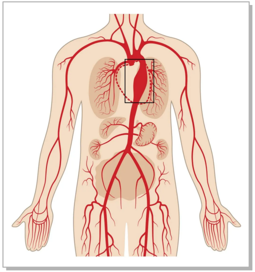
On Physics-Informed Neural Networks and DeepONet for Vascular Flow Simulations in Aortic Aneurysms

Oscar L. Cruz González, Valérie Deplano, Badih Ghattas

Research Funding
Post-doctoral fellowship funded by A*MIDEX
Host Laboratories
IRPHE (UMR7342, Bio-mechanics team)
AMSE (UMR7316, Econometrics and Statistics team)

Motivations

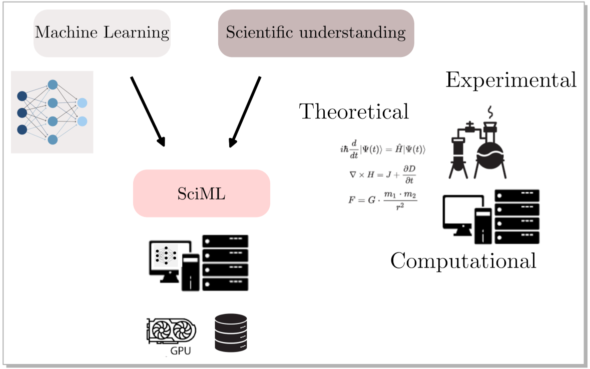
- Modeling of cardiovascular diseases



Why explore new data assimilation algorithms?

1. 4D MRI
   - Limited accuracy in identify hemodynamics.
   - Challenges in obtaining patient-specific flow boundary conditions.
2. CFD simulations
   - Computational demanding and time-consuming.

- AI for Engineering

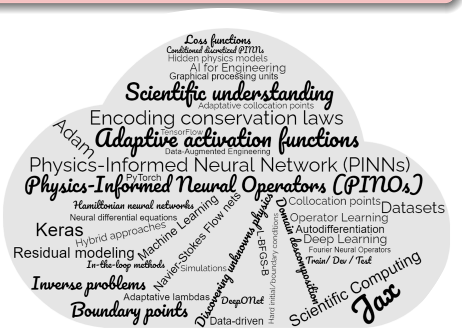


Outline

Aim of this work

- Apply PINNs and (PI-)DeepONets for predicting vascular flow simulations in the context of a 3D AAA idealized model.
- Assess the accuracy of predictions and check improvements in computational efficiency over classical CFD simulations.
- Provide a benchmark study that mimics clinical applications involving patient-specific inflow conditions.

1. Background: PINNs and (PI-)DeepONets
2. Formulation of the 3D Abdominal Aortic Aneurysm (AAA) Idealized Model
3. Adapting PINNs and (PI-)DeepONets for AAA Simulations
4. Results



PINNs

PDEs – illustrative example

$$\mathfrak{N}[u(\boldsymbol{x},t)] = 0, \quad \boldsymbol{x} \in \mathscr{B}, \quad t \in (0,T], \tag{1a}$$
$$\mathfrak{B}[u(\boldsymbol{x},t)] = 0, \quad \boldsymbol{x} \in \partial\mathscr{B}, \quad t \in (0,T], \tag{1b}$$
$$u(\boldsymbol{x},0) = u_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \overline{\mathscr{B}}. \tag{1c}$$

Total Loss functions

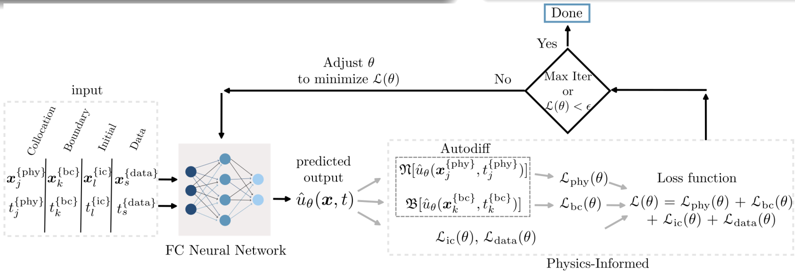
$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{phy}}(\theta) + \mathcal{L}_{\mathrm{bc}}(\theta) + \mathcal{L}_{\mathrm{ic}}(\theta) + \mathcal{L}_{\mathrm{data}}(\theta). \tag{2}$$

$$\mathcal{L}_{\mathrm{phy}}(\theta) = \frac{1}{P_{\mathrm{phy}}} \sum_{j=1}^{P_{\mathrm{phy}}} \left| \mathfrak{N}[\hat{u}_\theta(\boldsymbol{x}_j^{\{\mathrm{phy}\}}, t_j^{\{\mathrm{phy}\}})] \right|^2, \tag{3}$$

$$\mathcal{L}_{\mathrm{bc}}(\theta) = \frac{1}{P_{\mathrm{bc}}} \sum_{k=1}^{P_{\mathrm{bc}}} \left| \mathfrak{B}[\hat{u}_\theta(\boldsymbol{x}_k^{\{\mathrm{bc}\}}, t_k^{\{\mathrm{bc}\}})] \right|^2, \tag{4}$$

$$\mathcal{L}_{\mathrm{ic}}(\theta) = \frac{1}{P_{\mathrm{ic}}} \sum_{l=1}^{P_{\mathrm{ic}}} \left| \hat{u}_\theta(\boldsymbol{x}_l^{\{\mathrm{ic}\}}, 0) - u_0(\boldsymbol{x}_l^{\{\mathrm{ic}\}}) \right|^2, \tag{5}$$

$$\mathcal{L}_{\mathrm{data}}(\theta) = \frac{1}{P_{\mathrm{data}}} \sum_{s=1}^{P_{\mathrm{data}}} \left| \hat{u}_\theta(\boldsymbol{x}_s^{\{\mathrm{data}\}}, t_s^{\{\mathrm{data}\}}) - u(\boldsymbol{x}_s^{\{\mathrm{data}\}}, t_s^{\{\mathrm{data}\}}) \right|^2. \tag{6}$$



- Raissi et al. (2017). Preprint Part I.
- Raissi et al. (2017). Preprint Part II.
- Raissi et al. (2019). Journal of Computational Physics.

Operator Learning

Why?

- Some parameters of a given PDE system are allowed to change in a given range, e.g., source term, domain shape, initial and boundary conditions, coefficients, etc.
- Classical PINNs requires retraining when this occurs, which is disadvantageous and computationally expensive.

Parametric PDEs

$$\mathfrak{N}[u^{(i)}(\boldsymbol{x}, t); f^{(i)}(\boldsymbol{x})] = 0, \quad \boldsymbol{x} \in \mathscr{B}, \quad t \in (0, T], \tag{7a}$$

$$\mathfrak{B}[u^{(i)}(\boldsymbol{x}, t)] = 0, \quad \boldsymbol{x} \in \partial\mathscr{B}, \quad t \in (0, T], \tag{7b}$$

$$u^{(i)}(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \overline{\mathscr{B}}. \tag{7c}$$

- **Goal**: To learn an operator $\mathfrak{G}\colon \mathscr{X} \mapsto \mathscr{Y}$ such that for any input function $f^{(i)}$, we can compute the corresponding output function $u^{(i)} = \mathfrak{G}(f^{(i)})$.
- For all points $(\boldsymbol{x}, t)$ in the domain of the solution, the following relationship holds true, $u^{(i)}(\boldsymbol{x}, t) = \mathfrak{G}(f^{(i)})(\boldsymbol{x}, t)$.

Loss function:

$$\mathcal{L}(\theta) = \frac{1}{NP} \sum_{i=1}^{N} \sum_{j=1}^{P} \left| \hat{\mathfrak{G}}_\theta(f^{(i)})(\boldsymbol{x}_j^{(i)}, t_j^{(i)}) - \mathfrak{G}(f^{(i)})(\boldsymbol{x}_j^{(i)}, t_j^{(i)}) \right|^2, \tag{8}$$

Mishra, S. (2023). Learning Operators. Seminar for Applied Mathematics (SAM).

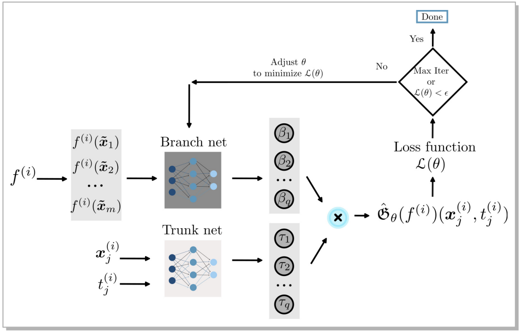
Lu, L., Jin, P., and Karniadakis, G. E. (2021). DeepONet.

Deep Operator Networks (DeepONets / PI-DeepONets)

The Authors in Lu, L.et al. (2021). DeepONet. used the Universal Approximation Theorem for Operator and proposed an expression to approximate $\mathfrak{G}(f^{(i)})(\mathbf{x}_j^{(i)}, t_j^{(i)})$ as follows,

$$\hat{\mathfrak{G}}_\theta(f^{(i)})(\mathbf{x}_j^{(i)}, t_j^{(i)}) := \sum_{k=1}^{q} \underbrace{\beta_k\left(f^{(i)}(\tilde{\mathbf{x}}_1), f^{(i)}(\tilde{\mathbf{x}}_2), \dots, f^{(i)}(\tilde{\mathbf{x}}_m)\right)}_{Branch} \underbrace{\tau_k(\mathbf{x}_j^{(i)}, t_j^{(i)})}_{Trunk}, \tag{9}$$

where the solution map is represented by an unstacked deep learning architecture called **DeepONet**.



Wang et al. (2021). PI-DeepONet.

- Combining PINNs loss functions and the network architecture of DeepONet, and proposed **PI-DeepONet**.
- Ensuring that the learned operator $\hat{\mathfrak{G}}_\theta$ not only fits the data but also satisfies the governing equations, and the initial and boundary conditions.

Formulation of AAA idealized model - steady flow

3D Navier-Stokes equations for incompressible Newtonian fluids

$$\rho_f\,(\boldsymbol{v}(\boldsymbol{x})\cdot\nabla)\,\boldsymbol{v}(\boldsymbol{x}) = -\nabla p(\boldsymbol{x}) + \mu_f \nabla^2 \boldsymbol{v}(\boldsymbol{x}), \quad \boldsymbol{x}\in\mathscr{B}, \tag{10a}$$

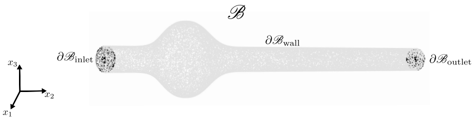
$$\nabla\cdot\boldsymbol{v}(\boldsymbol{x}) = 0, \quad \boldsymbol{x}\in\mathscr{B}. \tag{10b}$$

Boundary conditions

$$\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{v}^{\{\text{inlet}\}}(\boldsymbol{x}), \quad \boldsymbol{x}\in\partial\mathscr{B}_{\text{inlet}}, \tag{10c}$$

$$\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{0},\ \text{(non-slip condition)} \quad \boldsymbol{x}\in\partial\mathscr{B}_{\text{wall}}, \tag{10d}$$

$$\frac{\partial \boldsymbol{v}(\boldsymbol{x})}{\partial \boldsymbol{n}} = \boldsymbol{0},\ \text{(outflow condition)} \quad \boldsymbol{x}\in\partial\mathscr{B}_{\text{outlet}}. \tag{10e}$$



Parabolic profile for velocity – Poiseuille Flow

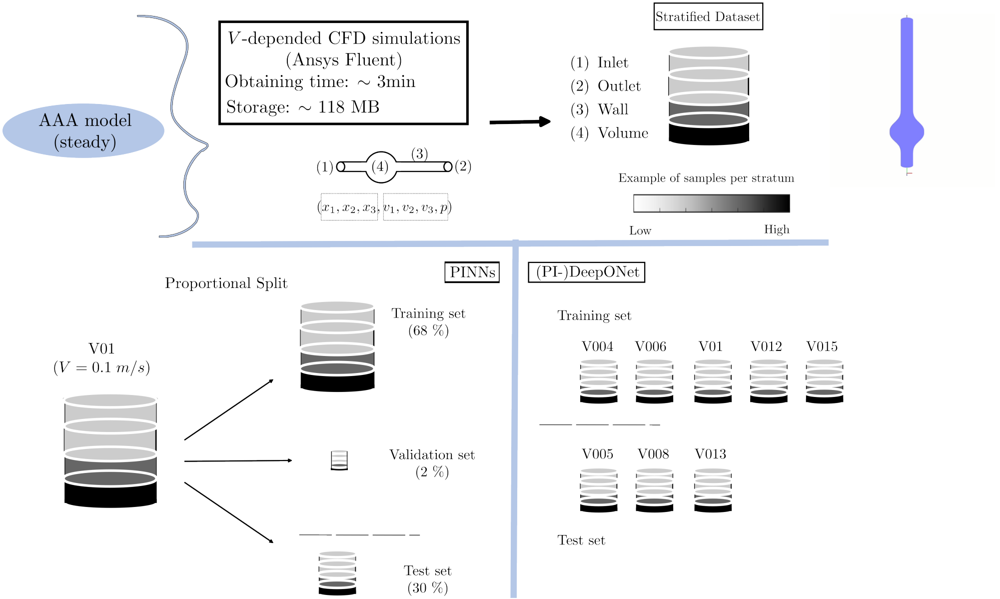
$$\boldsymbol{v}^{\{\text{inlet}\}}(\boldsymbol{x}) := (0, v_2^{\{\text{inlet}\}}(\boldsymbol{x}), 0)$$

$$v_2^{\{\text{inlet}\}}(\boldsymbol{x}) = V\left(1 - \frac{r^2(\boldsymbol{x})}{R^2}\right). \tag{11}$$

μ_f $[kg/(ms)]$	ρ_f $[kg/m^3]$	V $[m/s]$	R $[m]$	specimen length $[m]$
0.00399	1060	[0.04, 0.05, 0.06, 0.08, 0.1, 0.12, 0.13, 0.15]	0.010065	0.26009

Table: NSE parameters.

Ground trust - Dataset preparation



Training set – selective data usage

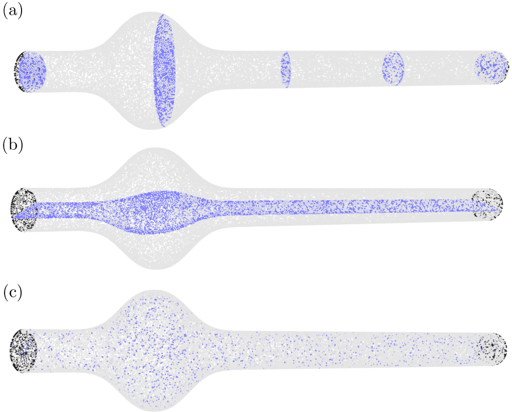
$$\text{Inlet}^* \quad (x_1, x_2, x_3, v_1, v_2, v_3, \text{NaN})$$

$$\text{Outlet}^* \quad (x_1, x_2, x_3, v_1, v_2, v_3, \text{NaN})$$

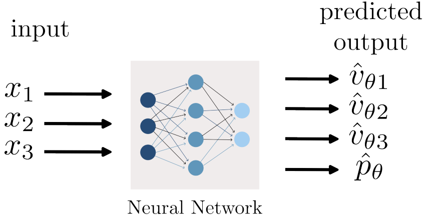
$$\text{Wall}^* \quad (x_1, x_2, x_3, 0, 0, 0, \text{NaN})$$

$$\text{Volume}^* \quad (x_1, x_2, x_3, \text{NaN}, \text{NaN}, \text{NaN}, \text{NaN})$$

$$\text{Data} \subset \text{Volume} \quad (x_1, x_2, x_3, v_1, v_2, v_3, p)$$



PINNs architecture and loss functions



Total loss functions

$$\mathcal{L}_{\text{DeepNN}}(\theta) = \mathcal{L}_{\text{data}}(\theta) + \mathcal{L}_{\text{inlet}}(\theta) + \mathcal{L}_{\text{outlet}}(\theta) + \mathcal{L}_{\text{wall}}(\theta), \tag{12a}$$

$$\mathcal{L}_{\text{PINN}}(\theta) = \mathcal{L}_{\text{data}}(\theta) + \mathcal{L}_{\text{inlet}}(\theta) + \mathcal{L}_{\text{outlet}}(\theta) + \mathcal{L}_{\text{wall}}(\theta) + \mathcal{L}_{\text{phy}}(\theta). \tag{12b}$$

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{P_{\text{data}}} \sum_{j=1}^{P_{\text{data}}} \left(\left\| \hat{\boldsymbol{v}}_\theta(\boldsymbol{x}_j^{\{\text{data}\}}) - \boldsymbol{v}(\boldsymbol{x}_j^{\{\text{data}\}}) \right\|^2 + \left| \hat{p}_\theta(\boldsymbol{x}_j^{\{\text{data}\}}) - p(\boldsymbol{x}_j^{\{\text{data}\}}) \right|^2 \right), \tag{13a}$$

$$\mathcal{L}_{\text{inlet}}(\theta) = \frac{1}{P_{\text{inlet}}} \sum_{k=1}^{P_{\text{inlet}}} \left\| \hat{\boldsymbol{v}}_\theta(\boldsymbol{x}_k^{\{\text{inlet}\}}) - \boldsymbol{v}(\boldsymbol{x}_k^{\{\text{inlet}\}}) \right\|^2, \tag{13b}$$

$$\mathcal{L}_{\text{wall}}(\theta) = \frac{1}{P_{\text{wall}}} \sum_{l=1}^{P_{\text{wall}}} \left\| \hat{\boldsymbol{v}}_\theta(\boldsymbol{x}_l^{\{\text{wall}\}}) - \boldsymbol{v}(\boldsymbol{x}_l^{\{\text{wall}\}}) \right\|^2, \tag{13c}$$

$$\mathcal{L}_{\text{outlet}}(\theta) = \frac{1}{P_{\text{outlet}}} \sum_{s=1}^{P_{\text{outlet}}} \left\| \hat{\boldsymbol{v}}_\theta(\boldsymbol{x}_s^{\{\text{outlet}\}}) - \boldsymbol{v}(\boldsymbol{x}_s^{\{\text{outlet}\}}) \right\|^2, \tag{13d}$$

$$\mathcal{L}_{\text{phy}}(\theta) = \frac{1}{P_{\text{phy}}} \sum_{r=1}^{N_{\text{phy}}} \left\| \hat{\boldsymbol{e}}_\theta(\boldsymbol{x}_r^{\{\text{phy}\}}) \right\|^2. \tag{13e}$$

PINNs architecture and loss function

Residual of the governing NSE

$$\hat{\mathbf{e}}_\theta = (\hat{e}_{\theta 1}, \hat{e}_{\theta 2}, \hat{e}_{\theta 3}, \hat{e}_{\theta 4})$$

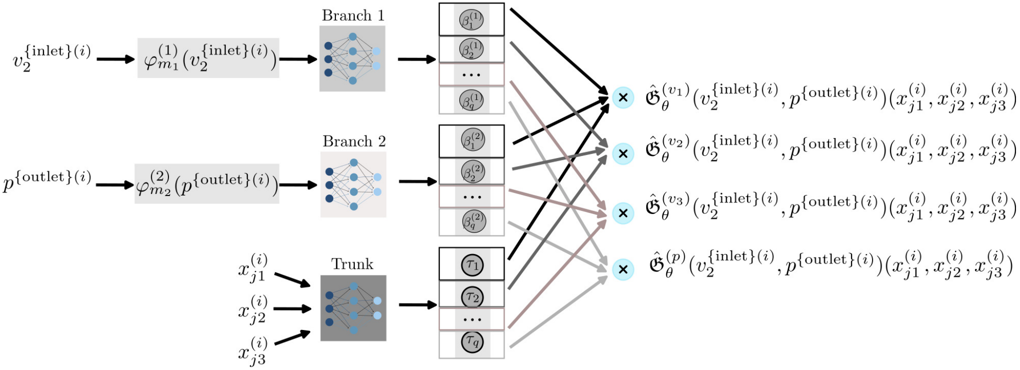
$$\hat{e}_{\theta 1} = \rho_f(\hat{v}_{\theta 1}\frac{\partial \hat{v}_{\theta 1}}{\partial x_1} + \hat{v}_{\theta 2}\frac{\partial \hat{v}_{\theta 1}}{\partial x_2} + \hat{v}_{\theta 3}\frac{\partial \hat{v}_{\theta 1}}{\partial x_3}) + \frac{\partial \hat{p}_\theta}{\partial x_1} - \mu_f(\frac{\partial^2 \hat{v}_{\theta 1}}{\partial x_1^2} + \frac{\partial^2 \hat{v}_{\theta 1}}{\partial x_2^2} + \frac{\partial^2 \hat{v}_{\theta 1}}{\partial x_3^2}), \quad (14a)$$

$$\hat{e}_{\theta 2} = \rho_f(\hat{v}_{\theta 1}\frac{\partial \hat{v}_{\theta 2}}{\partial x_1} + \hat{v}_{\theta 2}\frac{\partial \hat{v}_{\theta 2}}{\partial x_2} + \hat{v}_{\theta 3}\frac{\partial \hat{v}_{\theta 2}}{\partial x_3}) + \frac{\partial \hat{p}_\theta}{\partial x_2} - \mu_f(\frac{\partial^2 \hat{v}_{\theta 2}}{\partial x_1^2} + \frac{\partial^2 \hat{v}_{\theta 2}}{\partial x_2^2} + \frac{\partial^2 \hat{v}_{\theta 2}}{\partial x_3^2}), \quad (14b)$$

$$\hat{e}_{\theta 3} = \rho_f(\hat{v}_{\theta 1}\frac{\partial \hat{v}_{\theta 3}}{\partial x_1} + \hat{v}_{\theta 2}\frac{\partial \hat{v}_{\theta 3}}{\partial x_2} + \hat{v}_{\theta 3}\frac{\partial \hat{v}_{\theta 3}}{\partial x_3}) + \frac{\partial \hat{p}_\theta}{\partial x_3} - \mu_f(\frac{\partial^2 \hat{v}_{\theta 3}}{\partial x_1^2} + \frac{\partial^2 \hat{v}_{\theta 3}}{\partial x_2^2} + \frac{\partial^2 \hat{v}_{\theta 3}}{\partial x_3^2}), \quad (14c)$$

$$\hat{e}_{\theta 4} = \frac{\partial \hat{v}_{\theta 1}}{\partial x_1} + \frac{\partial \hat{v}_{\theta 2}}{\partial x_2} + \frac{\partial \hat{v}_{\theta 3}}{\partial x_3}. \quad (14d)$$

DeepONet architecture and loss functions



Total loss functions

$$\mathcal{L}_{\text{DeepONet}}(\theta) = \mathcal{L}_{\text{data}}(\theta) + \mathcal{L}_{\text{inlet}}(\theta) + \mathcal{L}_{\text{outlet}}(\theta) + \mathcal{L}_{\text{wall}}(\theta), \tag{15a}$$

$$\mathcal{L}_{\text{PI-DeepONet}}(\theta) = \mathcal{L}_{\text{data}}(\theta) + \mathcal{L}_{\text{inlet}}(\theta) + \mathcal{L}_{\text{outlet}}(\theta) + \mathcal{L}_{\text{wall}}(\theta) + \mathcal{L}_{\text{phy}}(\theta). \tag{15b}$$

Jin et al. (2022). SIAM Journal on Scientific Computing.

Wang et al. (2023). Journal of Computational Physics.

DeepONet architecture and loss functions

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{NP_{\text{data}}} \sum_{i=1}^{N} \sum_{j=1}^{P_{\text{data}}} \left(\left\| \hat{\mathfrak{G}}_{\theta}^{(\boldsymbol{v})}(v_2^{\{\text{inlet}\}(i)}, p^{\{\text{outlet}\}(i)})(\boldsymbol{x}_j^{\{\text{data}\}(i)}) - \boldsymbol{v}(\boldsymbol{x}_j^{\{\text{data}\}(i)}) \right\|^2 \right.$$

$$\left. + \left| \hat{\mathfrak{G}}_{\theta}^{(p)}(v_2^{\{\text{inlet}\}(i)}, p^{\{\text{outlet}\}(i)})(\boldsymbol{x}_j^{\{\text{data}\}(i)}) - p(\boldsymbol{x}_j^{\{\text{data}\}(i)}) \right|^2 \right), \tag{16a}$$

$$\mathcal{L}_{\text{inlet}}(\theta) = \frac{1}{NP_{\text{inlet}}} \sum_{i=1}^{N} \sum_{k=1}^{P_{\text{inlet}}} \left\| \hat{\mathfrak{G}}_{\theta}^{(\boldsymbol{v})}(v_2^{\{\text{inlet}\}(i)}, p^{\{\text{outlet}\}(i)})(\boldsymbol{x}_k^{\{\text{inlet}\}(i)}) - \boldsymbol{v}(\boldsymbol{x}_k^{\{\text{inlet}\}(i)}) \right\|^2, \tag{16b}$$

$$\mathcal{L}_{\text{wall}}(\theta) = \frac{1}{NP_{\text{wall}}} \sum_{i=1}^{N} \sum_{l=1}^{P_{\text{wall}}} \left\| \hat{\mathfrak{G}}_{\theta}^{(\boldsymbol{v})}(v_2^{\{\text{inlet}\}(i)}, p^{\{\text{outlet}\}(i)})(\boldsymbol{x}_l^{\{\text{wall}\}(i)}) - \boldsymbol{v}(\boldsymbol{x}_l^{\{\text{wall}\}(i)}) \right\|^2, \tag{16c}$$

$$\mathcal{L}_{\text{outlet}}(\theta) = \frac{1}{NP_{\text{outlet}}} \sum_{i=1}^{N} \sum_{s=1}^{P_{\text{outlet}}} \left\| \hat{\mathfrak{G}}_{\theta}^{(\boldsymbol{v})}(v_2^{\{\text{inlet}\}(i)}, p^{\{\text{outlet}\}(i)})(\boldsymbol{x}_s^{\{\text{outlet}\}(i)}) - \boldsymbol{v}(\boldsymbol{x}_s^{\{\text{outlet}\}(i)}) \right\|^2, \tag{16d}$$

$$\mathcal{L}_{\text{phy}}(\theta) = \frac{1}{NP_{\text{phy}}} \sum_{i=1}^{N} \sum_{r=1}^{N_{\text{phy}}} \left\| \hat{\boldsymbol{E}}_{\theta}(v_2^{\{\text{inlet}\}(i)}, p^{\{\text{outlet}\}(i)})(\boldsymbol{x}_r^{\{\text{phy}\}(i)}) \right\|^2, \tag{16e}$$

DeepONet architecture and loss function

Residual of the governing NSE

$$\hat{\boldsymbol{E}}_\theta = (\hat{E}_{\theta 1}, \hat{E}_{\theta 2}, \hat{E}_{\theta 3}, \hat{E}_{\theta 4})$$

$$\hat{E}_{\theta 1} = \rho_f(\hat{\mathfrak{G}}_\theta^{(v_1)} \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_1)}}{\partial x_1} + \hat{\mathfrak{G}}_\theta^{(v_2)} \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_1)}}{\partial x_2} + \hat{\mathfrak{G}}_\theta^{(v_3)} \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_1)}}{\partial x_3}) + \frac{\partial \hat{\mathfrak{G}}_\theta^{(p)}}{\partial x_1} - \mu_f(\frac{\partial^2 \hat{\mathfrak{G}}_\theta^{(v_1)}}{\partial x_1^2} + \frac{\partial^2 \hat{\mathfrak{G}}_\theta^{(v_1)}}{\partial x_2^2} + \frac{\partial^2 \hat{\mathfrak{G}}_\theta^{(v_1)}}{\partial x_3^2}), \quad (17a)$$

$$\hat{E}_{\theta 2} = \rho_f(\hat{\mathfrak{G}}_\theta^{(v_1)} \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_2)}}{\partial x_1} + \hat{\mathfrak{G}}_\theta^{(v_2)} \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_2)}}{\partial x_2} + \hat{\mathfrak{G}}_\theta^{(v_3)} \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_2)}}{\partial x_3}) + \frac{\partial \hat{\mathfrak{G}}_\theta^{(p)}}{\partial x_2} - \mu_f(\frac{\partial^2 \hat{\mathfrak{G}}_\theta^{(v_2)}}{\partial x_1^2} + \frac{\partial^2 \hat{\mathfrak{G}}_\theta^{(v_2)}}{\partial x_2^2} + \frac{\partial^2 \hat{\mathfrak{G}}_\theta^{(v_2)}}{\partial x_3^2}), \quad (17b)$$

$$\hat{E}_{\theta 3} = \rho_f(\hat{\mathfrak{G}}_\theta^{(v_1)} \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_3)}}{\partial x_1} + \hat{\mathfrak{G}}_\theta^{(v_2)} \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_3)}}{\partial x_2} + \hat{\mathfrak{G}}_\theta^{(v_3)} \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_3)}}{\partial x_3}) + \frac{\partial \hat{\mathfrak{G}}_\theta^{(p)}}{\partial x_3} - \mu_f(\frac{\partial^2 \hat{\mathfrak{G}}_\theta^{(v_3)}}{\partial x_1^2} + \frac{\partial^2 \hat{\mathfrak{G}}_\theta^{(v_3)}}{\partial x_2^2} + \frac{\partial^2 \hat{\mathfrak{G}}_\theta^{(v_3)}}{\partial x_3^2}), \quad (17c)$$

$$\hat{E}_{\theta 4} = \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_1)}}{\partial x_1} + \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_2)}}{\partial x_2} + \frac{\partial \hat{\mathfrak{G}}_\theta^{(v_3)}}{\partial x_3}. \quad (17d)$$

Results – AAA simulations via PINNs – DeepNNs Vs PINNs

Models' settings

NN (MLP, Tanh activation, 4 hidden layers, 256 neurons, Xavier normal initialization**)**

Training (200 000 iterations, 1024 batch size (batch training), Adam optimizer**)**

Techniques for Enhancing NN

- Dimensionless NSE
- Adaptive sampling within the mesh
- Exponential decay for the learning rate + optimizer scheduler
- Loss balancing (Grad Norm technique)

Dataset $V = 0.1\ m/s$		**DeepNN** L^2-relative error on the test set / computational efficiency			**PINN** L^2-relative error on the test set / computational efficiency		
Source of data	%	Magnitude of velocity	Pressure	Runtime(min)	Magnitude of velocity	Pressure	Runtime(h)
Cross-section 5 slices	1.42	1.1881e-01	1.5626e-01	28.30	4.6507e-02	4.8523e-02	2.95
Longitudinal 1 slice	1.87	2.7889e-01	7.0684e-01	28.11	4.0500e-02	1.1026e-02	2.95
Random Coord. points	0.3	23253e-02	1.4570e-02	28.84	8.7983e-03	4.7490e-03	2.95

*Runtime: NVIDIA RTX A6000 GPU with 48 GB of memory.

AAA simulations via PINNs – PINNs without Data

$$\mathcal{L}_{\text{PINN}}(\theta) = \cancel{\mathcal{L}_{\text{data}}}(\theta) + \mathcal{L}_{\text{inlet}}(\theta) + \mathcal{L}_{\text{outlet}}(\theta) + \mathcal{L}_{\text{wall}}(\theta) + \mathcal{L}_{\text{phy}}(\theta).$$

$$\text{Inlet}^* \quad (\boxed{x_1, x_2, x_3}, \boxed{v_1, v_2, v_3, \text{NaN}})$$

$$\text{Outlet}^* \quad (\boxed{x_1, x_2, x_3}, \boxed{v_1, v_2, v_3, \text{NaN}})$$

$$\text{Wall}^* \quad (\boxed{x_1, x_2, x_3}, \boxed{0, 0, 0, \text{NaN}})$$

$$\text{Volume}^* \quad (\boxed{x_1, x_2, x_3}, \boxed{\text{NaN}, \text{NaN}, \text{NaN}, \text{NaN}})$$

+

- **Fourier features embedding**
- **Modified MLP**
- **Random weight factorization (RWF)**

Wang et al. (2023). An Expert's Guide to Training Physics-informed Neural Networks.

300 000 iter - batch size 3000					L^2- relative error on the test set		
Exponential decay	**Grad Norm**	**Fourier embedding**	**Modified MLP**	**RWF**	**Magnitude of velocity**	**Pressure**	**Runtime (h)**
✓	✓	✓	✓	✓	2.4912e-02	2.8641e-02	9.60
✓	✓	✓	✓	✗	2.8433e-02	4.4519e-02	9.57
✗	✓	✓	✓	✓	3.6517e-02	4.7310e-02	9.62
✓	✓	✗	✓	✓	4.0986e-02	3.0455e-02	8.37
✓	✗	✓	✓	✓	4.3730e-02	4.1168e-02	5.51
✓	✓	✓	✗	✓	6.1967e-02	1.1688e-01	5.43
✗	✗	✗	✗	✗	2.2596e-01	5.0376e-01	4.21

Table: Ablation study

AAA simulations via PINNs – DeepONet Vs PI-DeepONet

Models' settings

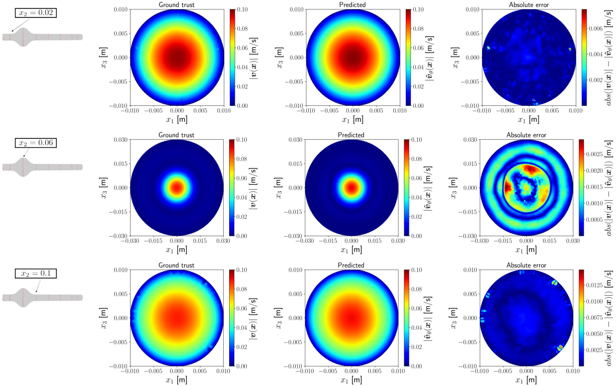
Branch 1 and Branch 2 (
MLP, Tanh activation,
3 hidden layers, 100 neurons,
1037 input dim, 400 output dim,
Xavier normal initialization**)**

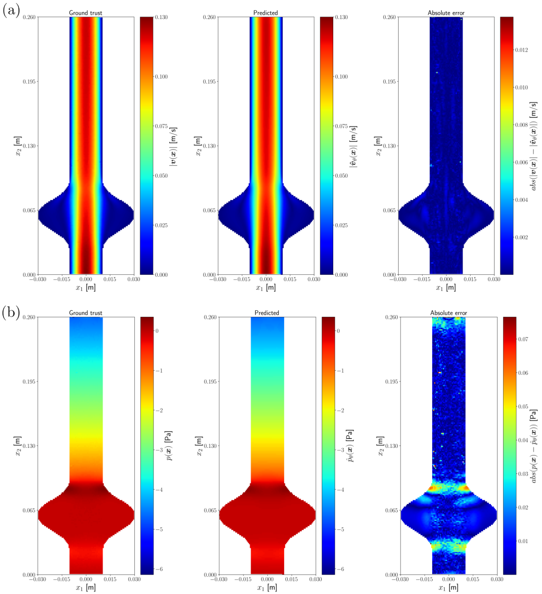
Trunk (MLP, Tanh activation,
4 hidden layers, 200 neurons,
3 input dim, 400 output dim,
Xavier normal initialization**)**

Training (200 000 iterations,
1024 batch size (batch training),
Adam optimizer**)**

- Dimensionless NSE
- Adaptive sampling within the mesh
- Exponential decay for the learning rate + optimizer scheduler

Data			**DeepONet**			**PI-DeepONet**		
Source of data	dataset		MoV	Pressure	Runtime(h)	MoV	Pressure	Runtime(h)
Cross-section / 5 slices	train	V004	1.2238E-01	8.4834E-02	8.93	5.4001E-02	2.0648E-02	14.52
		V006	1.1312E-01	8.9753E-02		3.4948E-02	2.0793E-02	
1.42% of Volume		V01	1.0223E-01	9.8372E-02		2.6794E-02	2.4334E-02	
		V012	9.8716E-02	1.0255E-01		2.7489E-02	2.7283E-02	
		V015	9.4953E-02	1.0834E-01		2.9623E-02	3.1953E-02	
	test	V005	1.1727E-01	8.8884E-02		4.2461E-02	2.3986E-02	
		V008	1.0679E-01	9.3711E-02		2.8150E-02	2.2159E-02	
		V013	9.7288E-02	1.0455E-01		2.8119E-02	2.8774E-02	
Longitudinal / 1 slice	train	V004	1.9664E-01	4.8856E-01	8.93394813	1.4748E-01	1.5595E-02	14.66
		V006	1.9120E-01	5.1318E-01		9.2258E-02	1.4848E-02	
1.87% of Volume		V01	1.8424E-01	5.6148E-01		4.9349E-02	1.5549E-02	
		V012	1.8177E-01	5.8341E-01		4.3853E-02	1.6737E-02	
		V015	1.7889E-01	6.1449E-01		4.3609E-02	1.9169E-02	
	test	V005	1.9376E-01	4.9920E-01		1.1667E-01	2.4341E-02	
		V008	1.8726E-01	5.3853E-01		6.2953E-02	1.6272E-02	
		V013	1.8073E-01	5.9400E-01		4.2964E-02	1.7408E-02	
Random / Coord. points	train	V004	1.7673E-02	6.2008E-03	8.93	3.1676E-02	1.7270E-02	14.74
		V006	1.7109E-02	6.4153E-03		2.0244E-02	1.4592E-02	
0.3% of Volume		V01	1.7208E-02	7.5501E-03		1.1227E-02	9.9699E-03	
		V012	1.7494E-02	8.2986E-03		1.0050E-02	7.7110E-03	
		V015	1.8183E-02	9.7196E-03		1.0552E-02	6.7369E-03	
	test	V005	1.7228E-02	1.2525E-02		2.5097E-02	1.9196E-02	
		V008	1.7087E-02	7.7839E-03		1.4320E-02	1.5595E-02	
		V013	1.7694E-02	8.7354E-03		9.9959E-03	7.0725E-03	


$x_2 = 0.02$
$x_2 = 0.06$
$x_2 = 0.1$
Ground trust
Predicted
Absolute error
x_1 [m]
x_3 [m]
$|\boldsymbol{v}(\boldsymbol{x})|$ [m/s]
$|\hat{\boldsymbol{v}}_\theta(\boldsymbol{x})|$ [m/s]
$abs(|\boldsymbol{v}(\boldsymbol{x})| - |\hat{\boldsymbol{v}}_\theta(\boldsymbol{x})|)$ [m/s]


(a)
(b)
Ground trust
Predicted
Absolute error
x_1 [m]
x_2 [m]
$|\boldsymbol{v}(\boldsymbol{x})|$ [m/s]
$|\hat{\boldsymbol{v}}_\theta(\boldsymbol{x})|$ [m/s]
$abs(|\boldsymbol{v}(\boldsymbol{x})| - |\hat{\boldsymbol{v}}_\theta(\boldsymbol{x})|)$ [m/s]
$p(\boldsymbol{x})$ [Pa]
$\hat{p}_\theta(\boldsymbol{x})$ [Pa]
$abs(p(\boldsymbol{x}) - \hat{p}_\theta(\boldsymbol{x}))$ [Pa]

Discussion and conclusions

PINNs

1. The method has generalized predictions across an entire domain using sparse data.
2. It provided robust initial predictions for flow simulations when the model was trained solely on governing equations and boundary conditions.

(PI-)DeepONets

1. We proposed an architecture that integrates multiple inputs and outputs while showing accurate results.
2. It can manage varying conditions without requiring the time-consuming retraining process.
3. The main strength lay in the ability to generalize inference to unseen input conditions, such as the possibility of inferring whole new V-dependent datasets, within a specified range of trained value (Inference runtime is faster than CFD simulations).

We compared against CFD simulations on benchmark datasets and demonstrate good agreement between the results.

Limitations

PINNs

1. The runtime of the training phase of PINNs method is significantly higher than the computational time of CFD simulations.

(PI-)DeepONets

1. It requires substantial more training data in comparison to PINNs, which can be costly to obtain.
2. The model may struggle to generalize to inputs outside the specified range of trained values.

Further work

1. Extensions to handle unsteady flows and moving boundaries.
2. Adaptation to more complex biological structures.
3. Parameterized PINNs can be considered to address the retraining issue associated with changing conditions.

On Physics-Informed Neural Networks and DeepONet for Vascular Flow Simulations in Aortic Aneurysms

Thanks for your attention!

For further information ...

AAA simulations via (PI-)DeepONet – Training/Test datasets resolution impact