is supported by techniques like gradient clipping and mixed precision to maintain numerical stability.

2.5 Advantages and Outlook

By incorporating physical laws within the loss function, PINNs can learn flow solutions in regions where no direct data are available. This property makes them appealing for scenarios in which collecting or storing extensive CFD results is infeasible. Additionally, once trained, PINNs can operate as a *surrogate model* for rapid evaluations under varying boundary conditions, geometry changes, or material properties. In the context of arterial haemodynamics, this could facilitate near real-time analysis of aneurysms, blood-flow distributions, or wall stresses, potentially aiding clinical decision-making.

Summary. In essence, PINNs unify traditional data-driven regression with fundamental conservation principles. Instead of learning from data *alone*, the network also “learns” from the PDEs, boundary conditions, and the known inlet/outlet configurations. The subsequent sections detail how our PINN-based method compares to high-fidelity CFD simulations in predicting key haemodynamic parameters (velocity, pressure, and WSS) in both healthy and aneurysmal aortic geometries.

2.6 Formulation of Physics-Informed Neural Networks

Modeling pulsatile flow in arterial segments requires enforcing the continuity and Navier–Stokes equations. Let $\mathbf{u} = (u, v, w)$ denote the velocity field, p the pressure, ρ the fluid density, and μ the dynamic viscosity. In a conventional (index) notation, these governing equations are expressed as

$$\begin{aligned} \frac{\partial u_i}{\partial x_i} &= 0, \\ \rho \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) &= -\frac{\partial p}{\partial x_i} + \mu \frac{\partial^2 u_i}{\partial x_j^2}, \end{aligned} \quad (3.1)$$

where the Einstein summation convention is assumed over repeated indices (i.e., $i, j = 1, 2, 3$). Equation (3.1) represents the continuity equation (first line) and the momentum (Navier–Stokes) equation (second line) in a compact, standard format.

Boundary conditions are imposed as follows:

- **No-slip at vessel walls:** $\mathbf{u} = \mathbf{0}$ on $\partial\Omega_{\text{wall}}$.
- **Zero relative pressure:** Applied at the outlet.
- **Pulsatile inlet velocity:** A time-dependent profile is prescribed on $\partial\Omega_{\text{inlet}}$ (Section 2.3).

In the PINN framework, the neural network approximates the functions

$$u(x, y, z, t), \quad v(x, y, z, t), \quad w(x, y, z, t), \quad p(x, y, z, t),$$

as well as the wall shear stress components

$$\tau_x(x, y, z, t), \quad \tau_y(x, y, z, t), \quad \tau_z(x, y, z, t).$$

The physics residual loss L_{physics} is defined by the deviation from the Navier–Stokes operator:

$$L_{\text{physics}} = \|F(\mathbf{u}, p)\|_2,$$

where $F(\cdot)$ represents the left-hand side of Equation (3.1) minus the right-hand side. Additional loss terms enforce the boundary conditions:

$$L_{\text{boundary}} = \|\mathbf{u}|_{\partial\Omega_{\text{wall}}} - \mathbf{0}\|_2, \quad L_{\text{inlet}} = \|\mathbf{u}|_{\Gamma_{\text{inlet}}} - \mathbf{u}_{\text{inlet}}(t)\|_2.$$

Furthermore, if CFD reference data $(\mathbf{u}_{\text{CFD}}, p_{\text{CFD}}, \tau_{\text{CFD}})$ are available, a data-fitting loss is included:

$$L_{\text{data}} = \|\mathbf{u}_{\text{NN}} - \mathbf{u}_{\text{CFD}}\|_2 + \|p_{\text{NN}} - p_{\text{CFD}}\|_2 + \|\tau_{\text{NN}} - \tau_{\text{CFD}}\|_2.$$

Thus, the total loss is given by

$$L_{\text{total}} = \lambda_{\text{physics}} L_{\text{physics}} + \lambda_{\text{boundary}} L_{\text{boundary}} + \lambda_{\text{inlet}} L_{\text{inlet}} + \lambda_{\text{data}} L_{\text{data}}, \quad (2)$$

where the weights λ_i are determined via a self-adaptive loss weighting scheme (Section 3.3.1).

2.7 Neural Network Architecture

In this study, we train separate PINNs for pressure p , velocity components (u, v, w) , and the three WSS components (τ_x, τ_y, τ_z) , each as a function of space and time t . Each network is a fully connected feed-forward architecture with $L = 10$ hidden layers and $N = 64$ neurons per layer. We employ the Swish activation function [?], defined as

$$\text{Swish}(x) = x \sigma(\beta x),$$

where $\sigma(\cdot)$ is the sigmoid function and β is a learnable parameter. Batch normalization is applied to each layer to improve stability, and weights are initialized using the Kaiming normal method [?].

2.7.1 Self-Adaptive Loss Weighting

Balancing multiple loss components (physics, boundary, inlet, and data) is critical for successful training. To achieve this, we employ a self-adaptive weighting strategy inspired by [?]. We introduce learnable parameters $\log \lambda_i$ (for $i \in \{\text{phys}, \text{bound}, \text{inlet}, \text{data}\}$) such that

$$\lambda_i = \exp(\log \lambda_i),$$

ensuring that the weights remain positive throughout training. These weights are optimized jointly with the network parameters, allowing the model to automatically balance the contributions of the various loss components. Algorithm ?? outlines this process.

Algorithm 1 Self-Adaptive Loss Weighting in PINNs

Require: Initial network weights θ , initial $\log \lambda_i$, learning rates for θ and $\log \lambda_i$

Ensure: Trained network weights θ^* and optimized λ_i^*

```

1: while not converged do
2:   Sample mini-batch of training points
3:   Forward pass to compute predictions  $\mathbf{u}_{\text{NN}}, p_{\text{NN}}, \boldsymbol{\tau}_{\text{NN}}$ 
4:   Compute individual loss components:
5:      $\mathcal{L}_{\text{phys}} = \|F(\mathbf{u}_{\text{NN}}, p_{\text{NN}})\|_2^2$ 
6:      $\mathcal{L}_{\text{bound}} = \|\mathbf{u}_{\text{NN}}|_{\partial\Omega_{\text{wall}}} - \mathbf{0}\|_2^2$ 
7:      $\mathcal{L}_{\text{inlet}} = \|\mathbf{u}_{\text{NN}}|_{\Gamma_{\text{inlet}}} - \mathbf{u}_{\text{inlet}}(t)\|_2^2$ 
8:      $\mathcal{L}_{\text{data}} = \|\mathbf{u}_{\text{NN}} - \mathbf{u}_{\text{CFD}}\|_2^2 + \|p_{\text{NN}} - p_{\text{CFD}}\|_2^2 + \|\boldsymbol{\tau}_{\text{NN}} - \boldsymbol{\tau}_{\text{CFD}}\|_2^2$ 
9:   Set  $\mathcal{L}_{\text{total}} = \lambda_{\text{phys}} \mathcal{L}_{\text{phys}} + \lambda_{\text{bound}} \mathcal{L}_{\text{bound}} + \lambda_{\text{inlet}} \mathcal{L}_{\text{inlet}} + \lambda_{\text{data}} \mathcal{L}_{\text{data}}$ 
10:  Backpropagate  $\mathcal{L}_{\text{total}}$  to compute gradients for  $\theta$  and  $\log \lambda_i$ 
11:  Update network weights:  $\theta \leftarrow \theta - \eta_{\theta} \nabla_{\theta} \mathcal{L}_{\text{total}}$ 
12:  Update log-weights:  $\log \lambda_i \leftarrow \log \lambda_i - \eta_{\lambda} \nabla_{\log \lambda_i} \mathcal{L}_{\text{total}}$ 
13: end while
14: Return  $\theta^*, \lambda_i^*$ 

```

2.8 Data Processing and Experimental Setup

CFD simulation data from healthy (0024 and 0142) and aneurysmal (0021, 0022, 0023, 0025) aortic geometries were obtained for both systolic and diastolic phases. Each dataset includes spatial coordinates (X, Y, Z), time, pressure, velocity (u, v, w), and wall shear stress (x, y, z). The data processing steps were as follows:

1. **Data Cleaning:** Remove incomplete records and verify physical consistency, yielding approximately 11,005 points per time step.

2. **Normalization:** Normalize spatial coordinates to the interval $[-1, 1]$, time to $[0, 1]$ over the cardiac cycle, and each of the physical variables independently using Min–Max scaling.
3. **Boundary Identification:** Identify vessel-wall points using a velocity magnitude threshold ($|u|, |v|, |w| < 10^{-5}$) to enforce the no-slip condition.
4. **Data Organization:** Organize the data by grouping healthy and aneurysmal cases with uniform time sampling.
5. **Quality Control:** Perform checks for consistency in velocity and pressure fields, verify mass conservation at inlet/outlet, and ensure temporal continuity.

2.9 Point Density Analysis and Spatial Discretization

A significant advantage of PINNs is the ability to produce mesh-free, continuous predictions of flow variables. Our analysis of the training dataset confirmed that 11,005 points provide uniform spatial coverage and sufficient resolution to capture the complex flow features in both healthy and aneurysmal geometries. Furthermore, once trained, the PINN can generate predictions at a density approximately 100.5 times higher (over 1,100,000 points) while maintaining rapid inference speeds.

Figure ?? displays the 2D projections of the training point distribution, while Figure ?? compares the original CFD point distribution with the high-resolution PINN predictions and illustrates the computational time benefits of the PINN approach.

2.10 Training of the PINNs

The PINN framework was implemented in Python using PyTorch [?] and trained on a high-performance computing cluster with NVIDIA RTX8000 GPUs. The spatiotemporal data points (x_j, y_j, z_j, t_j) were randomly sampled from the CFD datasets and normalized as described above. The networks were trained using the AdamW optimizer [?] with an initial learning rate of 1×10^{-4} , weight decay of 1×10^{-4} , and momentum parameters (0.9, 0.999). A StepLR scheduler reduced the learning rate by a factor of 0.9 every 200 epochs. Training employed mixed-precision arithmetic [?] and gradient clipping



Figure 1: 2D projections (XY, XZ, YZ) of the training points, demonstrating uniform spatial coverage.



Figure 2: Comparison between CFD data points and PINN predictions. (a) Original CFD distribution, (b) enhanced PINN prediction density, and (c) computational time comparison.

(maximum norm 1.0) for stability, and was terminated early if the validation loss did not improve for 5 consecutive epochs, with a maximum of 1000 epochs.

2.11 Relevance to Aneurysm Studies

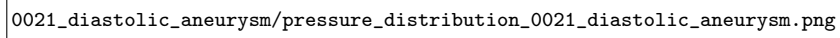
PINNs offer a rapid, mesh-free alternative to full CFD simulations, particularly when boundary conditions or geometries are modified. This is of considerable clinical interest for Marfan Syndrome aortic aneurysms, where the ability to quickly explore a range of inflow profiles, wall properties, and geometrical variations can significantly enhance diagnostic and treatment planning. By maintaining the governing physics while enabling high-resolution predictions, PINNs facilitate real-time assessments and efficient sensitivity analyses, making them a promising tool for personalized vascular evaluations.

Note: All code and numerical experiments for the PINN-based aneurysm flow analyses were implemented using Python and PyTorch. The self-adaptive weighting and physics-based PDE residuals were integrated into the backpropagation routine, ensuring reproducibility and transparency of the results.

2.12 Results

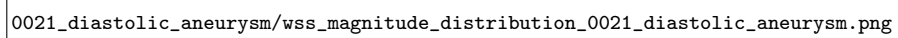
The PINN-based simulations were validated against benchmark CFD data for both healthy and Marfan Syndrome aortic geometries. The results demonstrate excellent agreement in velocity profiles, pressure distributions, and wall shear stress (WSS) patterns. In particular, the PINNs successfully captured complex flow phenomena such as vortex formation, flow separation, and recirculation within the aneurysm sac. The self-adaptive loss weighting mechanism effectively balanced the contributions from the physics residuals, boundary conditions, and data-fitting terms, guiding the network toward physically realistic solutions.

Figure ?? illustrates the comparison between CFD and PINN predictions for pressure and WSS in a representative Marfan Syndrome case (0021). Additionally, Figure ?? shows the evolution of the loss components over 1000 epochs, confirming stable convergence and effective balancing by the self-adaptive mechanism.



0021_diastolic_aneurysm/pressure_distribution_0021_diastolic_aneurysm.png

(a) Pressure distribution



0021_diastolic_aneurysm/wss_magnitude_distribution_0021_diastolic_aneurysm.png



Figure 4: Loss curves for the PINN training process, showing the evolution of the physics, boundary, inlet, and data-fitting losses over 1000 epochs. The self-adaptive loss weighting mechanism ensures balanced convergence.

References