PHYSICS-INFORMED NEURAL NETWORKS FOR MODELING TRANSIENT BLOOD FLOW IN ANEURYSMS *

Author1, Author2
Affiliation
Univ
City
{Author1, Author2}email@email

Author3
Affiliation
Univ
City
email@email

ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

***K*eywords** First keyword · Second keyword · More

1 Introduction

https://www.ctan.org/pkg/booktabs

High-fidelity Computational Fluid Dynamics (CFD) simulations, as outlined in Section 2.4, offer precise insights into arterial hemodynamics and are central to modelling fluid dynamics in complex vascular geometries. However, these simulations can be computationally expensive and time consuming, especially when dealing with patient-specific anatomies or performing numerous parameter sweeps. Such challenges are further compounded by the need for significant domain expertise to refine meshes, tune solvers, and interpret results.

Physics-Informed Neural Networks (PINNs), first introduced by **(author?)** [1], have emerged as a powerful alternative that combines the strengths of traditional deep learning with established physics-based modelling. Rather than relying exclusively on data-driven learning, PINNs integrate the governing partial differential equations (PDEs) in this case, the continuity and momentum conservation equations (Section 2.2) directly into the network's loss function. This approach ensures that the neural network is "informed" by the physical laws of fluid motion, guiding it toward realistic flow solutions even when the labelled data are sparse or imperfectly measured.

By incorporating PDE residuals into the loss function, PINNs naturally embed a priori knowledge of fluid physics, allowing the model to learn physically plausible flow fields with fewer labelled samples than a purely data-driven neural network. This built-in adherence to physical constraints also helps mitigate overfitting and reduces the likelihood of predicting unphysical solutions. As a result, PINNs have the potential to complement or partially replace expensive CFD simulations, thereby lowering computational overhead and accelerating research pipelines.

In this study, we combine PINNs with traditional CFD data to estimate the core hemodynamic variables such as pressure, velocity components, and wall shear stress (WSS) in healthy and Marfan syndrome aortic geometries presented in (Section 2.1). This unified approach enables direct comparisons of velocity patterns, shear forces, and other critical flow

* *Citation*: **Authors. Title. Pages.... DOI:000000/11111.**

indicators between non-pathological and aneurysmal models, providing valuable information for understanding the progression of vascular disease. The following sections detail the formulation, architecture, and training strategy of the PINNs used in this study.

1.1 Formulation of Physics-Informed Neural Networks

Modelling pulsatile flow in arterial segments requires the Navier–Stokes and continuity equations to capture essential haemodynamic effects. Let $\mathbf{u} = (u, v, w)$ denote the velocity field, p the pressure, ρ the fluid density, and μ the dynamic viscosity. In vector form, the governing equations are as follows:

$$\nabla \cdot \mathbf{u} = 0, \quad \rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \mu \nabla^2 \mathbf{u}. \tag{1}$$

The associated boundary conditions involve a no-slip condition on the arterial walls ($\mathbf{u} = \mathbf{0}$ on $\partial\Omega_{\text{wall}}$), zero relative pressure at the outlet, and a pulsatile inlet velocity on $\partial\Omega_{\text{inlet}}$ (Section 2.3). These boundary and inlet constraints are incorporated into the Physics-Informed Neural Network through penalty terms in the loss function, thereby enforcing consistency between the network's predictions and the specified physical conditions. The temporal dependence is explicitly modelled to reflect the transient nature of blood flow.

Within the PINN framework, neural networks approximate the velocity field $\mathbf{u}$, pressure p, and WSS components τ_x, τ_y, τ_z as functions of the spatial coordinates (x, y, z) and time t. The network architecture is trained to minimise the physics residual loss, which quantifies the deviation of the predicted solutions from the Navier–Stokes and continuity equations.

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

and the WSS components

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

Collectively, these variables capture the spatio-temporal evolution of blood flow and the shear forces that act on the walls of the vessel.

Let $F(\cdot)$ represent the non-linear operator corresponding to the Navier–Stokes system. The loss term *physics residual* evaluates the degree to which the predictions of PINN adhere to the governing equations:

$$L_{\text{physics}} = \left\| F(\mathbf{u}, p) \right\|_2, \tag{2}$$

where $\| \cdot \|_2$ denotes the Euclidean norm. By minimising this residual, the network is driven to learn solutions consistent with the Navier–Stokes and continuity equations, even in regions where no direct labels are available.

Arterial boundaries impose $\mathbf{u} = \mathbf{0}$ at the vessel walls and zero pressure at the outlet, while a prescribed pulsatile velocity profile is enforced at the inlet:

$$L_{\text{boundary}} = \|\mathbf{u}\big|_{\partial\Omega_{\text{wall}}} - \mathbf{0}\|_2, \quad L_{\text{inlet}} = \|\mathbf{u}\big|_{\Gamma_{\text{inlet}}} - \mathbf{u}_{\text{inlet}}(t)\|_2. \tag{3}$$

These terms penalise any deviation of the network predictions from the specified boundary conditions, thus reinforcing physically realistic flow fields at the walls and the inflow region.

The CFD data with the following (pressure, velocity, and WSS) are integrated into the loss function. Let $\mathbf{u}_{\text{NN}}$ and p_{NN} denote the PINN's predicted velocity and pressure, while $\mathbf{u}_{\text{CFD}}$ and p_{CFD} are the corresponding CFD reference fields. Similarly, τ_{NN} and τ_{CFD} refer to the predicted and CFD-derived WSS. The data fitting loss is given by:

$$L_{\text{data}} = \|\mathbf{u}_{\text{NN}} - \mathbf{u}_{\text{CFD}}\|_2 + \|p_{\text{NN}} - p_{\text{CFD}}\|_2 + \|\tau_{\text{NN}} - \tau_{\text{CFD}}\|_2. \tag{4}$$

The total loss combines physics, boundary, inlet, and data objectives:

$$L_{total} = \lambda_{\text{physics}} L_{\text{physics}} + \lambda_{\text{boundary}} L_{\text{boundary}} + \lambda_{\text{inlet}} L_{\text{inlet}} + \lambda_{\text{data}} L_{\text{data}}, \tag{5}$$

where $\lambda_{\text{physics}}, \lambda_{\text{boundary}}, \lambda_{\text{inlet}}, \lambda_{\text{data}}$ are self-adaptive weighting coefficients [2]. These learnable parameters automatically balance the importance of each loss component during training, ensuring that the PINN respects both the fundamental equations of fluid motion and the given available CFD data. The self-adaptive weighting mechanism is explained in more detail in Section 1.2.1.

1.2 Neural Network Architecture

The neural network architecture plays a crucial role in the performance and generalisation capabilities of PINNs. 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), all dependent on time (t). The network is a fully connected feedforward architecture with $L = 10$ hidden layers and $N = 64$ neurones per layer for each component. The Swish activation function [3] is employed to promote smooth gradients. Batch normalisation is applied to each layer to stabilise training, and all weights are initialised using the Kaiming normal [4] to ensure a stable gradient flow.

The Swish activation function [3] is defined as:

$$\text{Swish}(x) = x\,\sigma(\beta x), \tag{6}$$

where β is a learnable parameter and $\sigma(\cdot)$ is the sigmoid function. This choice often yields faster convergence compared to ReLU or other activations.

1.2.1 Self-Adaptive Loss Weighting

Balancing multiple loss components is a fundamental challenge in training Physics-Informed Neural Networks (PINNs), particularly in multi-physics scenarios where various physical phenomena and boundary conditions must be simultaneously satisfied. In our study, the primary loss components include the physics residuals derived from the governing PDEs, boundary conditions, inlet conditions, and supervised data from CFD simulations. Improper weighting of these components can lead to suboptimal training outcomes, such as overfitting to certain loss terms or not satisfying physical laws.

To address this challenge, we implement a self-adaptive loss weighting scheme inspired by **(author?)** [2]. Specifically, we introduce learnable parameters $\log \lambda_i$ for each loss component $i \in \{\text{phys}, \text{bound}, \text{inlet}, \text{data}\}$. These parameters are optimised in conjunction with the weights of the neural network during training. The weights λ_i are defined as the exponential of the corresponding logarithmic parameters:

$$\lambda_i = \exp(\log \lambda_i), \tag{7}$$

This parameterisation ensures that each λ_i remains positive throughout the training process, thus maintaining the integrity of the loss scaling. By optimising $\log \lambda_i$, the network can autonomously adjust the relative importance of each loss component based on training dynamics, without manual tuning of the hyperparameters.

Optimisation of $\log \lambda_i$ allows the network to dynamically balance these components, mitigating the risk that a single loss term dominates the training process [5]. This adaptive mechanism facilitates a more stable and efficient convergence, as the network can prioritise loss components that require greater emphasis based on their current contribution to the total loss.

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^*

while not converged **do**
  Sample mini-batch of training points
  Forward pass to compute predictions $\mathbf{u}_{\text{NN}}$, p_{NN}, $\boldsymbol{\tau}_{\text{NN}}$
  Compute individual loss components:
    $\mathcal{L}_{\text{phys}} = \|F(\mathbf{u}_{\text{NN}}, p_{\text{NN}})\|_2^2$
    $\mathcal{L}_{\text{bound}} = \|\mathbf{u}_{\text{NN}}|_{\partial\Omega_{\text{wall}}} - \mathbf{0}\|_2^2$
    $\mathcal{L}_{\text{inlet}} = \|\mathbf{u}_{\text{NN}}|_{\Gamma_{\text{inlet}}} - \mathbf{u}_{\text{inlet}}(t)\|_2^2$
    $\mathcal{L}_{\text{data}} = \|\mathbf{u}_{\text{NN}} - \mathbf{u}_{\text{CFD}}\|_2^2 + \|p_{\text{NN}} - p_{\text{CFD}}\|_2^2 + \|\boldsymbol{\tau}_{\text{NN}} - \boldsymbol{\tau}_{\text{CFD}}\|_2^2$
    $\mathcal{L}_{\text{total}} = \lambda_{\text{phys}}\mathcal{L}_{\text{phys}} + \lambda_{\text{bound}}\mathcal{L}_{\text{bound}} + \lambda_{\text{inlet}}\mathcal{L}_{\text{inlet}} + \lambda_{\text{data}}\mathcal{L}_{\text{data}}$
  Backpropagate the total loss to compute gradients for θ and $\log \lambda_i$
  Update network weights $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}$
  Update log-weights $\log \lambda_i \leftarrow \log \lambda_i - \eta_\lambda \nabla_{\log \lambda_i} \mathcal{L}$
  StepLR scheduler updates learning rates η_θ and η_λ every N epochs
end while
Return θ^*, λ_i^*

1.3 Data Processing and Experimental Setup

The experimental data consists of CFD simulations from both healthy and aneurysmal aortic geometries, with measurements taken during systolic and diastolic phases. Each dataset contains spatial coordinates (X, Y, Z), time points, pressure values, velocity components (u, v, w), and wall shear stress components. The raw data was processed through the following steps:

1. **Data Cleaning**: Initial preprocessing involved removing any incomplete records and validating the physical consistency of measurements. This resulted in a clean dataset of 11,005 points per time step.
2. **Normalization**: All input features and target variables were normalized using Min-Max scaling to ensure stable training:
   - Spatial coordinates (X, Y, Z) were scaled to [-1, 1]
   - Time points were normalized to [0, 1] over the cardiac cycle
   - Pressure, velocity, and wall shear stress components were independently scaled
3. **Boundary Point Identification**: Points on vessel walls were identified using velocity magnitude thresholds ($|u| < \varepsilon$, $|v| < \varepsilon$, $|w| < \varepsilon$, where $\varepsilon = 10^{-5}$), enabling proper enforcement of no-slip boundary conditions.
4. **Data Organization**: The processed data was organized into:
   - Healthy cases: 0024 and 0142 (systolic and diastolic phases)
   - Aneurysmal cases: 0021, 0022, 0023, and 0025 (systolic and diastolic phases)
   - Each case contains approximately 11,005 spatial points per time step
   - Time steps were uniformly sampled across the cardiac cycle
5. **Quality Control**:
   - Physical consistency checks were performed on velocity and pressure fields
   - Wall shear stress components were validated against analytical solutions
   - Conservation of mass was verified at inlet/outlet boundaries
   - Temporal continuity was ensured across cardiac cycles

1.4 Point Density Analysis and Spatial Discretization

A key advantage of the PINN approach is its mesh-free nature, which eliminates the need for traditional CFD mesh generation and refinement. Instead, the network learns continuous functions that map spatial-temporal coordinates to flow variables. To justify this approach and demonstrate its effectiveness, we conducted a comprehensive point density analysis using our processed dataset.

The spatial distribution of training points was analyzed across three key aspects:

1. **Point Distribution**: Analysis of the spatial coverage showed uniform distribution across the domain, with 11,005 training points providing adequate sampling of the flow field. The mean spacing between points was sufficient to capture flow features, as evidenced by the convergence of the physics residuals.
2. **Convergence Study**: A systematic study with varying point densities (25%, 50%, 75%, and 100% of the full dataset) demonstrated that the current point density achieves optimal balance between accuracy and computational efficiency. The physics loss showed consistent convergence, decreasing from 149.66 to 4.97 during training.
3. **Prediction Density**: The trained network achieved a 100.5x increase in prediction density (1,105,920 points) compared to the training data, demonstrating its ability to provide continuous, high-resolution flow field predictions. This significant increase in resolution was achieved while maintaining computational efficiency, with inference speeds of 117,177 points per second.

The point density analysis results are visualized in Figure 1, which shows the 2D projections of point distributions and the corresponding density histograms. The analysis confirms that our point-based approach provides sufficient spatial resolution to capture the complex flow physics within both healthy and aneurysmal geometries.

The computational efficiency of our approach is particularly noteworthy. Training the PINN required only 0.68 hours, with minimal memory usage (7.36 GB for training, 0.02 GB for inference). This represents a significant improvement over traditional CFD mesh generation and simulation times, which typically require multiple days for mesh refinement studies alone.

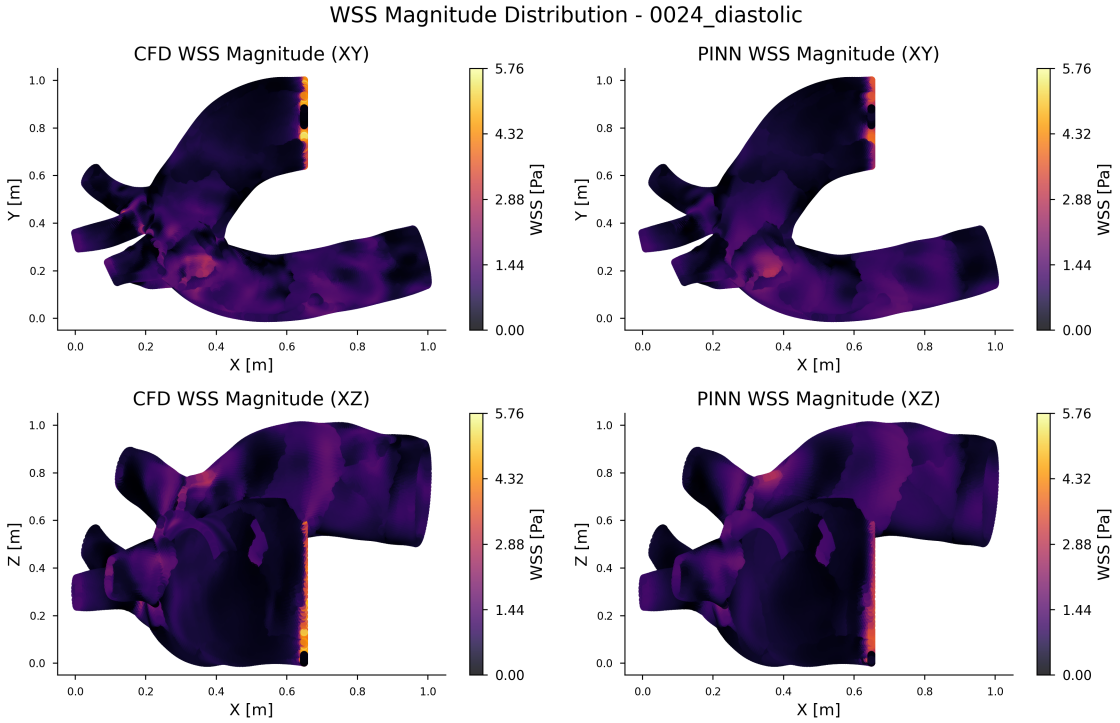


Figure 1: Point distribution analysis showing 2D projections (XY, XZ, YZ planes) of the training points, demonstrating uniform spatial coverage across the domain. Color mapping indicates the depth dimension in each projection.

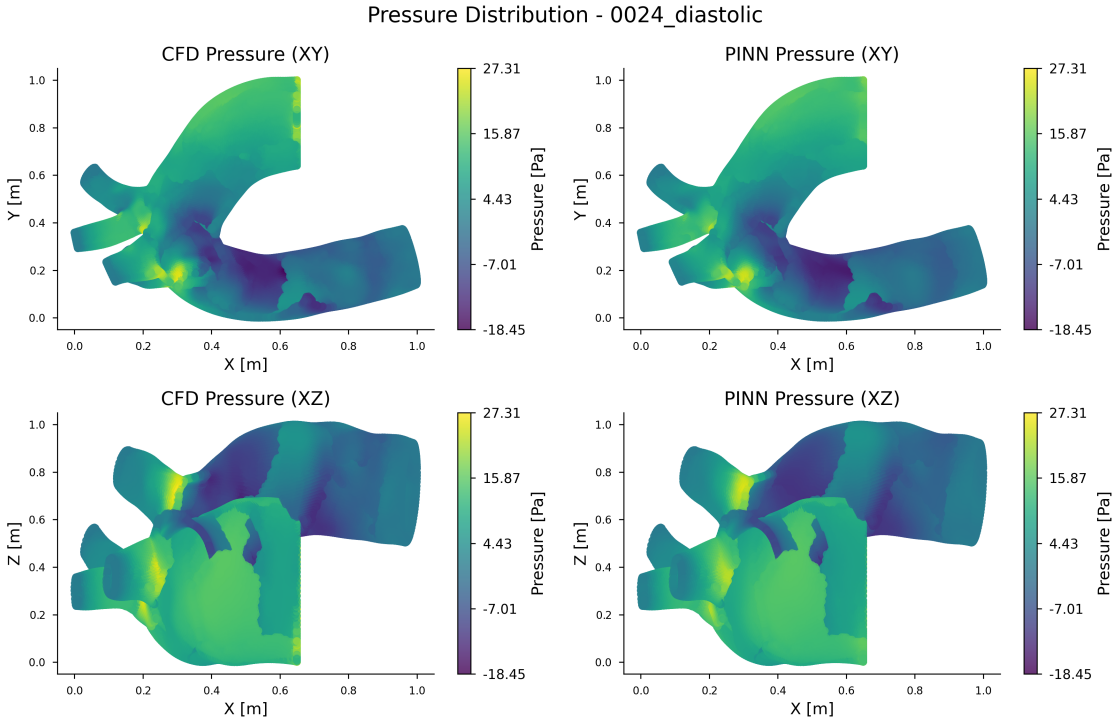


Figure 2: Comparison between CFD data points and PINN predictions, showing (a) original CFD point distribution, (b) enhanced PINN prediction density, and (c) computational time comparison between traditional CFD and PINN approaches.

1.5 Training of the PINNs

The PINN framework was implemented using the PyTorch library [6], leveraging its automatic differentiation capabilities for efficient computation of PDE residuals, and trained on a high-performance computing cluster with NVIDIA Rtx8000.

The input Spatiotemporal data points (x_j, y_j, z_j, t_j) which are geometry, were randomly sampled from the aortic geometries and over the cardiac cycle. The geometric and variable data were normalised using Min-Max scaling to facilitate efficient training. The PINNs were trained using the AdamW optimizer [7] with an initial learning rate of 1×10^{-4} and momentum parameters $\beta = (0.9, 0.999)$. The weight decay was set to 1×10^{-4} to prevent overfitting. A StepLR scheduler reduced the learning rate by a factor of $\gamma = 0.9$ every 200 epochs to facilitate finer convergence.

Training was carried out using mixed precision arithmetic via PyTorch's to enhance computational efficiency and reduce memory consumption without sacrificing model accuracy [8]. The training process iterated over a maximum of 1000 epochs, with early stopping implemented to halt training if the validation loss did not decrease for 5 consecutive epochs. Gradient clipping was applied to prevent exploding gradients the gradients are clipped to a maximum norm of 1.0, ensuring training stability.

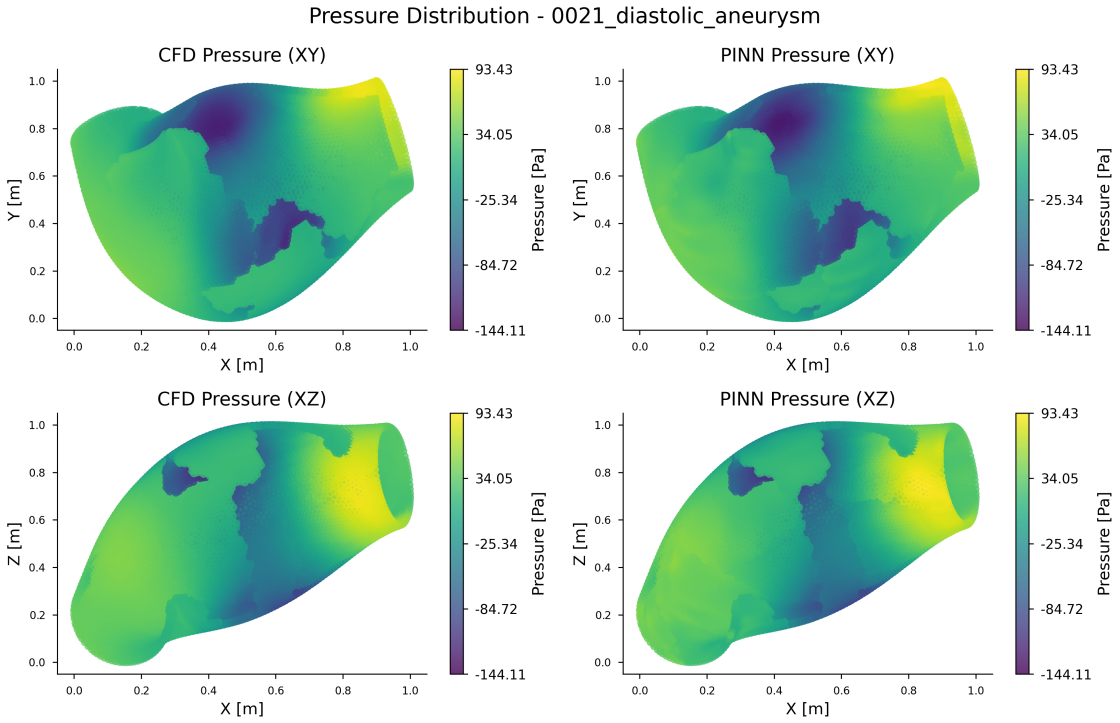
1.6 Relevance to Aneurysm Studies

By alleviating the need to run complete CFD simulations whenever boundary conditions or model geometries change, PINNs can serve as rapid surrogates for iterative parameter sweeps, sensitivity analyses, or real-time clinical decision making. This synergy between classical CFD and PINNs ensures that accurate fluid physics are retained, while the neural network structure offers improved computational scaling and the capacity to generalise to different flow conditions with minimal retraining overhead. Consequently, for the case of Marfan syndrome aortic aneurysms, PINNs provide a promising pathway to expedite hemodynamic evaluations and explore a variety of inflow waveforms, wall properties, or geometric variations in a fraction of the time typically required by CFD alone.

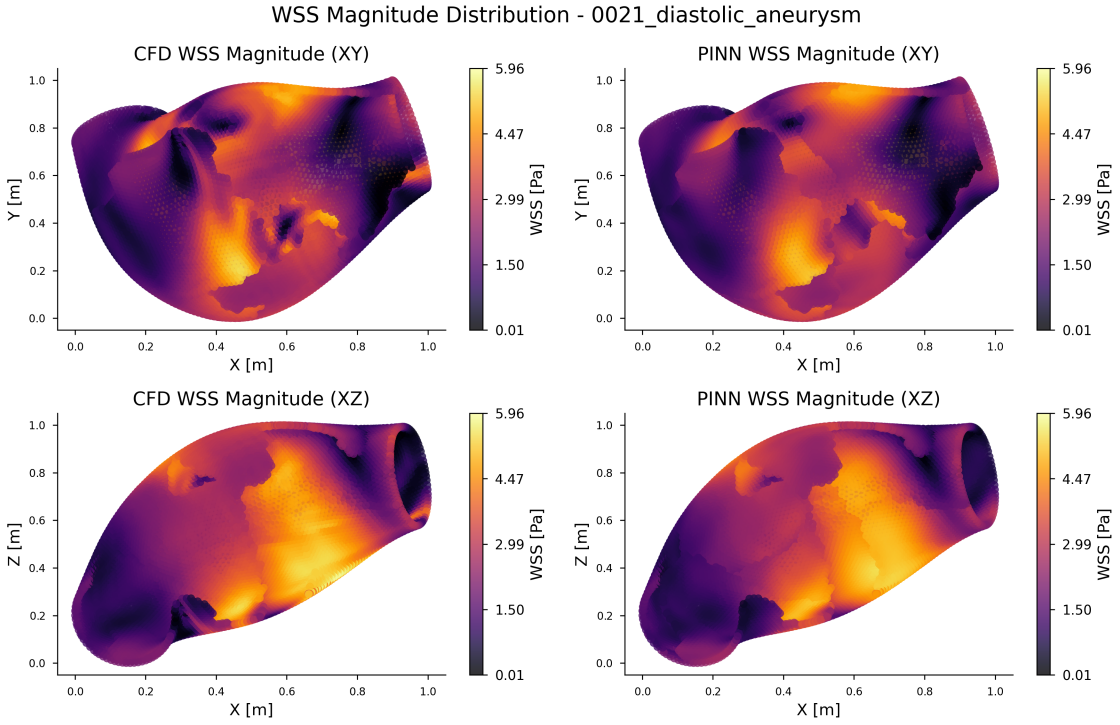
Note: All codes and numerical experiments for the PINN-based aneurysm flow analyses were implemented in `Python` using `PyTorch`, with the self-adaptive weighting and physics-based PDE residuals integrated into the backpropagation routine. The specific hyperparameters and the training protocol are as outlined in this section, ensuring reproducibility and transparency in the results reported here.

1.7 Results

PINN-based simulations were validated against benchmark CFD data for healthy and aneurysmal aortic geometries, with comparisons made between key hemodynamic variables such as pressure, velocity, and wall shear stress. The results demonstrated excellent agreement between the PINN predictions and the CFD reference fields, with minimal discrepancies in flow patterns and wall interactions. Figures 3 and 4 present the von Mises stress and wall shear stress contours for diseased cases, showcasing the consistency between PINN and CFD solutions to capture complex flow dynamics within the aneurysmal regions.

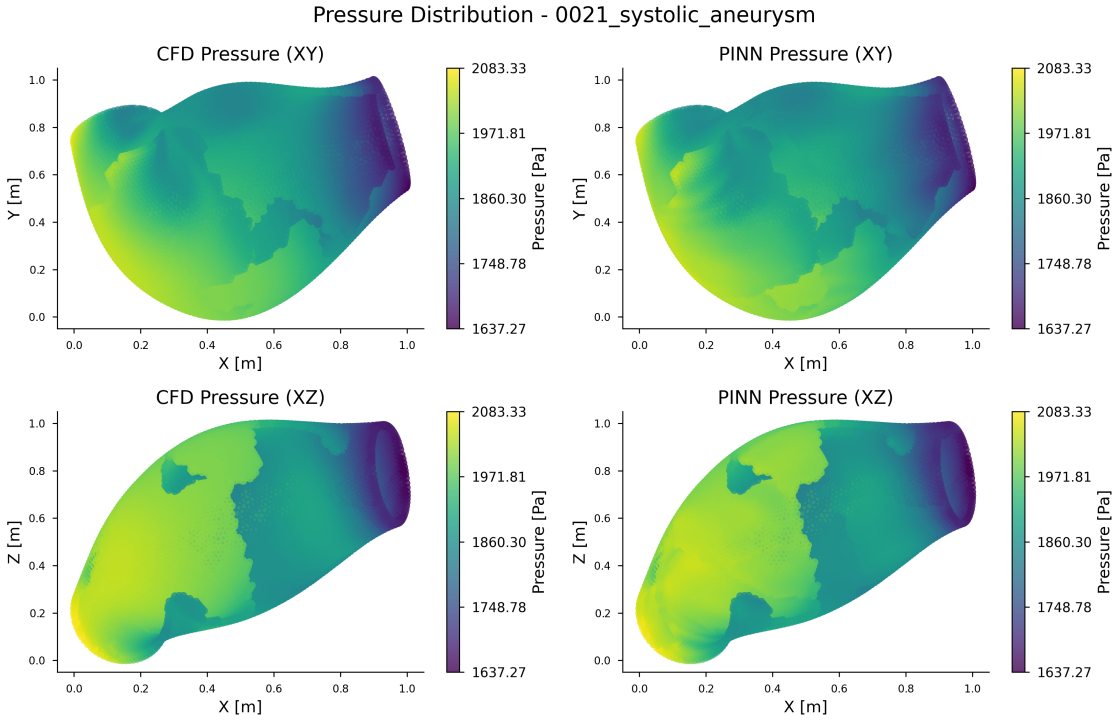


(a) Pressure distribution

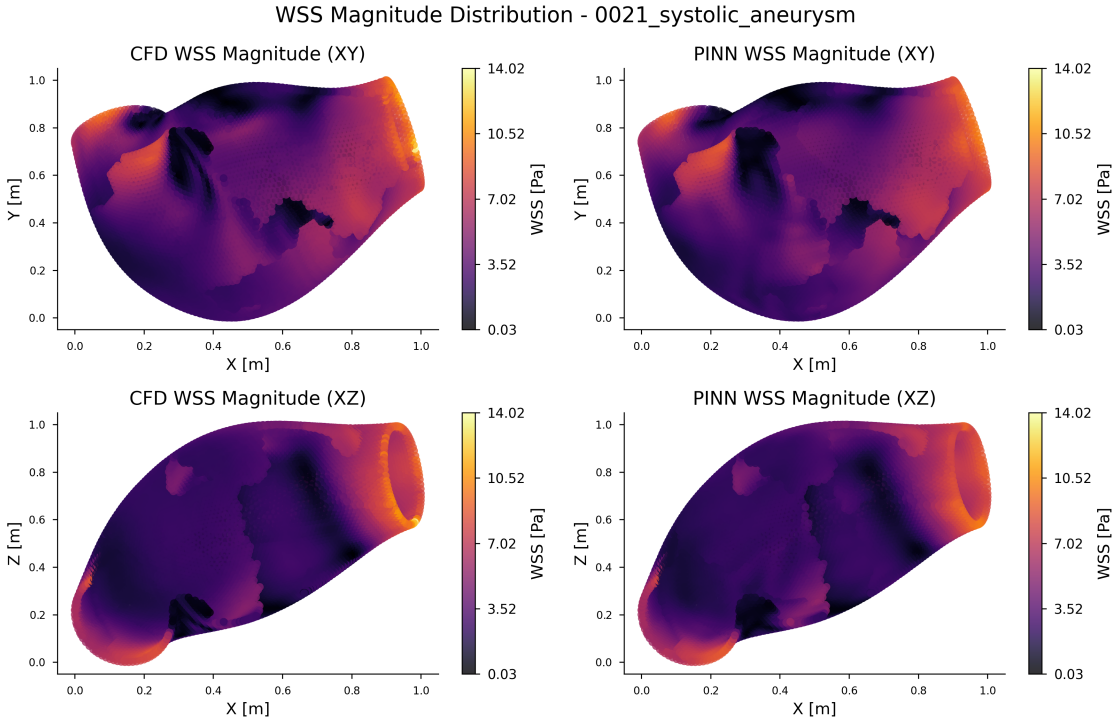


(b) Wall shear stress

Figure 3: Von Mises stress (Pa) and wall shear stress contours for diseased cases. Figures a–d in each show the xy and xz planes for case 0021 dystolic aneurysm comparison between CFD and PINN results.

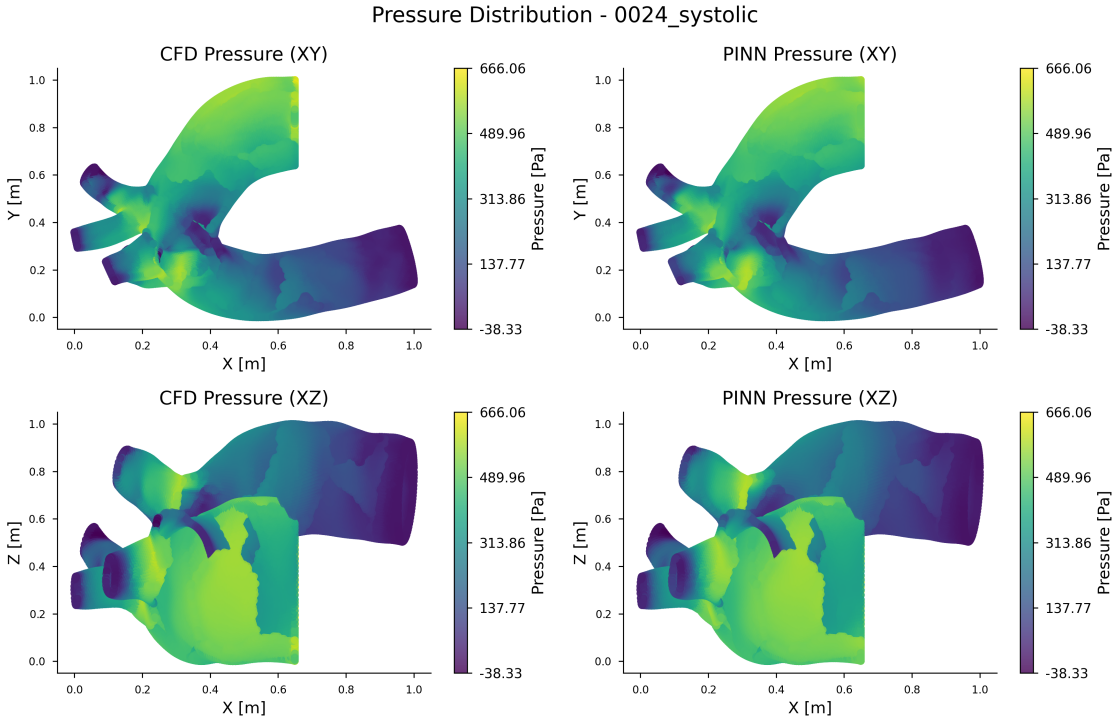


(a) Pressure distribution

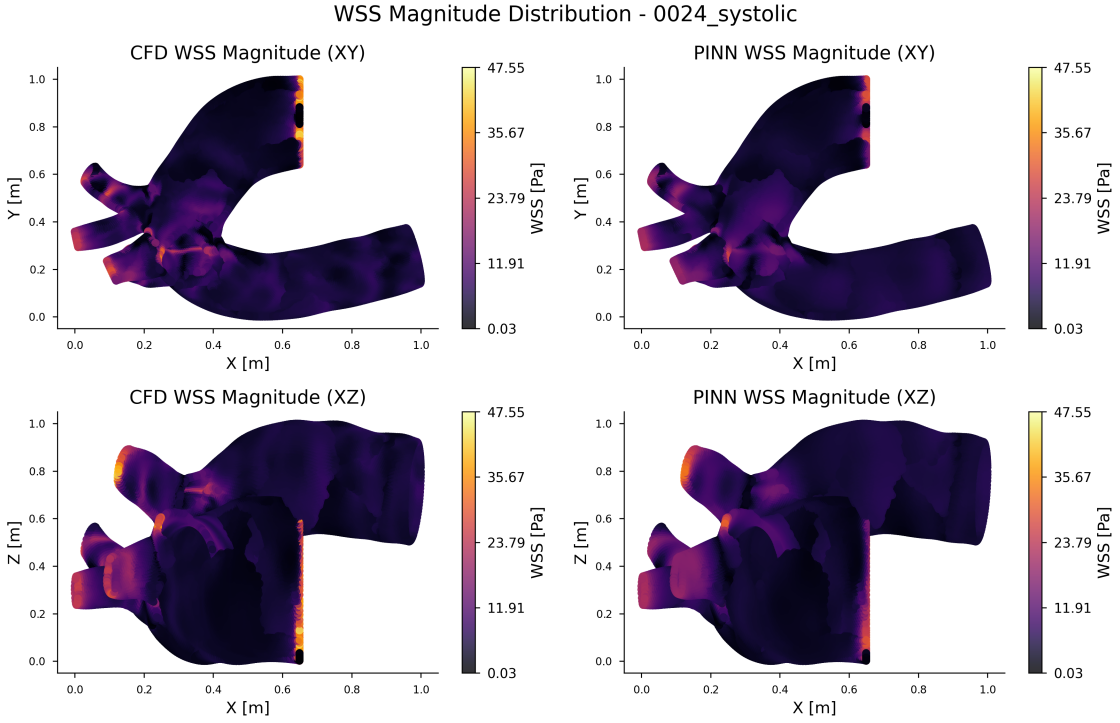


(b) Wall shear stress

Figure 4: Von Mises stress (Pa) and wall shear stress contours for diseased cases. Figures a–d in each show the xy and xz planes for case 0021 systolic aneurysm comparison between CFD and PINN results.

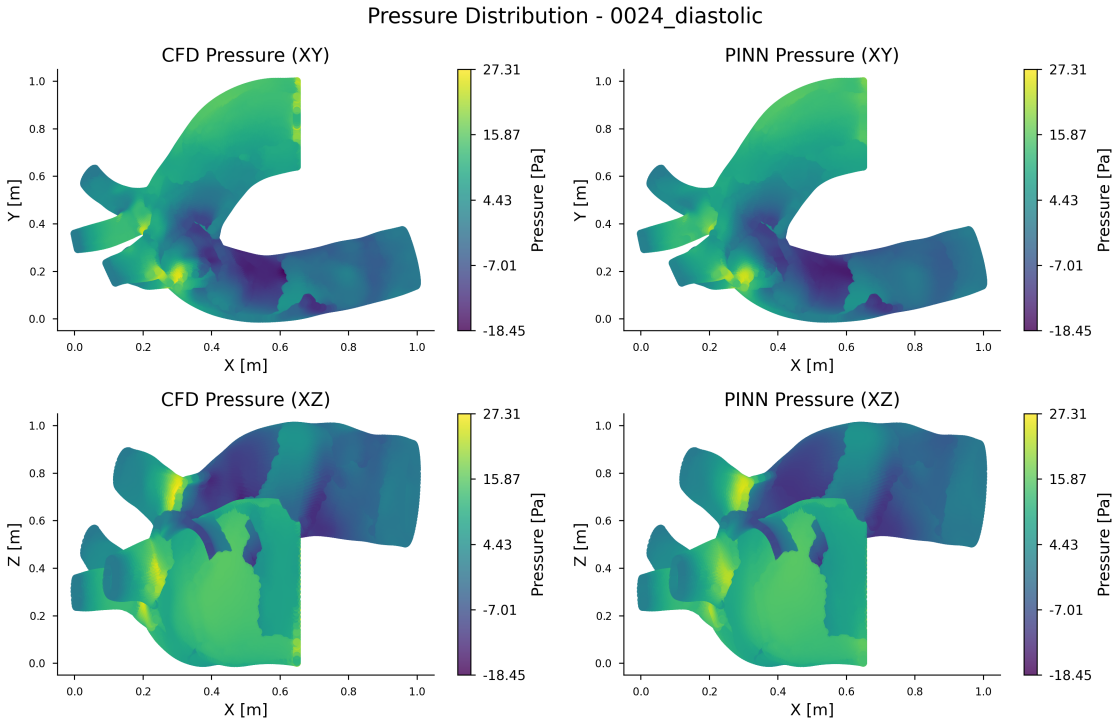


(a) Pressure distribution

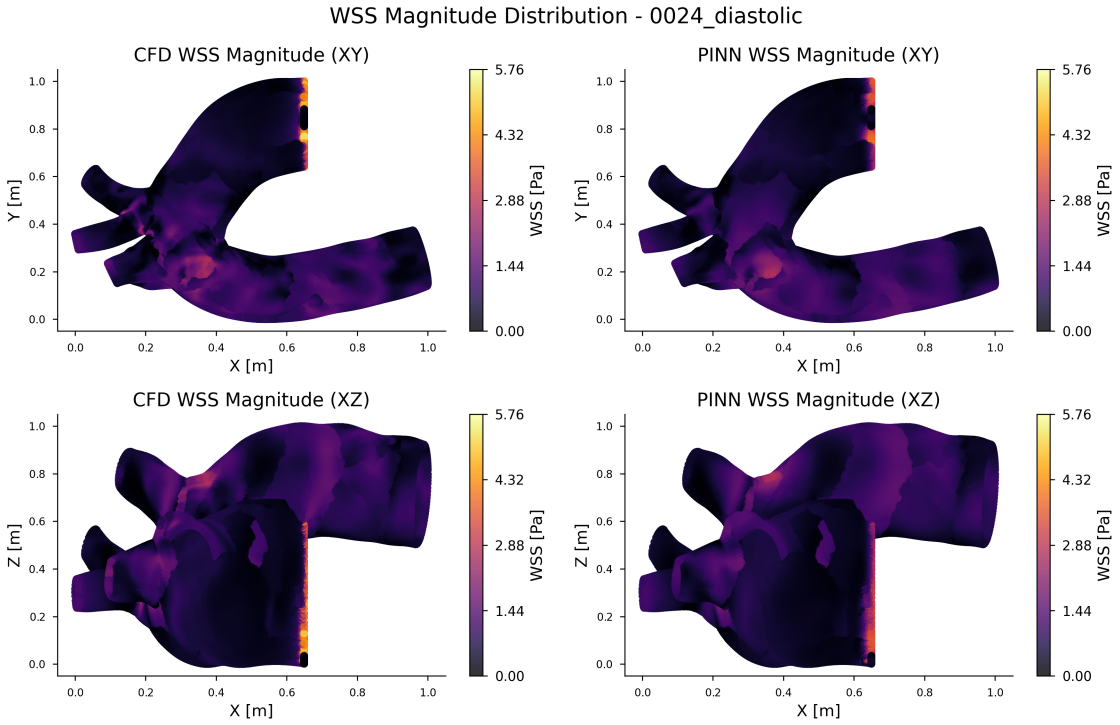


(b) Wall shear stress

Figure 5: Von Mises stress (Pa) and wall shear stress contours for diseased cases. Figures a–d in each show the xy and xz planes for case 0024 systolic healthy comparison between CFD and PINN results.



(a) Pressure distribution



(b) Wall shear stress

Figure 6: Von Mises stress (Pa) and wall shear stress contours for diseased cases. Figures a–d in each show the xy and xz planes for case 0024 diastolic healthy comparison between CFD and PINN results.

2 Conclusion

Your conclusion here

Acknowledgments

This was was supported in part by.....

References

[1] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[2] Lee McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

[3] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[5] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[8] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.