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

Michael Ajao-Olarinoye[1]

[1]Center 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, as outlined in Section 2.4, offer precise insights into arterial hemodynamics and are central to modeling fluid dynamics in complex vascular geometries. However, these simulations can be both computationally expensive and time-consuming, especially when dealing with patient-specific anatomies or conducting 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 Raissi et al. [1], have emerged as a powerful alternative that combines the strengths of traditional deep learning with established physics-based modeling. 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 labeled 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 labeled 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 overheads and accelerating research pipelines.

In this study, we combine PINNs with traditional CFD data to estimate core hemodynamic variables such as pressure, velocity components, and wall shear stress (WSS) across 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 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.

1.2 Formulation of Physics-Informed Neural Networks

Modelling pulsatile flow in arterial segments necessitates 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:

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

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. 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 wall shear stress (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 three wall shear stress (WSS) components

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

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

Let $F(\cdot)$ represent the non-linear operator corresponding to the Navier–Stokes system. The *physics residual* loss term evaluates how closely the PINN's predictions 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's predictions from the specified boundary conditions, thus reinforcing physically realistic flow fields at the walls and inflow region.

The CFD data (e.g. 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 futher in Section 1.3.1.

1.3 Neural Network Architecture

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 time dependant (t). Each network is a feed-forward, fully connected architecture with $L = 10$ hidden layers and $N = 64$ neurons per layer. The Swish activation function [3] is employed to promote smooth gradients. batch normalization is applied to each layer to stabilize training, and all weights are initialized using Kaiming normal [4] to ensure 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.

1.3.1 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 partial differential equations (PDEs), boundary conditions, inlet condi-

tions, and supervised data from Computational Fluid Dynamics (CFD) simulations. Improper weighting of these components can lead to suboptimal training outcomes, such as overfitting to certain loss terms or failure to adequately satisfy physical laws.

To address this challenge, we implement a self-adaptive loss weighting scheme inspired by McClenny and Braga-Neto [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 optimized alongside the neural network weights during training. The weights λ_i are defined as the exponential of the corresponding log-parameters:

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

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

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

1.4 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 the geometric were randomly sampled from the aortic geometries and over the cardiac cycle. The geometric and variables data were normalized 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.

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:
    $\mathcal{L}_{\text{phys}} = \|F(\mathbf{u}_{\text{NN}}, p_{\text{NN}})\|_2^2$
    $\mathcal{L}_{\text{bound}} = \|\mathbf{u}_{\text{NN}}|_{\partial\Omega_{\text{wall}}} - \mathbf{0}\|_2^2$
    $\mathcal{L}_{\text{inlet}} = \|\mathbf{u}_{\text{NN}}|_{\Gamma_{\text{inlet}}} - \mathbf{u}_{\text{inlet}}(t)\|_2^2$
    $\mathcal{L}_{\text{data}} = \|\mathbf{u}_{\text{NN}} - \mathbf{u}_{\text{CFD}}\|_2^2 + \|p_{\text{NN}} - p_{\text{CFD}}\|_2^2 + \|\boldsymbol{\tau}_{\text{NN}} - \boldsymbol{\tau}_{\text{CFD}}\|_2^2$
  Compute total loss using Equation (**??**)
  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 $\log \lambda_i \leftarrow \log \lambda_i - \eta_\lambda \nabla_{\log \lambda_i} \mathcal{L}$
  Apply any learning rate schedules or regularization as needed
end while
Return θ^*, λ_i^*

Training was conducted using mixed-precision arithmetic via PyTorch's 'torch.cuda.amp' 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.

1.5 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 for expediting haemodynamic evaluations and exploring a range 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 training protocol are as outlined in this section, ensuring reproducibility and transparency in the results reported herein.

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] Levi 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. In *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, 2019.

[7] I Loshchilov. 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.