Spectral features for Neural-initalized-Newton Solvers in Cardiovascular CFD simulations

Maximilian Pochapski
University of Waterloo
mpochaps@uwaterloo.ca

***Abstract*—Computational Fluid Dynamics (CFD) simulations of cardiovascular systems hold promise for personalized clinical care, but are hindered by their high computational cost. Neural-initialized-Newton (NiN) methods combine deep learning with iterative solvers to accelerate convergence, but existing approaches lack features that capture long-range physical couplings critical to cardiovascular flow. In this work, I propose a modified MeshGraphNets architecture augmented with spectral features derived from the finite element stiffness matrix K, enabling the model to encode global boundary condition information absent from standard message-passing. The modified Spectral MGN achieves a test RMSE of 1.082, with strong directional alignment between predicted and true solution deltas.**

I. Introduction

A. Motivation

Recent advances in computing and simulation software have boosted the popularity of Computational Fluid Dynamics (CFD) in the study of cardiovascular systems. CFD has already been established as a useful tool in the private sector, namely in aerospace, automotive, and civil engineering [1]. With modern medical equipment, accurate modeling of a patients cardiovascular system is now possible; with many commercial companies overseas such as United-Imaging Healthcare and Shanghai Mircroport Endovascular Medtech (EMT) [2] already leveraging CFD to personalize patient care and improve post-surgery outcomes. CFD simulations have been used to non invasively assess the severity of coronary artery aneurysms [3], treat congenital heart disease [4], and engineer medical devices [5]. However, wide-spread implementation in clinical practice is hindered by the high computational cost of simulation. Most CFD solvers use finite methods such as the Finite Element Method (FEM) or Finite Volume Method (FVM) to discretize the physics in space and time. The more elements in a simulation, the more accurate the solution becomes. However, introducing new elements increases the computational cost on the order of $O(n^2)$. New methods and accelerators are necessary for real-world applications in clinical practice. Two promising approaches are Reduced Order Models (ROMs) and Graph Neural Networks (GNNs).

B. Related Works

Clinicians are most interested in calculating the Wall-Shear-Stress (WSS) and fluid velocity within cardiovascular vessels. To this end, future optimized models must be able to quickly and accurately estimate these values from patient-specific data.

ROMs reformulate the full 3D formulation of the Navier-Stokes equation into a lower number of dimensions (one- and zero-dimensions); exchanging accuracy for running at a fraction of the computational cost. A recent study done using 72 models from the Vascular Model Repository compared zero and one-dimensional simulations to their 3D reference simulations [6]. The authors observed average relative errors of 2.1% and 1.8% for pressure and 3.9% and 3.4% on the flow rate (for the zero and one dimensional models, respectively).

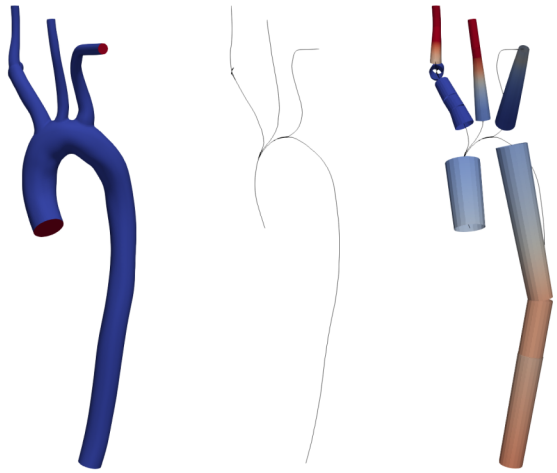

Fig. 1. (Left) the original 3D model of the Aorta used for simulation; areas of the mesh responsible for boundary conditions are highlighted in red. (Center) The extracted centerlines used for generating the segments in the 1D model. (Right) projections of the WSS onto the segments at $t = 0.38$

One-dimensional models are derived by integrating the 3D Navier Stokes equations over the vessel cross-section to reduce the equations to a single spatial variable [7]. Arterial trees are approximated as connections and junctions of the centerlines of vessel segments. Variables such as pressure, flow rate and vessel wall displacement are functions of the axial component only. The advantage of 1D models is their capacity to capture wave propagation phenomena due to the interaction of flow with elastic vessel walls.

Zero-dimensional (lumped-parameter network models) simulations are often used to approximate boundary conditions which are assumed to have a linear relationship with the rest of the model [8].

Recently, interest in deriving new ROMs using data-driven Graph Neural Networks (GNNs) has been gaining traction. GNNs are uniquely suited to accelerating finite-method analysis due to their message-passing architecture. MeshGraphNets (MGN) [9] was specifically designed for predicting mesh-based physics simulations. Most recently, MeshGraphNets have been used to derive a 1D surrogate ROM for cardiovascular simulation [10].

C. Problem Definition

While deep-learning (DL) models promise faster inference than standard iterative solvers, they are prone to hallucinations. GNNs suffer from a failure mode in the message passing step, where physical information propagates only in a local neighborhood; which is detrimental to accurately modelling the physics when simulating elliptical PDEs with incompressible flows, and propagating changes to the boundary conditions. Meanwhile, popular iterative solvers used in CFD, such as Newton methods, require a "good" initialization in order to guarantee convergence, leading to long simulation times and increased sensitivity to solver parameters. Classical solvers are also one-time use in nature. Their increased accuracy is limited by heavy computation that has to be repeated even for small perturbations in the input parameters.

Therefore, Neural-initialized-Newton (NiN) methods were invented to combine the best of deep-learning and iterative solvers [11]. At the start of the iteration loop, a DL model guesses a better initial step based on the current system parameters, and the Newton solver then computes a physically-accurate solution to the system.

In this paper, I consider an NiN approach using a modified MeshGraphNets model to initialize a modified Newton Raphson solver implemented in the open-source SimVascular software package. In summary, my contributions are:

1) I engineered features which accurately label the node type (Area, Flowrate, or Junction), captured long-range features and physical coupling terms using the eigenpairs of the stiffness matrix K.
2) I developed a modified version of the MeshGraphNet model capable of extracting high-frequency data from the spectral features using SIRENs.
3) I assessed the capability of my Spectral MGN to guess a modified initial state relative to the testing set.

II. Methodology

A. The Governing Equations

In this paper I used the 1D solver provided with the Simvascular open-source package [12]. The governing partial differential equations for the 1D model are:

$$\frac{\partial S}{\partial t} + \frac{\partial Q}{\partial z} = -\psi \tag{1}$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial z}((1+\delta)\frac{Q^2}{S}) + \frac{S}{\rho}\frac{\partial p}{\partial z} = Sf + N\frac{Q}{S} + \nu^2\frac{\partial^2 Q}{\partial z^2} \tag{2}$$

Where S is the cross-section area, Q is the volumetric flow rate, ρ is the density of the fluid (which is constant for our simulations), f is the external force, ν is the kinematic viscoscity, and ψ is an outflow function (zero for impermeable vessels). δ and N are defined by the choice of profile functions for the velocity of over the cross-section. Pressure is related to the luminal area S using the constitutive equation [13]:

$$\hat{p}(S(z,t),z,t) = p^0(z) + \frac{4Eh}{3r^0(z)}(1 - \sqrt{\frac{S^0(z)}{S(z,t)}}) \tag{3}$$

Further details on the strong form of the governing equations are covered in [14], [15].

B. The Finite Element Method

The 1D Simvascular solver implements a stabilized space-time finite element method based on the discontinious Galerkin method in time. It was chosen because it has been shown to yield stable, time-accurate solutions for advective-diffusive systems provided by fluid mechanics.

The strong form of the governing equations are too strict for iterative methods. To find a more pliable formulation, we derive the weak form of the equations by rewriting the system of PDEs in a quasi-linear form:

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{U}}{\partial z} - K\frac{\partial^2 \mathbf{U}}{\partial z^2} = G \tag{4}$$

Where

$$\mathbf{U} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} S \\ Q \end{bmatrix} \tag{5}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -(1+\delta)\left(\frac{U_2}{U_1}\right)^2 + \frac{U_1}{\rho}\frac{\partial \hat{p}}{\partial S} & (1+\delta)\frac{2U_2}{U_1} \end{bmatrix} \tag{6}$$

$$\mathbf{K} = \begin{bmatrix} 0 & 0 \\ 0 & \nu \end{bmatrix} \tag{7}$$

$$\mathbf{G} = \begin{bmatrix} -\psi \\ Sf + N\frac{U_2}{U_1} - \frac{U_1}{\rho}\frac{\partial \hat{p}}{\partial z} \end{bmatrix} \tag{8}$$

Therefore, the weak formulation of the initial boundary value problem is given as: Find $\mathbf{U}$ such that for all $\mathbf{W} = [W_1, W_2]^T$;

$$\begin{aligned} \int_0^T \int_0^L (-\mathbf{W}_{,t}^T\mathbf{U} + \mathbf{W}^T\mathbf{A}\mathbf{U}_{,z} + \mathbf{W}_{,z}^T\mathbf{K}\mathbf{U}_{,z} - \mathbf{W}^T\mathbf{G})\, dzdt \\ + \int_0^L \mathbf{W}^T(z,T)\mathbf{U}(z,t)\, dz \\ - \int_0^L \mathbf{W}^T(z,0)\mathbf{U}^0(z)\, dz = 0 \end{aligned} \tag{9}$$

After discretizing over space and making a piecewise constant approximation in time, the problem simplifies to:

$$\begin{aligned} \Delta t_s \int_0^L (\mathbf{W}^T\mathbf{A}\mathbf{U}_{,z}^{n+1} + \mathbf{W}_{,z}^T\mathbf{K}\mathbf{U}_{,z}^{n+1} - \mathbf{W}^T\mathbf{G} \\ + \mathbf{W}^T(\mathbf{U}^{n+1} - \mathbf{U}^n))\, dz = 0 \end{aligned} \tag{10}$$

Assuming we have n nodal points, we express the vector fields as:

$$\mathbf{W} = \sum_{A=1}^{n} N_A \mathbf{C}_A \tag{11}$$

$$\mathbf{U}^{n+1} = \sum_{B=1}^{n} N_B \mathbf{U}_B^{n+1} \tag{12}$$

Where $N_A(z)$, $N_B(z)$ are the usual piecewise-linear shape functions (the finite elements for our problem). The 1D solver uses a modified Newton-Raphson technique to solve the nonlinear system. The modification assumes a consistent tangent in $\mathbf{A}$, $\mathbf{C}$ and τ with respect to $\mathbf{U}$ in the calculation of the Jacobian. Therefore the iterative step is facilitated by:

$$\sum_{C} \tilde{\mathbf{K}}_{AC}^{n+1,k} \delta \mathbf{U}_C^{n+1,k+1} = R_A^{n,k} \tag{13}$$

$$\mathbf{U}_C^{n+1,k+1} = \mathbf{U}_C^{n+1,k} + \delta \mathbf{U}_C^{n+1,k+1} \tag{14}$$

C. Branch Points

Pressure continuity and conservation of mass are enforced using Lagrange multipliers (Λ) at branch points. For a branch point with m inlets and n outlets, the continuity equations can be formulated as a potential function Z:

$$Z = \Lambda_Q [\sum_{C=1}^{m} Q_C^{in} - \sum_{C=1}^{n} Q_C^{out}] + \sum_{C=2}^{m} [\Lambda_{P(c-1)}(p_C^{in} - p_l^{in})] + \sum_{C=1}^{n} [\Lambda_{p(C-1+m)}(p_C^{out} - p_l^{in})] \tag{15}$$

Besides the primary variable vector $\mathbf{U}$, an additional $m+n$ entries are concatenated as additional unknowns of the nonlinear system. The new arrays are added to the global stiffness matrix $\mathbf{K}$ and Residual $\mathbf{R}$ as outlined in [14].

Boundary conditions are applied directly to $\mathbf{K}$ on every iteration and are also detailed in [14].

D. Spectral Feature Extraction

Fourier Neural Operators (FNOs) [16] and Spectral Graph Neural Networks (SGNNs) [17] have become popular in Scientific Machine Learning (SciML). By computing the Fourier Transform, Spectral methods decompose the "frequencies" of the training data, enabling the model to contextualize local features into the grand physics of the problem.

Traditionally spectral features are encoded using the Graph Laplacian derived from the Adjacency matrix. However, I want to engineer features only using data that the solver had access to, and encode the physics of $\mathbf{K}$ for S, Q *and* their cross-coupling terms between node types: an important datapoint when solving nonlinear systems. Therefore, I decided to compute the spectral features of my problem from the stiffness matrix $\mathbf{K}$ directly.

Since $\mathbf{K}$ is not a symmetric matrix, it produces complex eigenpairs. In the context of waves the real components of the eigenvectors encode the magnitude of dissapative effects and the complex components model phase shifts at the node.

E. Feature Engineering

Each node has the following feature vector:

$$\mathbf{v}_n = [U_n, R_n, K_{(n,n)}, \Re(V_\lambda)_n, \Im(V_\lambda)_n, \alpha_n] \tag{16}$$

Where $K_{(n,n)}$, U_n, and R_n are the stiffness, solution and residual values at the n-th node before solving. $\Re(V_\lambda)_n$ and $\Im(V_\lambda)_n$ are the real and imaginary parts of the n-th components of the eigenvectors corresponding to the smallest-k eigenvalues of the stiffness matrix $\mathbf{K}$. The smallest-k eigenvalues correspond to the "low-frequencies" of the problem and transmit global information such as changes in the boundary conditions. α_n is the one-hot encoded vector for the node type: Luminal Area (S), Flow (Q) or Lagrange Multiplier (Λ).

Every edge has the following feature vector:

$$\mathbf{e}_n = [\mathbf{K}_{n,j}]$$

Where $\mathbf{K}_{n,j}$ is the n-th row of the Stiffness matrix, which contains the coupling coefficients between nodes across all three node types.

F. Forward step of the Spectral Graph Neural Network

I used the MeshGraphNets architecture as detailed in [9] [10]. The forward pass consists of the following three stages:

1) Encode: Latent embeddings are created from the input features. Particularly, I compute the node latent representations as $\mathbf{v}_n' = \mathbf{s}_{en}(v_n) \in \mathbb{R}^{n_l}$ and $\mathbf{w_{ij}}' = \mathbf{s}_{ee}(e_{ij}) \in \mathbb{R}^{n_l}$ for all nodes v_n and edges e_{ij} respectively. However unlike MeshGraphNets, I used a SIREN for my embedding function $\mathbf{s_e}$. SIRENs are Multi-layer Perceptions (MLPs) which use a sine as their nonlinear activation function after the affine transformation. The advantage is twofold, their second derivative is not zero, and therefore can capture information from higher-order derivatives (in contrast, the 2nd-order derivatives of ReLU functions is zero), and they can capture higher-frequency information from the complex eigenvector features [18]. I chose the latent space $n_l = 16$ to be the same dimension for both nodes and edges.
2) Process: The process stage is performed L times in two phases; computing new edge features, and then deriving the new node features from the aggregated edge features:

$$\mathbf{e}_{ij}^{(l)} = f_{pe}^{(l)}(\mathbf{e}_{ij}^{(l-1)}, v_i^{(l-1)}, v_j^{(l-1)}) \tag{17}$$

$$v_n^{(l)} = f_{pn}^{(l)}(v_n^{(l-1)}, \sum_{e_{nj}} \mathbf{e}_{ij}^{(l)}) \tag{18}$$

   Where $1 \leq l \leq L$ is the iteration number and $f_p^{(l)}$ are MLPs with ReLU activation functions.
3) Decode: Finally, all the node features are transformed from a latent space to the output space using a MLP $\delta U_n = f_{dn}(v_n) \in \mathbb{R}$. The desired output vector is the delta of the solution vector $\delta \mathbf{U}$ which is used to compute the initial guess for the solver $\mathbf{U}_g = \mathbf{U}_0 + \delta \mathbf{U}$.

G. Training

The network was trained to predict the difference between the final and inital solution vector states $\delta\mathbf{U} = \mathbf{U}_f - \mathbf{U}_0$. This was done so the model attempts to initialize the solver in the local neighborhood of the real solution. The loss function takes the form of:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} ||(\delta\mathbf{U}_i - \Psi(\Theta))^2|| \tag{19}$$

Where N is the batch size, Ψ is the modified MGN and Θ is a vector containing both the node and edge features for the MGN.

H. Dataset

I considered 5 healthy Aorta models selected from the Vascular Model Repository (VMR). The models were constructed from healthy patients of various ages and genders. I chose to use the Aorta because it presents features such as many junctions which are considered challenging for physics-based models. For each of the 5 original one dimensional simulations I created variations via random perturbations of the original boundary conditions. In particular, I multiplied the parameters of the inlets and outlets by a factor uniformly distributed in the range [0.8, 1.2]. Each simulation simulated 1000 steps at a time step of 0.001 seconds, simulating one full second of the heart beating. Each $\delta\mathbf{U}$ corresponds to one time step solved by the Newton solver. For the inlet I used a pulsatile boundary condition, meanwhile the outlets used a RCR (Windkessel) boundary condition. In total, 198 1D simulations were run. The dataset used a train-test-split of 0.7.

III. Results

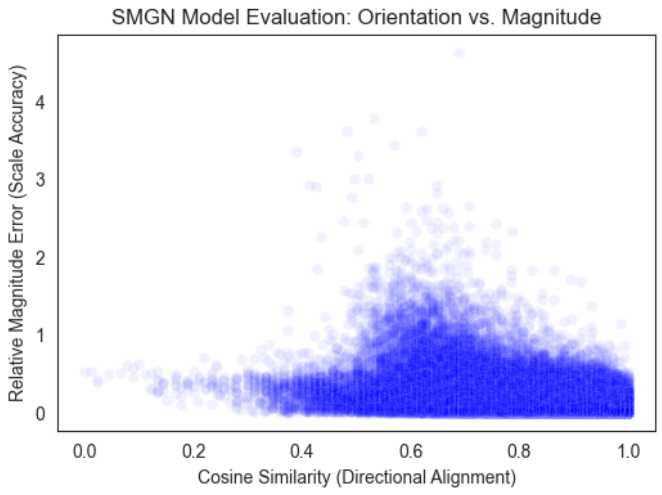

Fig. 2. 138839 datapoints are plotted. Each represents the error in the direction and scale of $\delta\mathbf{U}$ for a single time step. Based on the bottom-right grouping, The model is more likely to produce a correct answer if its prediction is in the right direction.

The model achieved a RMSE of ≈ 1.082 on the test set after 200 epochs of training. Since $\delta\mathbf{U}$ is a vector value, the magnitude error and cosine similarity were also analyzed in Fig.2. The model shows a strong directional alignment between predicted and true solution deltas, demonstrating the model's ability to consistently estimate both the direction and magnitude of the required Newton initialization step when the predicted direction is already correct. The variance between the values spike between [0.4, 0.8], indicating the existence of edge-cases where the model performs poorly. In summary, if the model gets the direction right, it also gets the magnitude right.

IV. Conclusion

In this work I introduced the architecture for a Neural-initialized-Newton solver for problems in reduced-order cardiovascular computational fluid dynamics. The modifications to the original MeshGraphNets architecture were justified and the model was trained on 137 simulations using spectral features derived from the stiffness matrix $\mathbf{K}$ to capture the long-range physical couplings absent from standard GNN message passing.

Future work entails identifying outliers in the training data, especially in the high-variance regions of [0.4, 0.8], and optimizing the training scheme. Next, full integration of the model into the Simvascular 1D solver will be necessary to measure the performance of the model in a real-world application.

References

[1] R. Raman, R. Kumar, and R. K. Singh, "A review on applications of computational fluid dynamics," *International Journal of Engineering Research and General Science*, vol. 6, no. 4, pp. 135–143, 2018.

[2] C. X. Tang, C. F. Liu, X. JCMY, J. B. Schoepf, C. Tesche, R. R. Bayer, H. T. Hudson, L. J. Zhang, and G. M. Lu, "Ct ffr for ischemia-specific cad with a new computational fluid dynamics algorithm: A chinese multicenter study," *JACC: Cardiovascular Imaging*, vol. 13, no. 4, pp. 980–990, 2020.

[3] S. Sankaran, M. E. Moghadam, A. M. Kahn, E. E. Tseng, J. M. Guccione, and A. L. Marsden, "Patient-specific multiscale modeling of blood flow for coronary artery bypass graft surgery," *Annals of Biomedical Engineering*, vol. 40, no. 10, pp. 2228–2242, 2012.

[4] A. L. Marsden, A. J. Bernstein, V. M. Reddy, S. C. Shadden, R. L. Spilker, F. P. Chan, C. A. Taylor, and J. A. Feinstein, "Evaluation of a novel y-shaped extracardiac fontan baffle using computational fluid dynamics," *The Journal of Thoracic and Cardiovascular Surgery*, vol. 137, no. 2, pp. 394–403, 2009.

[5] T. J. Gundert, A. L. Marsden, W. Yang, and J. F. LaDisa Jr, "Optimization of cardiovascular stent design using computational fluid dynamics," *Journal of Biomechanical Engineering*, vol. 134, no. 1, p. 011002, 2012.

[6] N. M. Wilson, A. K. Ortiz, and A. B. Johnson, "The vascular model repository: A public resource of medical imaging data and blood flow simulation results," *Journal of Medical Devices*, vol. 7, no. 4, p. 040923, 2013.

[7] M. R. Pfaller, J. Pham, A. Verma, L. Pegolotti, N. M. Wilson, D. W. Parker, W. Yang, and A. L. Marsden, "Automated generation of 0d and 1d reduced-order models of patient-specific blood flow," *International Journal for Numerical Methods in Biomedical Engineering*, vol. 38, no. 10, p. e3639, 2022.

[8] M. J. Grote and J. B. Keller, "Exact nonreflecting boundary conditions for the time dependent wave equation," *SIAM Journal on Applied Mathematics*, vol. 55, no. 2, pp. 280–297, 1995.

[9] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning mesh-based simulation with graph networks," deepMind, London, UK.

[10] L. Pegolotti, M. R. Pfaller, N. L. Rubio, K. Ding, R. B. Brufau, E. Darve, and A. L. Marsden, "Learning reduced-order models for cardiovascular simulations with graph neural networks," preprint. Preliminary work.

[11] K. Taghikhani, Y. Yamazaki, J. P. Varghese, M. Apel, R. N. Asl, and S. Rezaei, "Neural-initialized newton: Accelerating nonlinear finite elements via operator learning," 2025.
[12] A. Updegrove, N. M. Wilson, J. Merkow, H. Lan, A. L. Marsden, and S. C. Shadden, "Simvascular: An open source pipeline for cardiovascular simulation," *Annals of Biomedical Engineering*, vol. 45, no. 3, pp. 525–541, 2017.
[13] M. S. Olufsen, "Structured tree outflow condition for blood flow in larger systemic arteries," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 276, no. 1, pp. H257–H268, 1999.
[14] J. Wan, B. Steele, S. A. Spicer, S. Strohband, G. R. Feijóo, T. J. Hughes, and C. A. Taylor, "A one-dimensional finite element method for simulation-based medical planning for cardiovascular disease," pp. 195–206, 2002.
[15] C. A. Taylor, M. T. Draney, J. P. Ku, D. Parker, B. N. Steele, K. C. Wang, and C. K. Zarins, "Predictive medicine: computational techniques in therapeutic decision-making," pp. 231–247, 1999.
[16] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://openreview.net/forum?id=vl89_Xeb8V
[17] S. Geisler, A. Kosmala, D. Herbst, and S. Günnemann, "Spatio-spectral graph neural networks," preprint. Under review.
[18] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," preprint. Under review.