LEARNING NONLINEAR TRANSIENT DYNAMICS IN FLUID FLOW NETWORKS WITH THE NEURAL BASIS METHOD

Anonymous authors
Paper under double-blind review

ABSTRACT

Fluid flow networks are ubiquitous across engineered and natural systems, including pipeline infrastructure and biological vasculature. In many practical regimes, they are governed by nonlinear edgewise hyperbolic laws coupled through continuity and conservation at junctions. Learning their dynamics is important in many-query scenarios requiring rapid evaluation across unseen parametric settings. Conventional methods re-solve the system for each new query, resulting in prohibitive cost. Existing physics-informed neural approaches typically impose governing equations through penalty losses over generic neural approximators, rather than embedding the physical laws in their neural formulation. Such losses act only as optimization surrogates during training, with no guarantee that their reduction reduces solution error. Moreover, the resulting optimization problem is prone to ill-conditioning, particularly for hyperbolic systems. Building on the "representation-then-projection" principle of the neural basis method, we introduce a new approach for modeling fluid flow networks. Fixed neural bases define the representation space on each edge, while projection of the discretized equations determines the expansion coefficients, embedding edgewise laws into a conservative formulation. Edge subdomains are coupled through non-overlapping domain-decomposition. A neural operator then learns, directly in this reduced space, the parametric map to the corresponding expansion coefficients. On two representative network topologies, looped and branching, we reproduce references with relative L^2 errors below 0.02%, whereas the physics-informed neural network baseline fails. Across unseen boundary conditions in both interpolation and extrapolation, we maintain mean pressure errors below 1% and mean flux errors below 3.3%, while the physics-informed DeepONet baseline reaches 8–18% pressure errors and flux errors of 85–95%, failing to capture the underlying dynamics. The results demonstrate the accuracy and efficiency of the proposed framework for learning nonlinear transient dynamics in fluid flow networks, highlighting its potential for real-time monitoring, design exploration, and uncertainty quantification.

Keywords: fluid flow network, nonlinear transient dynamics, operator learning, physics-informed machine learning, neural basis method

1 INTRODUCTION

Fluid flow networks, such as natural gas transmission systems, are governed by nonlinear hyperbolic conservation laws on each pipe, coupled at junctions by pressure continuity and mass conservation (Banda et al., 2006b). Planning, control, and uncertainty quantification require the transient network state under many boundary conditions, and re-solving with conventional methods for each query is the bottleneck. Current learned surrogates face critical limitations: physics-informed networks and PI-DeepONet enforce the equations as heuristically weighted penalty losses whose magnitude is not tied to solution error, and which are poorly conditioned for transient hyperbolic dynamics (Wang et al., 2021; Strelow et al., 2023); data-driven neural operators do not embed junction coupling or discrete conservation (Lu et al., 2019; Li et al., 2020). The Neural Basis Method (NBM) (Wang &

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Wang, 2026) avoids penalty training by projecting discretized equations onto a fixed neural basis, but has so far been developed with global bases on single domains. Networks, however, demand a representation that is local to each pipe yet conserves mass across junctions, and a discretization that stays stable for nonlinear hyperbolic transients.

We introduce NBM to fluid flow networks. Each edge carries a local neural basis, junction pressures enter as dedicated coefficients, and the staggered finite-volume equations of the whole network are projected in a single monolithic least-squares solve. A network trained on the projected discrete residual then maps time and inlet pressure to a POD-compressed latent state, returning the transient at any queried time without time stepping.

Our contributions are summarized as:

- **Localized NBM on metric graphs.** We couple edgewise bases through junction conditions in one monolithic projection, extending NBM from single-domain global bases to network domains.
- **NBM for nonlinear hyperbolic systems.** Unlike collocation on the strong form, we project a staggered finite-volume discretization of the isothermal Euler equations onto the neural basis, extending NBM beyond the elliptic–parabolic setting in which it was introduced; it recovers the discrete solution on both looped and branching topologies to relative L^2 errors of 10^{-5} in pressure and 10^{-4} in flux.
- **Operator learning on networks.** NBM-OL predicts the transient state at held-out inlet pressures in a single forward pass. Averaged over held-out points, relative L^2 errors in pressure/flux are $0.065\%/1.05\%$ (interpolation) and $0.23\%/1.62\%$ (extrapolation) on the 7-edge network, and $0.12\%/3.25\%$ and $0.97\%/2.84\%$ on the tree network, compared with 7.9–17.9% pressure and 85–95% flux errors for PI-DeepONet.

2 Related Work

Neural operators. Neural operators learn maps between function spaces that approximate the solution operator of a PDE (Kovachki et al., 2023). DeepONet factorizes the operator into branch and trunk networks (Lu et al., 2019), and the Fourier Neural Operator parameterizes the kernel in the Fourier domain (Li et al., 2020). PI-DeepONet adds the PDE residual to the training objective, reducing reliance on paired data (Wang et al., 2021); we use a graph-adapted variant as a baseline. Large pretrained models such as MPP (McCabe et al., 2024), DPOT (Hao et al., 2024), and Poseidon (Herde et al., 2024) typically learn a one-step update applied autoregressively, which can accumulate error over long rollouts (Lippe et al., 2023); continuous-time alternatives learn the dynamics directly (Hou et al., 2026). Closest to our setting, Wu et al. (2025) and Ždrale et al. (2026) develop neural operators for networked systems and for hyperbolic conservation laws on graphs. NBM-OL learns a direct map, here from (t, P_{in}) to the latent state, but represents the solution in a fixed basis obtained by projecting the discretized network equations rather than in a learned one.

Fixed neural bases and least-squares projection. A separate line of work fixes the network weights and solves for expansion coefficients by least squares, rather than training a network to satisfy the PDE. Random feature methods (Chen et al., 2022) and extreme learning machines use randomly initialized single-hidden-layer networks as basis functions; local variants partition the domain and couple subdomains through interface conditions (Dong & Li, 2021). These constructions are effective but their non-orthogonal bases can become severely ill-conditioned as the basis is enriched. The Neural Basis Method (Wang & Wang, 2026) addresses this with a multilayer basis generator, physics-conforming vector bases, and residual weighting consistent with the operator, and extends the construction to parametric operator learning. It has so far been developed with global bases on single domains, with localization identified as an open direction. We take that step for network domains: edgewise bases coupled through junction rows, applied to a nonlinear hyperbolic system, with the projection performed on a staggered finite-volume discretization rather than on collocated strong-form residuals.

Reduced-order and latent-space operator learning. Projection-based reduced-order models classically compress the state on a POD subspace and recover the reduced dynamics either by Galerkin projection or, as in least-squares Petrov–Galerkin schemes, by minimizing the discrete residual over the reduced space (Carlberg et al., 2017). POD-NN replaces the reduced solve with a network regressing POD coefficients on the parameters (Hesthaven & Ubbiali, 2018), and Kontolati et al.

(2024) learn autoencoder latents and a neural operator evolving them. NBM-OL sits between these: the spatial reduction is a fixed, untrained basis rather than a learned encoder, the latent dynamics are a direct map from time and parameter rather than a reduced time-stepping scheme, and training minimizes the projected discrete residual, keeping the reduced coordinate interpretable in terms of the finite-volume discretization.

Numerical methods for gas pipeline networks. Transient gas flow in pipeline networks is classically formulated as the isothermal Euler equations on a metric graph with Kirchhoff-type coupling at junctions (Osiadacz, 1984; Banda et al., 2006b;a; Herty, 2008). Explicit staggered-grid schemes conserve mass exactly by construction (Gyrya & Zlotnik, 2019), and operator-splitting schemes give unconditionally stable, second-order simulation on looped networks (Dyachenko et al., 2017). Strelow et al. (2023) study PINNs for gas transport and find that adaptive loss balancing offers no advantage over well-chosen constant weights. Our staggered finite-volume discretization (Appendix A) follows the same physical formulation, but serves as the target of a reduced, learnable projection rather than as a standalone forward solver.

3 Problem Setup

Network model. We consider isothermal compressible gas flow on a pipeline network, represented as a directed metric graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each edge $e \in \mathcal{E}$ is identified with an interval $[0, L_e]$, oriented from its tail vertex ($x = 0$) to its head vertex ($x = L_e$); the orientation fixes the sign convention for flow. On each edge the state consists of the pressure $P_e(x,t)$ and the mass flux $q_e(x,t) = \rho_e u_e$ (mass flow per unit cross-sectional area), which evolve according to the isothermal Euler equations in pressure–flux form (Appendix A.2):

$$\partial_t P_e + a^2\, \partial_x q_e = 0, \tag{3.1}$$

$$\partial_t q_e + \partial_x P_e + \partial_x \left(\frac{a^2 q_e^2}{P_e} \right) + \frac{f a^2 |q_e| q_e}{2 D P_e} = 0, \tag{3.2}$$

for $x \in [0, L_e]$, $t \in [0, T]$, where a is the isothermal sound speed, D the pipe diameter, and f the Darcy friction factor. equation 3.1 expresses mass conservation; equation 3.2 balances inertia, pressure gradient, momentum advection, and wall friction. Both nonlinear terms are singular as $P \to 0$, so pressures remain strictly positive throughout.

Boundary and junction conditions. The vertex set splits as $\mathcal{V} = \mathcal{V}_D \,\dot\cup\, \mathcal{V}_J$ into boundary vertices with prescribed pressure and internal junctions. $\mathcal{V}_D$ contains a single inlet v_{in} held at P_{in}, and outlet vertices held at a fixed $P_{\text{out}} < P_{\text{in}}$. At each junction $v \in \mathcal{V}_J$, pressure is continuous and mass is conserved:

$$P_e(x_v, t) = P_{e'}(x_v, t), \qquad \forall\, e, e' \text{ incident to } v, \tag{3.3}$$

$$\sum_{e \in \text{in}(v)} q_e(L_e, t) = \sum_{e \in \text{out}(v)} q_e(0, t). \tag{3.4}$$

Since all pipes share the same diameter, conservation of mass flow reduces to conservation of mass flux.

Initial condition. The network starts at rest at the inlet pressure, $P_e(x, 0) = P_{\text{in}}$, $q_e(x, 0) = 0$ for all $e \in \mathcal{E}$. This state is incompatible with the outlet condition: at $t = 0^+$ the outlet pressure drops to P_{out}, launching an expansion wave into the network. The resulting transient, with reflections at junctions and boundaries, relaxes toward a friction-dominated steady state with nonzero throughflow. This transient is what the models must capture.

Solution operator. With topology, pipe parameters, and P_{out} fixed, the network state is determined by the inlet boundary data. We write the solution operator as

$$S : [0, T] \times \mathcal{P} \to \mathcal{X}, \qquad (t, P_{\text{in}}) \mapsto \big(P_e(\cdot, t), q_e(\cdot, t)\big)_{e \in \mathcal{E}}, \tag{3.5}$$

where $\mathcal{X}$ is the space of pressure–flux fields on $\mathcal{G}$ and $\mathcal{P}$ is the space of admissible inlet data. In this work, $\mathcal{P}$ is the one-parameter family of constant inlet pressures $P_{\text{in}} \in [P_{\text{in}}^{\min}, P_{\text{in}}^{\max}]$; time-varying and multi-inlet boundary data are discussed in Section 6.

Discrete target and learning task. Our target is the solution operator of the staggered finite-volume scheme of Appendix A, which carries the conservation structure and junction coupling the learned model inherits. The NBM solver (Section 4) reproduces this scheme in the neural basis, and the learned model is trained on NBM reference trajectories at a finite set of inlet pressures in $[P_{\text{in}}^{\min}, P_{\text{in}}^{\max}]$. It returns the network state at any queried time in a single evaluation, without time stepping, for P_{in} inside that interval (interpolation) or beyond it (extrapolation). Errors are measured against these discrete references.

4 METHOD OVERVIEW

NBM-OL proceeds in two stages, illustrated in Figure 1. The first is a deterministic solve: each edge is equipped with a fixed neural basis (Section 4.1), and the staggered finite-volume equations of the network, junction and boundary rows included, are projected onto it and solved by weighted least squares at every time step (Section 4.2). Repeating this over a grid of inlet pressures yields coefficient trajectories. The second stage learns in a reduced coordinate: proper orthogonal decomposition compresses the coefficients from K to $r \ll K$ dimensions, and a network maps (t, P_{in}) to this latent state, trained on the projected discrete residual (Section 4.3). No basis function is ever trained; only the latent map is. At inference, one forward pass returns the network state at any queried time, with no time stepping.

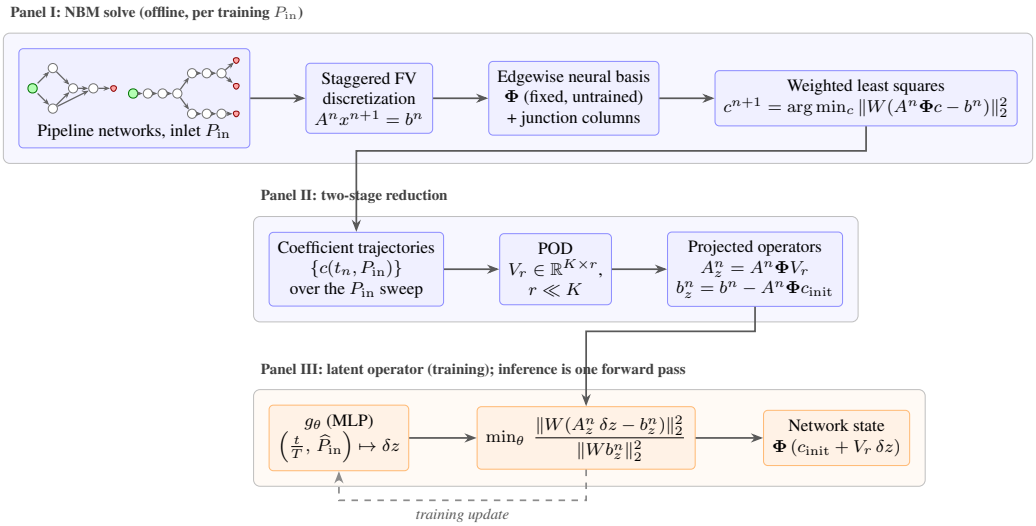


Figure 1: Overview of NBM-OL. **Panel I**: for each training inlet pressure, the staggered finite-volume system of the whole network, including junction and boundary rows, is projected onto a fixed edgewise neural basis and solved by weighted least squares, giving coefficient trajectories; the thumbnails show the two topologies used in our experiments (Section 5.1). **Panel II**: POD compresses these trajectories to $r \ll K$ dimensions, and the projected operators (A_z^n, b_z^n) are precomputed. **Panel III**: an MLP maps normalized time and inlet pressure to the latent state, trained on the projected residual; at inference, one forward pass returns the network state at any queried time.

4.1 NEURAL BASIS ON A NETWORK

Following Wang & Wang (2026), spatial bases are generated by a fixed residual network with layers

$$h_\ell = \beta_\ell\, h_{\ell-1} P_\ell^\top + \tanh\big(\alpha_\ell \odot (h_{\ell-1} W_\ell^\top + b_\ell)\big), \qquad \ell = 1, 2, \tag{4.1}$$

where W_ℓ, P_ℓ, b_ℓ are fixed after random initialization, α_ℓ is a per-neuron shape vector, and β_ℓ a scalar residual strength. The network is never trained. We evaluate it on the normalized edge coordinate $x/L_e \in [0, 1]$ at the finite-volume cell centers (for P) and faces (for q), and orthonormalize the resulting columns, which controls the conditioning of the projected system. Since all edges share

the same normalized discretization, one local basis per field, with k functions, is reused on every edge.

The global basis Φ is block diagonal, with one pressure block and one flux block per edge, and one additional unit column per internal junction, so that each junction pressure is a coefficient. The coefficient vector $c \in \mathbb{R}^K$ therefore has $K = 2k|E| + |V_J|$ entries. Edges interact only through the junction rows of the projected system below, which is what localizes NBM to network domains: each edge carries its own expansion, coupled to its neighbors through pressure continuity and mass conservation, and the whole network is solved in a single monolithic projection.

4.2 Projected Finite-Volume System

The network equations are discretized by a staggered finite-volume scheme with backward Euler time stepping and LUST reconstruction of momentum advection (Appendix A). Freezing the friction coefficient and advective velocity at the previous state u^n gives, at each step, a linear system $A^n x^{n+1} = b^n$ over all cells, faces, junction, and boundary rows, where both A^n and b^n depend on u^n. Substituting $x^{n+1} = \Phi c^{n+1}$ yields an overdetermined system, solved as

$$c^{n+1} = \arg\min_c \left\| W\left(A^n \Phi c - b^n\right) \right\|_2^2, \tag{4.2}$$

where the diagonal matrix W assigns a constant penalty weight to boundary and junction rows and unit weight to interior rows. We solve equation 4.2 via the normal equations and a Cholesky factorization. we use a constant penalty on boundary and junction rows rather than the energy-consistent weights of Wang & Wang (2026), which places all residual rows on a common physical scale.

4.3 Latent Operator and Training Objective

A multilayer perceptron g_θ maps normalized time and inlet pressure to the latent state,

$$\delta z(t, P_{\text{in}}) = z_{\text{scale}} \odot g_\theta\left(\tfrac{t}{T},\ \tfrac{P_{\text{in}} - P_{\text{in}}^{\min}}{P_{\text{in}}^{\max} - P_{\text{in}}^{\min}}\right), \tag{4.3}$$

where z_{scale} is the componentwise standard deviation of the training latents $V_r^\top(c_{\text{ref}} - c_{\text{init}})$, offset by one. The predicted state is $\Phi\big(c_{\text{init}} + V_r\,\delta z\big)$: a combination of fixed basis functions with coefficients computed from the inlet data, i.e. the branch–trunk structure of DeepONet with a fixed trunk ΦV_r derived from the discretized physics.

For a training pair (t_{n+1}, P_{in}), we assemble A^n and b^n of equation 4.2 from the reference state $u^n = \Phi\, c_{\text{ref}}(t_n, P_{\text{in}})$ and substitute $c = c_{\text{init}} + V_r \delta z$, giving the latent residual

$$R^n(\delta z) = W\big(A_z^n \delta z - b_z^n\big), \qquad A_z^n = A^n \Phi V_r, \quad b_z^n = b^n - A^n \Phi\, c_{\text{init}}(P_{\text{in}}). \tag{4.4}$$

Training minimizes

$$\mathcal{L}(\theta) = \mathbb{E}\left[\frac{1}{m} \frac{\|R^n(\delta z(t_{n+1}, P_{\text{in}}))\|_2^2}{\|W b_z^n\|_2^2}\right], \tag{4.5}$$

over all training pairs, where m is the number of rows. For each pair, the objective is minimized by the latent one-step solution from the reference state, so training fits reference next states in the metric induced by the projected operator rather than in Euclidean coefficient distance. Reference data enter NBM-OL through the POD basis and the states u^n. No supervised loss term is used: the residual objective alone suffices. Because each A_z^n has r rather than K columns, storing the projected operators for all 33,000 training pairs of the 7-edge network requires 13.0 GB, whereas the full-coefficient equivalent would require 209.3 GB, a factor $K/r = 16.1$. Training hyperparameters are listed in Table 5.

Residual as an error certificate. Fix a training pair, let $M = W A_z^n$ have full column rank, and let $\delta z^\star$ minimize $\|R^n\|_2$. Since $R^n(\delta z^\star)$ is orthogonal to the range of M,

$$\|\delta z - \delta z^\star\|_2 \le \frac{\sqrt{\|R^n(\delta z)\|_2^2 - \|R^n(\delta z^\star)\|_2^2}}{\sigma_{\min}(M)}. \tag{4.6}$$

Since Φ has orthonormal columns and V_r is orthonormal, ΦV_r is an isometry, so the left-hand side equals the distance in grid values between the prediction and the best one-step solution representable in the latent space. The training objective therefore bounds a solution error, with a computable constant, rather than serving only as an optimization surrogate.

5 EXPERIMENTS

We ask two questions: whether NBM reproduces the finite-volume scheme on network domains (Section 5.2), and whether NBM-OL learns the transient across inlet pressures, inside and outside the training range (Section 5.3).

5.1 SETUP

Networks. We use two topologies with identical pipes ($L = 100\,\mathrm{m}$, $D = 0.5\,\mathrm{m}$, $f = 0.015$, $a = 350\,\mathrm{m/s}$) and discretization (200 cells per edge, $\Delta t = 0.01\,\mathrm{s}$, $T = 10\,\mathrm{s}$); see Table 3. The *7-edge network* is a looped topology with two parallel paths and a cross-link, one inlet, and one outlet. The *tree network* is a 12-edge branching topology adapted from a gas-network schematic (Villalobos-Morales et al., 2003), with one inlet and three outlets over two levels of branching; compressor stations in the schematic are modeled as plain junctions. Outlets are held at $P_{\mathrm{out}} = 900\,\mathrm{Pa}$.

Flow regime. Absolute pressures and pipe lengths are smaller than in transmission pipelines; the dynamics are set by dimensionless ratios. A sound wave crosses an edge $aT/L = 35$ times over the horizon, so the transient is wave-dominated, with repeated reflections at junctions. At the highest inlet pressure tested, the steady-state Mach number reaches roughly 0.5 on the most loaded edges, so momentum advection is not negligible and the problem is nonlinear throughout.

Metrics. We report relative L^2 errors, computed separately for pressure and flux after concatenating all edges, spatial points (cell centers for P, faces for q), and time steps into one vector per variable.

5.2 SINGLE-INSTANCE VERIFICATION

For a fixed boundary condition ($P_{\mathrm{in}} = 1000\,\mathrm{Pa}$), we compare NBM against the finite-volume (FV) solution of the same discretization, which NBM projects, and against a PINN (Raissi et al., 2019) trained on the continuous PDE, boundary, initial, and junction residuals without data (Appendix E). The NBM–FV error measures how well the fixed basis reproduces the scheme; the FV discretization error is small compared with the PINN errors and does not affect that comparison.

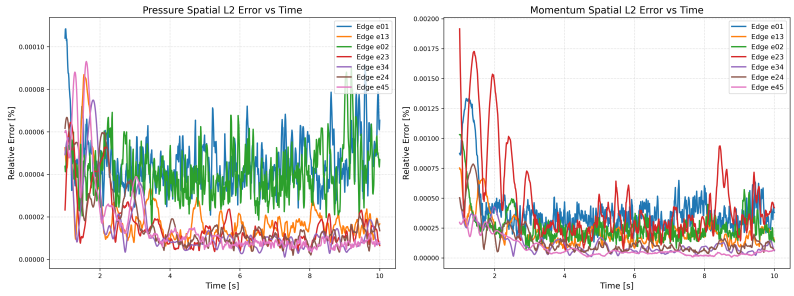
Table 1: Single-instance verification against the finite-volume reference: relative L^2 error over all edges, spatial points, and time steps.

	7-edge network		Tree network	
Method	Pressure	Flux	Pressure	Flux
NBM (ours)	$\mathbf{7.0 \times 10^{-6}}$	$\mathbf{6.9 \times 10^{-5}}$	$\mathbf{1.0 \times 10^{-5}}$	$\mathbf{1.3 \times 10^{-4}}$
PINN	2.0×10^{-2}	8.8×10^{-1}	3.3×10^{-2}	9.2×10^{-1}

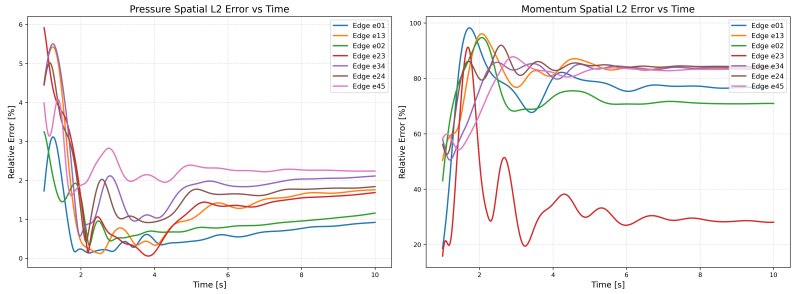
NBM reproduces the scheme to relative errors of 10^{-5} in pressure and 10^{-4} in flux on both networks (Table 1), and the per-edge error stays at this level over the whole transient (Figure 2a). The PINN recovers pressure to a few percent but not the flux, with errors of 88% and 92%: it does not capture the throughflow set up by the transient (Figure 2b).

5.3 OPERATOR LEARNING

Setup. NBM-OL is trained on NBM trajectories at 33 inlet pressures on the 7-edge network and 20 on the tree network, evenly spaced in $[1000, 1800]\,\mathrm{Pa}$ (Table 5). We test at five held-out interpolation points, none of them training pressures (on the tree network, each lies midway between two training pressures), and at two extrapolation points: 1840 and $1880\,\mathrm{Pa}$ on the 7-edge network, and 1820 and $1880\,\mathrm{Pa}$ on the tree network, i.e. 2.5–10% of the training range beyond its upper end. Errors are measured against the NBM reference, which differs from FV by less than 0.02% (Table 1).



(a) NBM vs. Finite Volume



(b) PINN vs. Finite Volume

Figure 2: Per-edge relative L^2 error over time, 7-edge network, against the finite-volume reference: (a) NBM, (b) PINN. The corresponding tree-network plots are in Appendix D.1.

Table 2: Operator learning: relative L^2 error (%) averaged over held-out inlet pressures. Per-point errors are in Appendix D.2.

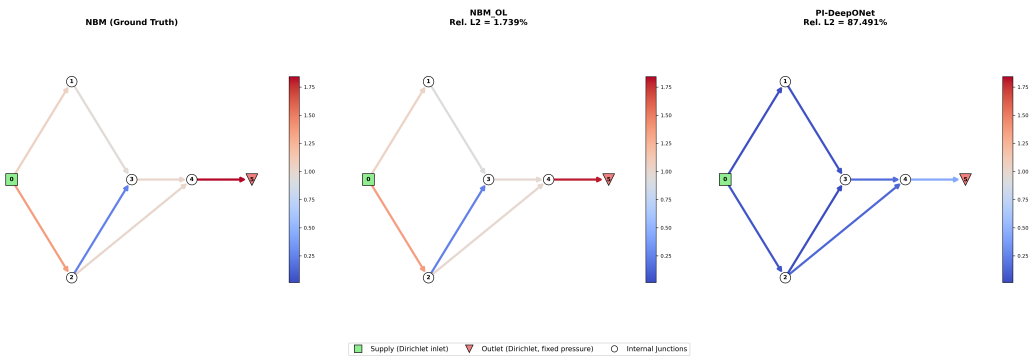
		Interpolation		Extrapolation	
Network	Method	Pressure	Flux	Pressure	Flux
7-edge	NBM-OL (ours)	**0.065**	**1.05**	**0.23**	**1.62**
	PI-DeepONet	7.90	87.3	11.8	85.4
Tree	NBM-OL (ours)	**0.12**	**3.25**	**0.97**	**2.84**
	PI-DeepONet	13.0	95.3	17.9	94.1

Baseline. PI-DeepONet (Wang et al., 2021), adapted to networks with an edge one-hot input to the trunk, is trained on the same inlet-pressure range from PDE, boundary, initial, and junction residuals (Appendix E.6).

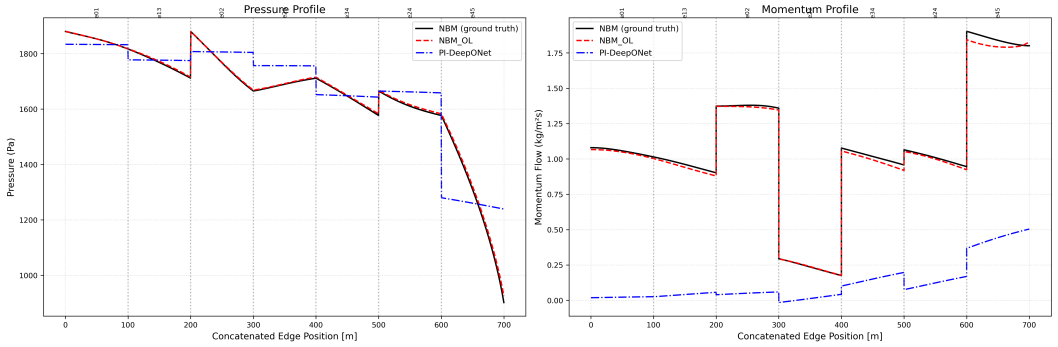
Results. NBM-OL's mean errors are 18–122× lower than PI-DeepONet's in pressure and 29–83× lower in flux across both networks and both regimes (Table 2). In interpolation, pressure errors stay at or below 0.15% at every test point. Flux errors are largest at the lowest test pressures (1.32% at $1215\,\mathrm{Pa}$ on the 7-edge network, 7.15% at $1063.2\,\mathrm{Pa}$ on the tree network) and fall as P_{in} increases. This is consistent with the driving pressure difference $P_{\mathrm{in}} - P_{\mathrm{out}}$, and hence the throughflow, being smallest there, so that a similar absolute error yields a larger relative one.

Under extrapolation, NBM-OL pressure error grows with distance from the training range, most visibly on the tree network, from 0.24% at $+2.5\%$ to 1.69% at $+10\%$. Flux error at $+10\%$ is at most about twice the interpolation mean. Figures 3 and 5 show the $+10\%$ case: NBM-OL follows the transient and the steady state closely, with a small steady-state offset on some edges.

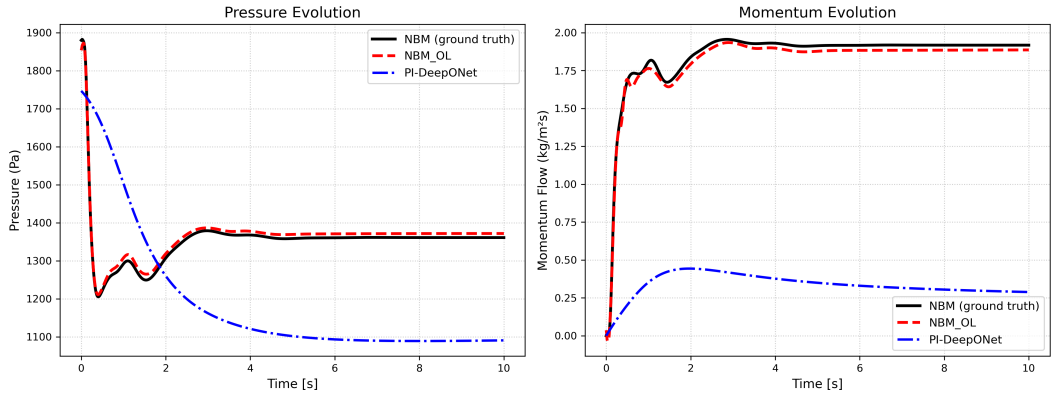
PI-DeepONet reaches flux errors of 85–96% at every test point, in both regimes. Its predicted throughflow remains a small fraction of the true value (Figures 3 and 5), and its pressure profiles are distorted at junctions. As with the PINN, the physics-only objective fails to establish the throughflow, the feature of the solution that the network coupling determines.



(a) Network heatmap, momentum q at $t = 2.0$s



(b) Spatial profile at $t = 2.0$s



(c) Time evolution, edge e45

Figure 3: Extrapolation, 7-edge network, $P_{\text{in}} = 1880\,\text{Pa}$ (10% of the training range beyond its upper end). Reference NBM (black), NBM-OL (red), PI-DeepONet (blue). (a) Flux over the network at $t = 2\,\text{s}$. (b) Pressure and flux profiles along the concatenated edges at $t = 2\,\text{s}$. (c) Time evolution at the midpoint of edge e45. Extrapolation, tree network, $P_{\text{in}} = 1880\,\text{Pa}$ is in Appendix D.3.

6 Conclusions and Future Work

We developed the Neural Basis Method to fluid flow networks. Edgewise neural bases, coupled through junction conditions in a single monolithic projection of a staggered finite-volume scheme, reproduce the discrete solution of the isothermal Euler equations to relative errors of 10^{-5} in pressure and 10^{-4} in flux on looped and branching networks. Building on this representation, NBM-OL maps time and inlet pressure to a POD-compressed latent state, trained with the projected discrete residual along reference trajectories. At held-out inlet pressures, inside the training range and up to 10% of its width beyond it, mean errors stay below 1% in pressure and 3.3% in flux, one to two

orders of magnitude below a physics-informed DeepONet, and a single forward pass returns the network state at any queried time.

Several directions point to future work. *Boundary data*: the learned operator is exercised on a one-parameter family of constant inlet pressures; time-varying and multi-inlet boundary data change only the input to the coefficient network, but remain to be tested. *Scale and regime*: the construction is local by design, since each edge carries its own basis and edges meet only through junction rows, so larger networks enlarge the projected system without changing its structure; applying it at transmission scale, with kilometer-long pipes and megapascal pressures, is a direct next step. *Extrapolation*: the latent basis is built from training trajectories, so predictions outside the training range are limited by its span; our tests extend only 10% beyond it, and pressure error grows with distance. *Reference data*: NBM-OL uses reference coefficient trajectories for its POD basis and for the states entering the training residual. Inference requires neither: a single forward pass at (t, P_{in}) returns the state. Assembling the residual from the model's own predictions would remove the remaining dependence and make the residual computable as a self-check at inference as well.

In our experiments, placing the physics in a fixed, network-aware discretization rather than in a penalty loss yields a surrogate that follows the transient and establishes the throughflow, which the physics-only neural baselines did not. Extending this construction to richer boundary data and larger networks is a step toward fast, reliable surrogates for many-query operation of fluid networks.

Reproducibility Statement

Section 4 specifies the NBM-OL construction: the neural basis and its junction coupling, the weighted least-squares projection, the POD reduction, and the training objective. Appendix A gives the governing equations and the staggered finite-volume discretization, including the LUST reconstruction, boundary treatment, and lagged-coefficient time integration; Appendix B gives the basis construction and the projected coefficient-space system. Network configurations for both topologies are in Table 3, and all NBM-OL training settings – training grid, latent dimension, architecture, optimizer, and schedule – in Table 5. Section 5 states the evaluation protocol, including the held-out P_{in} values for both topologies, which are listed individually in Appendix D.2. Appendices E and E.6 give the architecture, loss formulation (including the nondimensionalization of the PDE residual, equation E.3), collocation-point counts (Table 11), and training hyperparameters (Tables 10 and 12) for the two baselines, applied identically across both topologies except where topology-derived quantities necessarily differ. All runs use a fixed random seed. Code for both network topologies, together with trained model checkpoints, is included as anonymous supplementary material; the repository will be made public upon acceptance.

AI use statement

The core research idea, methodology, experimental design, and scientific findings of this work are the authors' own and were not based on LLM usage.

In this work, we have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, or assisting with translation, and assisting in the writing of proofs, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable to this work. However, we used generative AI tools for creating or editing code, editing formatted tables, and editing and paraphrasing parts of the manuscript text to improve readability. We have reviewed all AI-assisted work. All AI-assisted code was executed, and its output was verified by the authors. The authors take full responsibility for the accuracy and integrity of the manuscript, including all reported results, analysis, and conclusions.

References

Mapundi K Banda, Michael Herty, and Axel Klar. Coupling conditions for gas networks governed by the isothermal euler equations. *Networks Heterog. Media*, 1(2):295–314, 2006a.

Mapundi K Banda, Michael Herty, and Axel Klar. Gas flow in pipeline networks. *Networks and Heterogeneous media*, 1(1):41, 2006b.

Kevin Carlberg, Matthew Barone, and Harbir Antil. Galerkin v. least-squares petrov–galerkin projection in nonlinear model reduction. *Journal of Computational Physics*, 330:693–734, 2017.

Jingrun Chen, Xurong Chi, Zhouwang Yang, et al. Bridging traditional and machine learning-based algorithms for solving pdes: the random feature method. *J. Mach. Learn.*, 1(3):268–298, 2022.

Suchuan Dong and Zongwei Li. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 387:114129, 2021.

Sergey A Dyachenko, Anatoly Zlotnik, Alexander O Korotkevich, and Michael Chertkov. Operator splitting method for simulation of dynamic flows in natural gas pipeline networks. *Physica D: Nonlinear Phenomena*, 361:1–11, 2017.

Vitaliy Gyrya and Anatoly Zlotnik. An explicit staggered-grid method for numerical simulation of large-scale natural gas pipeline networks. *Applied Mathematical Modelling*, 65:34–51, 2019.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: Auto-regressive denoising operator transformer for large-scale PDE pre-training. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17616–17635, 2024. URL https://proceedings.mlr.press/v235/hao24d.html.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel De Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

Michael Herty. Coupling conditions for networked systems of euler equations. *SIAM Journal on Scientific Computing*, 30(3):1596–1612, 2008.

Jan S Hesthaven and Stefano Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

Xianglong Hou, Xinquan Huang, and Paris Perdikaris. Cfo: Learning continuous-time pde dynamics via flow-matched neural operators. In *International Conference on Learning Representations*, volume 2026, pp. 86968–86994, 2026.

Katiana Kontolati, Somdatta Goswami, George Em Karniadakis, and Michael D Shields. Learning nonlinear operators in latent spaces for real-time predictions of complex dynamics in physical systems. *Nature Communications*, 15(1):5101, 2024.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for spatiotemporal surrogate models. *Advances in Neural Information Processing Systems*, 37:119301–119335, 2024.

Andrzej Osiadacz. Simulation of transient gas flows in networks. *International journal for numerical methods in fluids*, 4(1):13–24, 1984.

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.

Saleh Rezaeiravesh, Timofey Mukha, and Mattias Liefvendahl. Systematic study of accuracy of wall-modeled large eddy simulation using uncertainty quantification techniques. *Computers & Fluids*, 185:34–58, 2019.

Erik Laurin Strelow, Alf Gerisch, Jens Lang, and Marc E Pfetsch. Physics informed neural networks: A case study for gas transport problems. *Journal of Computational Physics*, 481:112041, 2023.

Yanet Villalobos-Morales, Diana Cobos-Zaleta, Humberto J Flores-Villarreal, Conrado Borraz-Sánchez, and Roger Z Ríos-Mercado. On nlp and minlp formulations and preprocessing for fuel cost minimization of natural gas transmission networks. In *Proceedings of the 2003 NSF design, service and manufacturing grantees and research conference*, pp. 109, 2003.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

Yuhe Wang and Min Wang. Solving and learning advective multiscale darcian dynamics with the neural basis method. *arXiv preprint arXiv:2602.17776*, 2026.

Hilary Weller. Controlling the computational modes of the arbitrarily structured c grid. *Monthly Weather Review*, 140(10):3220–3234, 2012.

Weidong Wu, Yong Zhang, Lili Hao, Yang Chen, Xiaoyan Sun, and Dunwei Gong. Physics-informed partitioned coupled neural operator for complex networks. *Engineering Applications of Artificial Intelligence*, 159:111567, 2025.

Dimitrije Ždrale, Cassie An Jeng, Katie Wang, Sonia Vanier, Alexandre Bayen, and Hossein Nick Zinat Matin. Hypno: A graph-based neural operator with physics-informed message passing for hyperbolic conservation laws. *arXiv preprint arXiv:2607.20541*, 2026.

APPENDIX

A GOVERNING EQUATIONS AND FV DISCRETIZATION

A.1 CONSERVATIVE FORMULATION

The dynamics of gas flow in a single pipe are governed by the one-dimensional isothermal Euler equations. We write these in conservative form, using fluid density ρ and mass flux m as the primary variables:

$$\frac{\partial \rho}{\partial t} + \frac{\partial m}{\partial x} = 0, \tag{A.1}$$

$$\frac{\partial m}{\partial t} + \frac{\partial}{\partial x}\left(\frac{m^2}{\rho}\right) + a^2 \frac{\partial \rho}{\partial x} + \frac{f|m|m}{2D\rho} = 0. \tag{A.2}$$

Here a is the isothermal speed of sound, D is the pipe diameter, and f is the Darcy friction factor. The pressure gradient term in equation A.2 already reflects the isothermal equation of state $P = \rho a^2$.

A.2 Transformation to Pressure–Flow Variables

Density is not the most convenient variable for network assembly, since pressure is the quantity naturally shared across pipe junctions. We eliminate ρ in favor of P using the isothermal equation of state:

$$\rho = \frac{P}{a^2} \quad \Longrightarrow \quad \frac{\partial \rho}{\partial t} = \frac{1}{a^2}\frac{\partial P}{\partial t}, \qquad \frac{\partial \rho}{\partial x} = \frac{1}{a^2}\frac{\partial P}{\partial x}. \tag{A.3}$$

Substituting equation A.3 into equation A.1–equation A.2 casts the system entirely in (P, q) variable space, where $q \equiv m = \rho u$ is the mass flux per unit cross-sectional area; the same quantity is the momentum per unit volume, which is why it is the natural unknown in equation A.5:

$$\frac{\partial P}{\partial t} + a^2 \frac{\partial q}{\partial x} = 0, \tag{A.4}$$

$$\frac{\partial q}{\partial t} + \frac{\partial P}{\partial x} + \frac{\partial}{\partial x}\left(\frac{a^2 q^2}{P}\right) + \frac{f a^2 |q| q}{2DP} = 0. \tag{A.5}$$

We work with this formulation throughout the paper. Both the advective and friction terms in equation A.5 are singular as $P \to 0$; strictly positive boundary and initial pressures are therefore required to keep the assembled system well-conditioned.

A.3 Network Formulation

The pipeline network is represented as a directed metric graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with each edge $e \in \mathcal{E}$ an interval $[0, L_e]$ on which equation A.4–equation A.5 hold independently. At every internal (non-boundary) vertex $k \in \mathcal{V}$, two coupling conditions are enforced: continuity of pressure across all incident edges, and mass conservation,

$$\sum_{e:\, e \to k} q_e(L_e, t) \;=\; \sum_{e:\, k \to e} q_e(0, t), \tag{A.6}$$

i.e., total incoming momentum flow equals total outgoing momentum flow, with no accumulation at internal vertices. Boundary vertices are held at a prescribed Dirichlet pressure.

A.4 Spatial Discretization

We solve equation A.4–equation A.5 using a finite-volume method on a staggered mesh. Pressure is stored at cell centers, indexed by i. Momentum is stored at cell faces, indexed by j, so that the two boundary faces of an edge coincide with its incident vertices. This choice avoids pressure–velocity decoupling, a well-known failure mode on collocated grids that manifests as checkerboard oscillations in the pressure field.

A.5 Discrete Mass Equation

We integrate the mass equation, equation A.4, over the finite volume centered at pressure node P_i. Its boundary faces coincide with momentum nodes q_j and q_{j+1}. Applying the divergence theorem gives

$$\frac{P_i^{n+1} - P_i^n}{\Delta t} + a^2 \frac{q_{j+1}^{n+1} - q_j^{n+1}}{\Delta x} = 0. \tag{A.7}$$

We collect all t^{n+1} unknowns on the left-hand side. This yields the row contributed by node i to the global linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$:

$$\left(\frac{1}{\Delta t}\right) P_i^{n+1} + \left(\frac{a^2}{\Delta x}\right) q_{j+1}^{n+1} - \left(\frac{a^2}{\Delta x}\right) q_j^{n+1} = \frac{P_i^n}{\Delta t}. \tag{A.8}$$

A.6 Discrete Momentum Equation

We integrate the momentum equation, equation A.5, over the control volume centered at momentum face q_j. This gives

$$\frac{q_j^{n+1} - q_j^n}{\Delta t} + \frac{P_i^{n+1} - P_{i-1}^{n+1}}{\Delta x} + C_{\text{fric}}^n \, q_j^{n+1} + \frac{F_{\text{LUST},i}^{n+1} - F_{\text{LUST},i-1}^{n+1}}{\Delta x} = 0, \tag{A.9}$$

where F_{LUST} is the advective momentum flux, reconstructed as described in Section A.7. Collecting t^{n+1} unknowns gives

$$\left(\frac{1}{\Delta t} + C^n_{\text{fric}}\right) q_j^{n+1} + \left(\frac{1}{\Delta x}\right) P_i^{n+1} - \left(\frac{1}{\Delta x}\right) P_{i-1}^{n+1} + \big[\text{LUST stencil}\big] = \frac{q_j^n}{\Delta t}. \tag{A.10}$$

A.7 LUST Reconstruction

The flux term F_{LUST} requires momentum at pressure-cell centers, where no momentum degree of freedom lives; we reconstruct it from face values using a Linear Upwind Stabilised Transport (LUST) scheme (Weller, 2012), a blend of central differencing (second-order accurate but dispersive near sharp gradients) and second-order linear upwind differencing, which introduces numerical dissipation (Rezaeiravesh et al., 2019). For a lagged advective velocity $u^n_{\text{adv}} \geq 0$, the reconstructed center value is

$$q^{n+1}_{\text{center},i} = \beta \underbrace{\left(\frac{q_j^{n+1} + q_{j+1}^{n+1}}{2}\right)}_{\text{CD}} + (1-\beta) \underbrace{\left(1.5\, q_j^{n+1} - 0.5\, q_{j-1}^{n+1}\right)}_{\text{LUD}}, \tag{A.11}$$

with the stencil reversed for $u^n_{\text{adv}} < 0$ and a reduced two-point stencil at boundary-adjacent cells. We use $\beta = 0.75$, the standard LUST weighting (Rezaeiravesh et al., 2019). The reconstructed flux enters the momentum equation as

$$F^{n+1}_{\text{LUST},i} = u^n_{\text{adv}} \cdot q^{n+1}_{\text{center},i}. \tag{A.12}$$

.

A.8 Boundary Treatment

At each edge endpoint, the momentum equation is discretized over a half-cell (of width $\Delta x/2$) between the boundary face and the first interior cell center, giving a boundary-face pressure-gradient term scaled by $2/\Delta x$ rather than $1/\Delta x$. For an edge endpoint incident to a Dirichlet (boundary) vertex, the corresponding node pressure is substituted as a known constant; for an endpoint incident to an internal vertex, the vertex's own (unknown) pressure is included in the linear system alongside the Kirchhoff constraint (Eq. A.6).

A.9 Time Integration

Spatial discretization of Section A reduces equation A.4–equation A.5 to a semi-discrete system $d\mathbf{U}/dt = \mathbf{F}(\mathbf{U})$, where $\mathbf{U}$ collects all pressure and momentum degrees of freedom. We advance this system with Backward Euler,

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t\, \mathbf{F}(\mathbf{U}^{n+1}), \tag{A.13}$$

.

$\mathbf{F}$ is nonlinear: friction depends on $|q|$, and advection on a velocity computed from the solution itself. Solving equation A.13 exactly would require an inner Newton iteration at every step. We avoid this by linearizing $\mathbf{F}$ around the previous state, freezing its nonlinear coefficients at t^n so that equation A.13 reduces to a linear system in $\mathbf{U}^{n+1}$,

$$\left(\mathbf{I} - \Delta t\, \mathbf{A}^n\right) \mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t\, \mathbf{b}^n, \tag{A.14}$$

where $\mathbf{A}^n$ and $\mathbf{b}^n$ depend only on known, t^n-level quantities.

Two coefficients enter $\mathbf{A}^n$. The friction coefficient,

$$C^n_{\text{fric}} = \frac{f a^2 |q^n|}{2D \bar{P}^n_{\text{face}}}, \tag{A.15}$$

uses $\bar{P}^n_{\text{face}}$, the arithmetic average of the two pressure cells adjacent to the face. The advective velocity,

$$u^n_{\text{adv}} = \frac{a^2 \bar{q}^n_{\text{center}}}{P^n_{\text{center}}}, \tag{A.16}$$

uses $\bar{q}^n_{\text{center}}$, a cell-centered momentum reconstructed from the two adjacent faces.

B THE NEURAL BASIS METHOD

B.1 BASIS REPRESENTATION

The discretization described in Section A produces one unknown per physical grid location: one pressure value per cell, one momentum value per face. For a network of any practical size, this yields a prohibitively large state vector. The Neural Basis Method reduces this dimensionality by representing each field as a linear combination of a small number of fixed spatial basis functions, with only the combination coefficients evolving in time:

$$P^{n+1}(x) \approx \mathbf{\Phi}_P(x)\,\mathbf{c}_P^{n+1}, \qquad q^{n+1}(x) \approx \mathbf{\Phi}_q(x)\,\mathbf{c}_q^{n+1}. \tag{B.1}$$

Here $\mathbf{\Phi}_P$ and $\mathbf{\Phi}_q$ are fixed, precomputed basis matrices evaluated at the pressure cell centers and momentum faces respectively (constructed as described in Section B.2), and $\mathbf{c}_P^{n+1}$, $\mathbf{c}_q^{n+1}$ are the time-dependent coefficient vectors we solve for at each step, in place of the full physical state. The number of coefficients is chosen to be much smaller than the number of physical degrees of freedom; the resulting compression is what makes network-scale simulation tractable.

B.2 BASIS CONSTRUCTION

For each field (pressure and momentum), a shared local basis is obtained by evaluating a fixed, randomly initialized neural network at the field's grid locations (cell centers for pressure, faces for momentum) on the edge's normalized domain $[0, 1]$, then orthonormalizing the resulting evaluations via QR decomposition.

Because every edge shares the same length, spatial resolution, and non-dimensionalized domain, this local basis is computed once and reused across all edges, placed in a block-diagonal arrangement within the global basis matrix $\mathbf{\Phi}$: edge e's pressure coefficients occupy their own dedicated block of columns of $\mathbf{\Phi}$, multiplying only that edge's pressure degrees of freedom, and likewise for momentum. Internal junction vertices are not represented by a spatial basis expansion (a single point has no spatial extent to expand), and instead receive one dedicated coefficient each, entering $\mathbf{\Phi}$ as a single unit entry in its own column, so that an internal vertex's coefficient equals its pressure directly.

B.3 PROJECTED DISCRETE EQUATIONS

We obtain the coefficient-space equations by substituting the basis representation, equation B.1, directly into the assembled finite volume equations of Section A. This substitution is performed independently at every cell center i and every momentum face j.

Substituting into the discrete mass equation, equation A.8, at center i gives

$$\left[\frac{1}{\Delta t}\,\mathbf{\Phi}_P(x_i)\right]\mathbf{c}_P^{n+1} + \left[\frac{a^2}{\Delta x}\left(\mathbf{\Phi}_q(x_{j+1}) - \mathbf{\Phi}_q(x_j)\right)\right]\mathbf{c}_q^{n+1} = \frac{P_i^n}{\Delta t}. \tag{B.2}$$

Substituting into the discrete momentum equation, equation A.10, at face j gives

$$\left[\frac{\mathbf{\Phi}_P(x_i) - \mathbf{\Phi}_P(x_{i-1})}{\Delta x}\right]\mathbf{c}_P^{n+1} + \left[\left(\frac{1}{\Delta t} + C_{\text{fric}}^n\right)\mathbf{\Phi}_q(x_j)\right]\mathbf{c}_q^{n+1} = \frac{q_j^n}{\Delta t}. \tag{B.3}$$

Junction rows (Eq. A.6) and boundary rows (Section A.8) are projected identically, substituting $\mathbf{\Phi}_q$ (and, at an internal vertex, that vertex's own unit basis column) in place of the raw physical unknowns.

B.4 GLOBAL COEFFICIENT-SPACE SYSTEM

Collecting the coefficients into a single unknown vector $\mathbf{x} = [\mathbf{c}_P^{n+1}; \mathbf{c}_q^{n+1}]^\top$, the assembled rows from equation B.2 and equation B.3, taken over the entire network, form the global block system

$$\begin{bmatrix}\mathbf{A}_{PP} & \mathbf{A}_{Pq} \\ \mathbf{A}_{qP} & \mathbf{A}_{qq}\end{bmatrix}\begin{bmatrix}\mathbf{c}_P^{n+1} \\ \mathbf{c}_q^{n+1}\end{bmatrix} = \begin{bmatrix}\mathbf{b}_P \\ \mathbf{b}_q\end{bmatrix}, \tag{B.4}$$

with block entries defined row-wise by

$$\mathbf{A}_{PP} = \frac{1}{\Delta t}\,\mathbf{\Phi}_P(x_i), \tag{B.5}$$

$$\mathbf{A}_{Pq} = \frac{a^2}{\Delta x}\,\big(\mathbf{\Phi}_q(x_{j+1}) - \mathbf{\Phi}_q(x_j)\big), \tag{B.6}$$

$$\mathbf{A}_{qP} = \frac{1}{\Delta x}\,\big(\mathbf{\Phi}_P(x_i) - \mathbf{\Phi}_P(x_{i-1})\big), \tag{B.7}$$

$$\mathbf{A}_{qq} = \frac{1}{\Delta t}\,\mathbf{\Phi}_q(x_j) + C^n_{\text{fric}}\,\mathbf{\Phi}_q(x_j) + \text{[LUST terms]}. \tag{B.8}$$

The bracketed LUST terms in equation B.8 are the projection of the advective stencil: substituting equation A.11 into equation A.12 and then into the flux difference of equation A.9 gives a linear combination of $\mathbf{\Phi}_q$ evaluated at the faces neighbouring x_j, with coefficients set by β, u^n_{adv}, and the upwind direction.

Because equation B.4 is assembled from one row per physical cell and face, but only a small number of coefficients are being solved for, the resulting system is rectangular and overdetermined: the number of rows equals the number of physical degrees of freedom in the original finite volume discretization, while the number of columns equals the number of basis coefficients, chosen to be much smaller. No exact solution exists in general; we instead solve equation B.4 in the weighted least-squares sense,

$$\mathbf{x}^{n+1} = \arg\min_{\mathbf{x}} \big\|\mathbf{W}\big(\mathbf{A}\mathbf{x} - \mathbf{b}\big)\big\|_2^2, \tag{B.9}$$

where $\mathbf{W}$ is a diagonal weighting matrix, with a substantially larger weight assigned to boundary and junction rows than to interior rows. This allows the solver to prioritize satisfying boundary conditions and junction mass conservation over the interior rows of equation B.4.

B.5 Network Instantiations

Both the finite-volume reference solver and the Neural Basis Method are applied, without modification to the governing equations or numerical scheme, to two network topologies differing only in graph structure and boundary configuration, summarized in Table 3.

Table 3: Network configurations for the two topologies. Both use identical pipe parameters ($a = 350$ m/s, $D = 0.5$ m, $f = 0.015$) and per-edge resolution ($L = 100$ m, $\Delta x = 0.5$ m, $N = 200$ cells/edge).

	7-edge network	Tree network
Edges $\lvert\mathcal{E}\rvert$	7	12
Vertices $\lvert\mathcal{V}\rvert$	6	13
Topology	2 parallel paths, 1 cross-link	Single-inlet branching tree
Dirichlet inlets	1 (node 0)	1 (node 0)
Dirichlet outlets	1 (node 5)	3 (nodes 7, 8, 12)
Internal (junction) vertices	4	9
Δt	0.01 s	0.01 s
T_{final}	10 s	10 s

Table 4: NBM solver configuration.

	7-edge	Tree
Basis functions per field per edge	40	40
Basis network	2-layer residual net, widths [128, 40], $\tanh$	
Boundary/junction penalty weight	10^4	10^4
Grid degrees of freedom	2811	4821
NBM coefficients K	564	969

C NBM-OL IMPLEMENTATION

Data generation. For each network we run the NBM solver (Section 4) once per training inlet pressure on a uniform grid over $[1000, 1800]$ Pa: 33 inlet pressures for the 7-edge network and 20 for the tree network. Each run advances $T/\Delta t = 1000$ steps with $\Delta t = 0.01$ s from the shut-in initial state, storing the coefficient vector $c(t_n, P_{\text{in}}) \in \mathbb{R}^K$ at every step together with the basis matrix Φ, the weight vector $\mathrm{diag}(W)$, and the degree-of-freedom map that records which rows and columns belong to each edge's pressure and flux blocks. Held-out test pressures are excluded from this grid and their reference trajectories are generated by separate runs of the same solver.

Latent basis. The stored coefficient trajectories are concatenated across all training inlet pressures into a snapshot matrix of size $(n_{P_{\text{in}}} \cdot 1001) \times K$, mean-centred, and reduced by a thin singular value decomposition; the leading r right singular vectors form V_r. The matrix includes the initial state at $t = 0$, while the 1000 subsequent states per trajectory form the training pairs. We use $r = 35$ for both networks, which retains more than 99.99% of the coefficient-space energy on each.

Initial-state coefficient. The shut-in state is uniform pressure P_{in} with zero flux. We compute it once as the least-squares projection $c_1 = (\Phi^\top \Phi)^{-1} \Phi^\top e_P$ of the unit-pressure, zero-flux grid state e_P (ones on pressure degrees of freedom, zeros on flux), and set $c_{\text{init}}(P_{\text{in}}) = P_{\text{in}}\, c_1$. The same expression is used in data generation and inside the model, so the predicted state at $t = 0$ is consistent with the reference by construction.

Precomputing the projected operators. For every training pair (t_{n+1}, P_{in}) we reconstruct the reference grid state $u^n = \Phi\, c_{\text{ref}}(t_n, P_{\text{in}})$, assemble the discrete operator A^n and right-hand side b^n from it exactly as in the solver, apply the row weights W, and store

$$A_z^n = W A^n \Phi V_r \in \mathbb{R}^{m \times r}, \qquad b_z^n = W\big(b^n - A^n \Phi\, c_{\text{init}}(P_{\text{in}})\big) \in \mathbb{R}^m, \tag{C.1}$$

where m is the number of rows of the discrete system (2811 for the 7-edge network, 4821 for the tree). This is done once, before training, in single precision. The resulting tensors occupy 13.0 GB and 13.5 GB respectively.

Training. g_θ is a three-layer perceptron (two hidden layers of width 128 with $\tanh$, linear output of width r) taking normalized time and inlet pressure. Its output is scaled componentwise by z_{scale}, the standard deviation of the training latents $V_r^\top (c_{\text{ref}} - c_{\text{init}})$ offset by one, which puts all latent components on a comparable scale at initialization. Training uses Adam at learning rate 10^{-3} with batches of 16 training pairs drawn in a fresh random permutation each epoch, and gradients clipped at max-norm 20. Each batch evaluates the residual by a batched matrix-vector product $A_z^n \delta z$, so no operator is ever assembled during training. We keep the checkpoint with the lowest epoch-mean training residual. Full settings are in Table 5.

Evaluation. At a queried (t, P_{in}) the model returns $c_{\text{pred}} = c_{\text{init}}(P_{\text{in}}) + V_r\, \delta z$ in one forward pass; the grid state follows as Φc_{pred}. Relative L^2 errors are computed separately for pressure and flux by selecting the corresponding degree-of-freedom indices from the reconstructed state and comparing against the NBM reference trajectory at the same inlet pressure over all 1000 stored times.

Table 5 lists the training configuration of NBM-OL for both networks.

Table 5: NBM-OL training hyperparameters.

	7-edge network	Tree network
Training range of P_{in}	$[1000, 1800]$	$[1000, 1800]$
Training inlet pressures $n_{P_{\text{in}}}$	33 (spacing 25)	20 (spacing ≈ 42.1)
Time steps per trajectory	1000	1000
Training pairs (t_{n+1}, P_{in})	33,000	20,000
NBM coefficients K	564	969
Latent dimension r	35	35
Stored latent operators A_z^n	13.0 GB	13.5 GB
MLP g_θ	2 hidden layers, width 128, $\tanh$	
Latent scaling z_{scale}	std. of training latents $+ 1$	
Optimizer	Adam, initial learning rate 10^{-3}	
Batch size	16	
Epochs	3000	2000
LR schedule (ReduceLROnPlateau, factor 0.5)	patience 500, min 10^{-5}	patience 300, min 10^{-6}
Gradient clipping (max norm)	20	
Checkpoint selection	lowest training residual/latest	

D Additional Results

D.1 Tree Network Verification

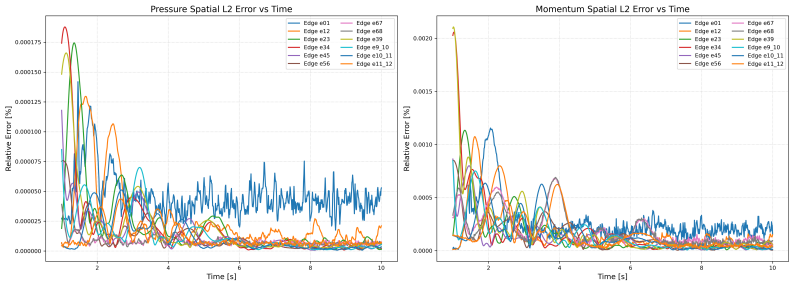
Figure 4 shows the per-edge relative L^2 error over time for the tree network, the counterpart of Figure 2; both are plotted from $t = 1\,\text{s}$, after the initial expansion wave has crossed the network. NBM remains below $2 \times 10^{-4}\%$ in pressure and $2 \times 10^{-3}\%$ in flux on every edge, decaying to roughly $5 \times 10^{-5}\%$ and $2 \times 10^{-4}\%$ once the transient passes. The PINN errors are four to five orders of magnitude larger, and the two variables behave differently: pressure error peaks near 11% around $t = 2\,\text{s}$ and then decays to below 2%, whereas flux error stays between 60% and 100% on every edge for the entire horizon. The PINN therefore recovers the pressure field approximately while never establishing the throughflow, the same failure mode seen on the 7-edge network.

D.2 Per-Point Operator-Learning Errors

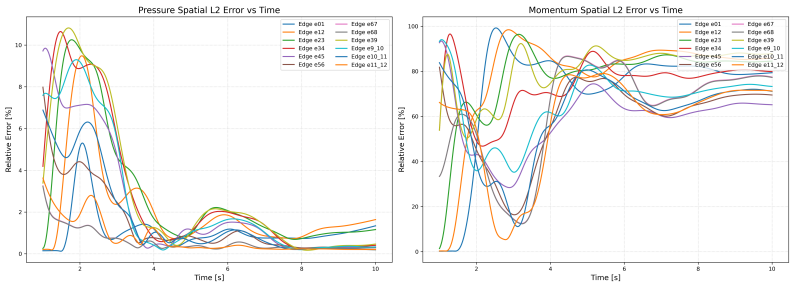
Tables 6–9 report the relative L^2 error at each held-out inlet pressure; their means appear in Table 2. Each entry is computed over all edges, spatial points, and time steps of a single trajectory, obtained from one forward pass of NBM-OL per queried time. Three patterns are visible across the four tables. First, NBM-OL's pressure error varies little within the training range, staying between 0.048% and 0.081% on the 7-edge network and between 0.097% and 0.145% on the tree network, and grows outside it, most steeply at $1880\,\text{Pa}$ on the tree network (1.69%). Second, its flux error is largest at the lowest test pressures, 1.32% at $1215\,\text{Pa}$ and 7.15% at $1063.2\,\text{Pa}$, and falls as P_{in} increases before rising again slightly at the top of the range; the smallest driving pressure difference $P_{\text{in}} - P_{\text{out}}$ occurs at the lowest test point, so the same absolute error there corresponds to a larger relative one. Third, PI-DeepONet shows the opposite trend in pressure: its error grows monotonically with P_{in}, from 5.24% to 10.57% on the 7-edge network and from 6.49% to 17.74% on the tree network, while its flux error stays near saturation at 85–96% at every point in both regimes. Its pressure predictions are constrained to $[P_{\text{out}}, 1.2\,P_{\text{in}}^{\max}]$ by construction (Appendix E.6), so the growth with P_{in} reflects an increasing deviation within that interval rather than a boundary-condition violation. The tree network's test pressures each lie midway between two training pressures on the 20-point grid, so no interpolation point coincides with a training value.

D.3 Tree Network Extrapolation

Figure 5 shows the tree network at $P_{\text{in}} = 1880\,\text{Pa}$, 10% of the training range beyond its upper end, the same case as Figure 3 for the 7-edge network. NBM-OL reproduces the pressure drop along the inlet chain and across both branches, and reaches the correct steady-state flux on each outlet branch;



(a) NBM vs. Finite Volume



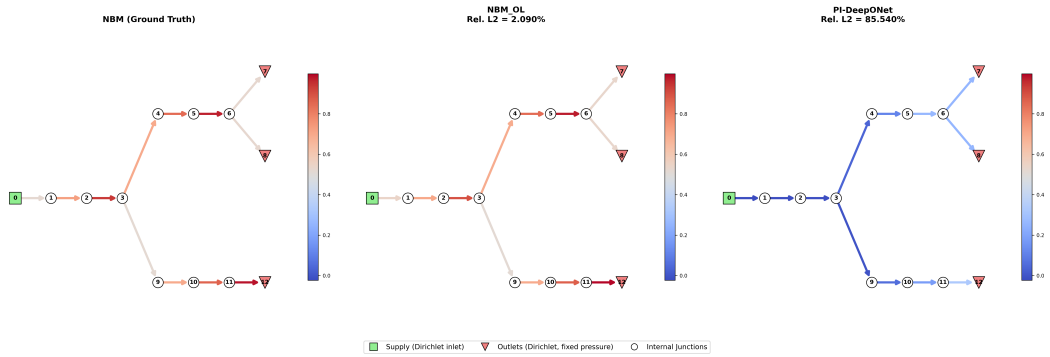
(b) PINN vs. Finite Volume

Figure 4: Per-edge relative L^2 error over time, tree network, against the finite-volume reference: (a) NBM, (b) PINN.

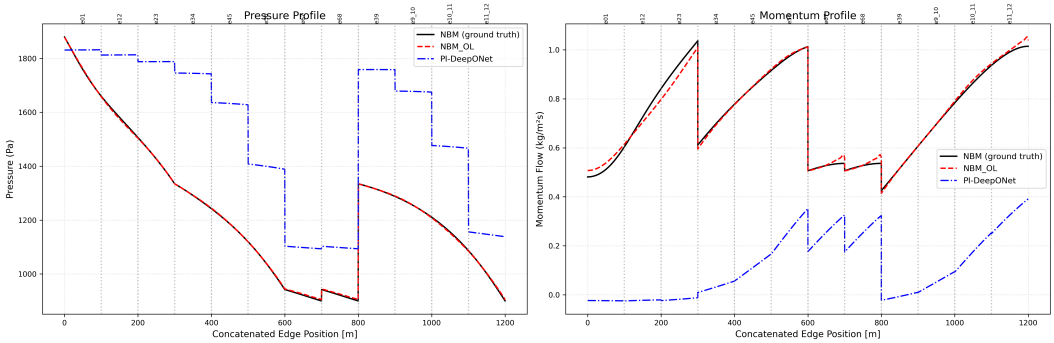
Table 6: Interpolation error (%) at each held-out inlet pressure, 7-edge network (training range $[1000, 1800]$ Pa).

	NBM-OL		PI-DeepONet	
P_{in}	Pressure	Flux	Pressure	Flux
1215	**0.0483%**	**1.3165%**	5.2368%	89.4339%
1365	**0.0574%**	**0.9894%**	6.6521%	88.0012%
1415	**0.0605%**	**0.9398%**	7.0646%	87.5787%
1715	**0.0772%**	**0.9546%**	9.9673%	85.7612%
1765	**0.0811%**	**1.0283%**	10.5748%	85.5879%
Mean	**0.0649%**	**1.0457%**	7.8991%	87.2726%

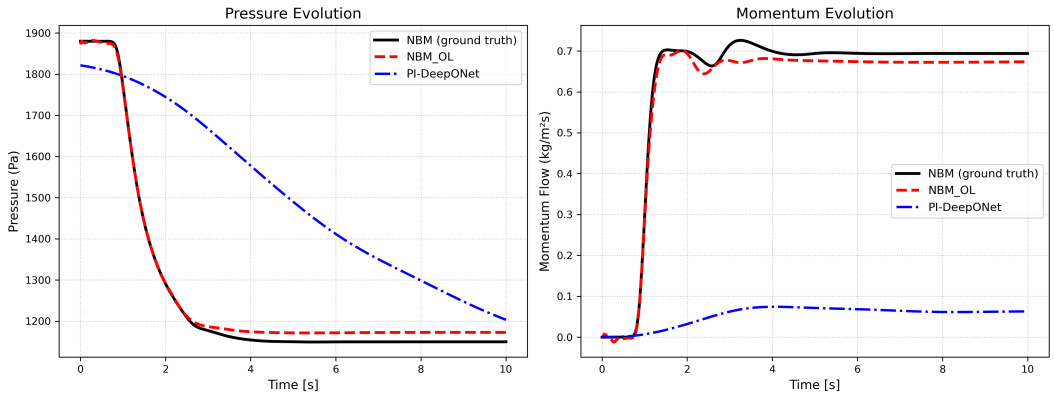
the visible discrepancy is a small steady-state offset on edge e34, where the predicted flux settles slightly below the reference. PI-DeepONet under-predicts the flux throughout by close to an order of magnitude and produces a pressure profile with spurious jumps at the branch points, consistent with its 94% mean flux error in Table 9.



(a) Network heatmap, momentum q at $t = 2.0$s



(b) Spatial profile at $t = 2.0$s



(c) Time evolution, edge e34

Figure 5: Extrapolation, tree network, $P_{\text{in}} = 1880\,\text{Pa}$ (10% of the training range beyond its upper end). Reference NBM (black), NBM-OL (red), PI-DeepONet (blue). (a) Flux over the network at $t = 2\,\text{s}$. (b) Pressure and flux profiles along the concatenated edges at $t = 2\,\text{s}$. (c) Time evolution at the midpoint of edge e34.

Table 7: Extrapolation error (%) at each held-out inlet pressure above the training range, 7-edge network.

	NBM-OL		PI-DeepONet	
P_{in}	Pressure	Flux	Pressure	Flux
1840	$\mathbf{0.1253}\%$	$\mathbf{1.2055}\%$	11.5568%	85.3996%
1880	$\mathbf{0.3354}\%$	$\mathbf{2.0254}\%$	12.1181%	85.3338%
Mean	$\mathbf{0.2304}\%$	$\mathbf{1.6154}\%$	11.8374%	85.3667%

Table 8: Interpolation error (%) at each held-out inlet pressure, tree network (training range $[1000, 1800]$ Pa). Each test pressure lies midway between two training pressures.

	NBM-OL		PI-DeepONet	
P_{in}	Pressure	Flux	Pressure	Flux
1063.2	$\mathbf{0.1230}\%$	$\mathbf{7.1528}\%$	6.4937%	96.4810%
1231.6	$\mathbf{0.0973}\%$	$\mathbf{3.6356}\%$	10.4083%	96.0434%
1400.0	$\mathbf{0.1139}\%$	$\mathbf{1.9213}\%$	13.9584%	95.2578%
1568.4	$\mathbf{0.1296}\%$	$\mathbf{1.5229}\%$	16.3645%	94.6084%
1736.8	$\mathbf{0.1454}\%$	$\mathbf{2.0224}\%$	17.7417%	94.2131%
Mean	$\mathbf{0.1218}\%$	$\mathbf{3.2510}\%$	12.9933%	95.3207%

Table 9: Extrapolation error (%) at each held-out inlet pressure above the training range, tree network.

	NBM-OL		PI-DeepONet	
P_{in}	Pressure	Flux	Pressure	Flux
1820	$\mathbf{0.2374}\%$	$\mathbf{2.3682}\%$	17.9332%	94.0869%
1880	$\mathbf{1.6927}\%$	$\mathbf{3.3212}\%$	17.8690%	94.0200%
Mean	$\mathbf{0.9650}\%$	$\mathbf{2.8447}\%$	17.9011%	94.0534%

E PHYSICS-INFORMED NEURAL NETWORK BASELINE

We compare against a physics-informed neural network (PINN), trained at a fixed boundary condition ($P_{\text{in}} = 1000$, $P_{\text{out}} = 900$) matching the finite-volume reference and NBM solution used in the single-instance benchmark. The PINN follows the standard formulation of Raissi et al. (2019): a fully-connected network trained by minimizing a weighted sum of PDE-residual, boundary-condition, initial-condition, and junction-coupling losses via automatic differentiation, with no solution data. Loss weights are held fixed throughout training, consistent with Strelow et al. (2023), who find that adaptive loss-balancing schemes offer no benefit over well-chosen constant weights for gas transport problems specifically.

E.1 ARCHITECTURE

The PINN baseline learns a single, network-wide map $(\text{edge}, x, t) \mapsto (P, q)$, implemented as a feed-forward multilayer perceptron. The edge is encoded as a one-hot vector of length $|\mathcal{E}|$, concatenated with normalized position $x/L \in [0, 1]$ and normalized time $t/T_{\text{final}} \in [0, 1]$, giving an input of dimension $|\mathcal{E}| + 2$. The network has 4 hidden layers of width 128 with $\tanh$ activations, followed by a linear output layer producing two raw scalars.

The raw outputs are mapped to physical pressure and momentum flow via

$$P = P_{\text{out}} + (P_{\text{in}} - P_{\text{out}})\,\sigma(\hat{o}_1), \tag{E.1}$$

$$q = Q_{\text{ref}}\,\hat{o}_2, \tag{E.2}$$

where $\hat{o}_1, \hat{o}_2$ are the two raw network outputs and $\sigma(\cdot)$ is the logistic sigmoid. equation E.1 constrains predicted pressure to the physical interval $[P_{\text{out}}, P_{\text{in}}]$ by construction. Q_{ref} in equation E.2 is not a hand-tuned constant, but derived directly from the problem's physical scales (Section E.2).

E.2 PDE RESIDUAL

For collocation points sampled per edge, spatial and temporal derivatives of P and q are obtained via automatic differentiation with respect to the normalized inputs, rescaled by $1/L$ and $1/T_{\text{final}}$ respectively to recover physical-unit derivatives. The residuals of equation A.4–equation A.5 are evaluated directly at each collocation point, with pressure clamped to a small positive floor before entering the advective and frictional terms, consistent with the singularity discussed in Section A.2.

The raw residuals are non-dimensionalized before entering the loss. Writing $P = P_{\text{ref}}\hat{P}$, $q = Q_{\text{ref}}\hat{q}$, $x = L\hat{x}$, $t = T_{\text{final}}\hat{t}$ and substituting into the mass equation, the scale Q_{ref} that makes the resulting advective coefficient equal to 1 is fixed by the other, already-physical scales:

$$Q_{\text{ref}} = \frac{P_{\text{ref}}\,L}{a^2\,T_{\text{final}}}. \tag{E.3}$$

Dividing the raw mass and momentum residuals by $P_{\text{ref}}/T_{\text{final}}$ and $Q_{\text{ref}}/T_{\text{final}}$ respectively is equivalent to substituting the hat variables into the PDE directly, without separately hardcoding the resulting dimensionless groups.

One resulting dimensionless group, $(a\,T_{\text{final}}/L)^2 \approx 1225$ for this problem's parameters, is intrinsically large: the acoustic wave transit distance ($a\,T_{\text{final}} \approx 3500\,\text{m}$) far exceeds the domain length ($100\,\text{m}$), so the pressure-gradient term dominates the momentum residual even after correct nondimensionalization. This reflects the problem's genuinely fast wave dynamics rather than a scaling artifact, and the momentum residual is correspondingly harder to drive down than the mass residual throughout training.

E.3 LOSS TERMS

In addition to the PDE residual, three supervised loss terms enforce the network formulation of Section A.3. All pressure-mismatch terms are normalized by P_{in}^2 and all flow-mismatch terms by Q_{ref}^2, bringing them to the same numeric scale as the dimensionless PDE residual; without this normalization, a fixed loss weight cannot make progress against raw, unnormalized-scale terms, and training collapses to the trivial solution $q \equiv 0$, $P \equiv \text{const}$.

Boundary loss. For every edge endpoint incident to a Dirichlet vertex k with prescribed pressure P_k,

$$\mathcal{L}_{\text{bc}} = \sum_{k \in \mathcal{V}_{\text{Dirichlet}}} \sum_{e\,:\,k \in e} \mathbb{E}\left[\left(\frac{P_e(x_k,t) - P_k}{P_{\text{in}}}\right)^2\right].$$

Initial-condition loss. At $t = 0$, the network is driven toward the shut-in state (uniform pressure at $\max_k P_k$, zero flow):

$$\mathcal{L}_{\text{ic}} = \sum_{e \in \mathcal{E}} \mathbb{E}\left[\left(\frac{P_e(x,0) - P_{\text{shut-in}}}{P_{\text{in}}}\right)^2 + \left(\frac{q_e(x,0)}{Q_{\text{ref}}}\right)^2\right].$$

Junction loss. At every internal vertex k, pressure continuity is enforced by penalizing deviation from the mean pressure across incident edges, and mass conservation (equation A.6) is enforced directly as a soft penalty:

$$\mathcal{L}_{\text{junction}} = \sum_k \left[\sum_{e\,:\,k \in e} \mathbb{E}\left[\left(\frac{P_e(x_k,t) - \bar{P}_k(t)}{P_{\text{in}}}\right)^2\right] + \mathbb{E}\left[\left(\frac{\sum_{e:e\to k} q_e(L_e,t) - \sum_{e:k\to e} q_e(0,t)}{Q_{\text{ref}}}\right)^2\right]\right].$$

The total training loss is a weighted sum,

$$\mathcal{L} = w_{\text{pde}}\,\mathcal{L}_{\text{pde}} + w_{\text{bc}}\,\mathcal{L}_{\text{bc}} + w_{\text{ic}}\,\mathcal{L}_{\text{ic}} + w_{\text{junction}}\,\mathcal{L}_{\text{junction}},$$

with $w_{\text{bc}} = w_{\text{ic}} = w_{\text{junction}} = 10$ and $w_{\text{pde}} = 1$ throughout.

E.4 Training Procedure

The network is trained for 20,000 epochs with the Adam optimizer, learning rate 10^{-3} decayed via ReduceLROnPlateau (factor 0.5, patience 1000), gradient clipping at max norm 20.0, and loss weights $w_{\text{pde}} = 1.0$, $w_{\text{bc}} = w_{\text{ic}} = w_{\text{junction}} = 10.0$. Collocation points are resampled independently every epoch for the PDE residual, boundary, initial, and junction terms.

E.5 Network Instantiations

The PINN architecture, loss formulation, and training procedure are applied unmodified to both network topologies; only the topology-derived quantities (edge count, Dirichlet vertex set, and the resulting junction structure) differ, as summarized in Table 3. All other hyperparameters (architecture size, loss weights, optimizer settings) are held identical across both networks, summarized in Table 10.

Table 10: PINN training hyperparameters, identical across both network topologies.

Hyperparameter	Value
Hidden layers	4
Hidden width	128
Activation	$\tanh$
Optimizer	Adam
Learning rate	10^{-3}
LR schedule	ReduceLROnPlateau (factor 0.5, patience 1000, min 10^{-5})
Gradient clipping (max norm)	20
Collocation points per term	256
Loss weights ($w_{\text{pde}}, w_{\text{bc}}, w_{\text{ic}}, w_{\text{junction}}$)	$1, 10, 10, 10$
Training epochs	20,000
Random seed	0

E.6 PI-DeepONet Baseline: Architecture and Training Details

We benchmark against a physics-informed DeepONet (PI-DeepONet), adapted from the collocation-based training paradigm of Wang et al. (2021) to the graph-structured pipeline networks. PI-DeepONet is trained with no supervised trajectory data: all training signal comes from PDE, boundary-condition, initial-condition, and junction residuals evaluated via automatic differentiation at collocation points resampled every epoch. The same architecture and training procedure are applied to both network topologies without modification; only the topology-derived quantities (edge count, boundary-incident edges, junction count) differ, as summarized below.

Architecture. PI-DeepONet uses a branch-trunk architecture with latent width $p = 64$. The branch network maps the (normalized) scalar boundary condition P_{in} to a $2p$-dimensional embedding via a 3-layer MLP with hidden width 128 and $\tanh$ activations. The trunk network takes a concatenated input of [edge one-hot ($\in \mathbb{R}^{|\mathcal{E}|}$), normalized spatial coordinate x/L, normalized time t/T_{final}] and similarly outputs a $2p$-dimensional embedding via a 3-layer MLP with hidden width 128. Branch and trunk embeddings are split into two p-dimensional halves, combined via inner product plus a learned bias, and mapped to pressure P and flow q outputs. The edge one-hot encoding is a necessary adaptation of the original single-domain PI-DeepONet formulation to a multi-edge network domain, since the trunk net must distinguish which edge a given (x, t) query refers to.

Pressure output is constrained via $P = P_{\text{out}} + (P_{\text{in}}^{\max} - P_{\text{out}}) \cdot 1.2\,\sigma(\hat{P})$, where $\hat{P}$ is the raw network output, σ is the sigmoid function, and $P_{\text{in}}^{\max}$ is the fixed upper end of the training range (not the sampled P_{in} of the current query, which would give every training example a differently sized output window and a P_{in}-dependent error). Flow output is given by $q = Q_{\text{ref}}\,\hat{q}$, where $\hat{q}$ is the raw network output and Q_{ref} is derived from the problem's physical scales rather than a hand-tuned constant, as detailed below.

Training objective. The total loss is a weighted sum of four residual terms, evaluated at collocation points resampled every epoch:

$$\mathcal{L} = w_{\text{pde}}\mathcal{L}_{\text{pde}} + w_{\text{bc}}\mathcal{L}_{\text{bc}} + w_{\text{ic}}\mathcal{L}_{\text{ic}} + w_{\text{junction}}\mathcal{L}_{\text{junction}},$$

with $w_{\text{bc}} = w_{\text{ic}} = w_{\text{junction}} = 10$ and $w_{\text{pde}} = 1$.

PDE residual. Mass and momentum conservation in (P, q) form, evaluated at interior collocation points $(x, t) \sim U[0, 1]^2$, $P_{\text{in}} \sim U[P_{\text{in,min}}, P_{\text{in,max}}]$, for each edge independently:

$$r_{\text{mass}} = \partial_t P + a^2 \partial_x q,$$
$$r_{\text{mom}} = \partial_t q + \partial_x P + \partial_x\left(\frac{a^2 q^2}{P}\right) + \frac{f a^2 |q| q}{2DP}.$$

The raw residuals are non-dimensionalized before entering the loss, following the same derivation as the plain-PINN baseline: writing $P = P_{\text{ref}}\hat{P}$, $q = Q_{\text{ref}}\hat{q}$, $x = L\hat{x}$, $t = T_{\text{final}}\hat{t}$ and substituting into the mass equation, the scale that makes the resulting advective coefficient equal to 1 is

$$Q_{\text{ref}} = \frac{P_{\text{ref}}\, L}{a^2\, T_{\text{final}}},$$

with $P_{\text{ref}} = P_{\text{in}}^{\max}$, the same fixed constant used for the pressure output constraint above. Dividing the raw residuals by $P_{\text{ref}}/T_{\text{final}}$ and $Q_{\text{ref}}/T_{\text{final}}$ respectively gives the non-dimensionalized residuals $\hat{r}_{\text{mass}}, \hat{r}_{\text{mom}}$ entering the loss: $\mathcal{L}_{\text{pde}} = \sum_{\text{edges}} \mathbb{E}[\hat{r}_{\text{mass}}^2 + \hat{r}_{\text{mom}}^2]$. Without this normalization, the residual is dominated by the coefficient $a^2 = 122500$, several orders of magnitude larger than the already-normalized boundary, initial-condition, and junction terms below, and training collapses to the trivial solution $q \equiv 0$, $P \equiv$ const, which trivially satisfies the (unnormalized) interior residual while violating the boundary data. As in the plain-PINN baseline, one resulting dimensionless group, $(a\, T_{\text{final}}/L)^2 \approx 1225$, is intrinsically large for this problem's parameters, so the momentum residual remains harder to drive down than the mass residual even after correct non-dimensionalization.

Boundary condition loss. At every edge endpoint incident to the inlet vertex ($x = 0$): $P = P_{\text{in}}$ (the sampled value). At every edge endpoint incident to an outlet vertex ($x = 1$): $P = P_{\text{out}}$ (fixed). A vertex with multiple incident edges contributes one evaluation per edge, since each edge's prediction

is an independent network output with no shared degree of freedom across edges. Both pressure-mismatch terms are normalized by $(P_{\text{in}}^{\max})^2$.

Initial condition loss. At $t = 0$, all edges: $P = P_{\text{in}}$, $q = 0$ (shut-in state, matching the NBM/FV initial condition exactly). The pressure term is normalized by $(P_{\text{in}}^{\max})^2$ and the flow term by Q_{ref}^2.

Junction loss. At each internal junction node: pressure continuity across all incident edges, plus mass conservation (net inflow = net outflow). The pressure-continuity term is normalized by $(P_{\text{in}}^{\max})^2$ and the mass-conservation term by Q_{ref}^2.

Collocation points. Table 11 summarizes the collocation points sampled per loss term, per epoch, for each topology. All points are freshly resampled every epoch; no fixed training set exists, so no discrete P_{in} value is ever repeated across training. Held-out interpolation points therefore lie within the domain $[P_{\text{in,min}}, P_{\text{in,max}}]$ that the physics losses were enforced over continuously, but were never seen as discrete supervised targets.

Table 11: Collocation points sampled per training epoch for PI-DeepONet. 7-edge: 2 inlet-incident edges, 1 outlet-incident edge, 4 junctions. Tree: 1 inlet-incident edge, 3 outlet-incident edges, 9 junctions.

Loss term	7-edge network	Tree network
PDE residual	$256 \times 7 = 1792$	$256 \times 12 = 3072$
BC loss	$256 \times 3 = 768$	$256 \times 4 = 1024$
IC loss	$256 \times 7 = 1792$	$256 \times 12 = 3072$
Junction loss	$256 \times 4 = 1024$	$256 \times 9 = 2304$

Hyperparameters. Table 12 summarizes training hyperparameters, identical across both topologies.

Table 12: PI-DeepONet training hyperparameters, identical across both network topologies.

Hyperparameter	Value
P_{in} range	$[1000, 1800]$
P_{out} (fixed)	900.0
T_{final}	10.0 s
Epochs	20,000
Collocation points per instance (n_{colloc})	256
Latent width p	64
Branch / trunk hidden width	128
Network depth	3 layers
Optimizer	Adam
Learning rate	1×10^{-3}
LR schedule	ReduceLROnPlateau (factor 0.5, patience 1000, min 10^{-6})
Gradient clipping (max norm)	20
Loss weights ($w_{\text{pde}}, w_{\text{bc}}, w_{\text{ic}}, w_{\text{junction}}$)	$1, 10, 10, 10$
Seed	0