TUNABLE NON-LINEAR MANIFOLD ROMS FOR PDES VIA MATRIX-FREE PETROV–GALERKIN PROJECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The cost of a full-order PDE solve grows quickly as the mesh is refined, and neural operators such as Fourier Neural Operators and DeepONets generally provide only one accuracy-efficiency trade-off per trained model, and adjusting it requires retraining the model. We present a nonlinear-manifold reduced-order model (NM-ROM) whose query time grows far more slowly with the mesh than the full-order solver's, so its speedup grows as the mesh is refined, and whose balance between accuracy and speed is chosen when the model is used, not when it is trained. Its *spatial bank*, a set of learned spatial functions, is ordered once by importance after training, so a query chooses how many bank functions to use, and whether to solve with a nonlinear head or in the linear span of the bank, without changing a learned weight. For Burgers, Poisson and heat, the trained network can be used unchanged on meshes much finer than those it was trained on, while our POD-LSPG baseline has to be rebuilt for each mesh. The model is a coordinate-network bank with exact Dirichlet enforcement, solved by least-squares Petrov–Galerkin projection of the discrete residual onto fixed tests. Every linear term is precomputed, the solver runs matrix-free in JAX, and the nonlinear term uses empirical quadrature. On viscous Burgers and three-dimensional Navier–Stokes it is up to $27.5\times$ faster than the full-order solver, and on linear Poisson and heat problems $108\times$ and $216\times$ faster at 0.83 % and 0.13 % worst-case error. On Burgers, refining the mesh from 256^2 to 4096^2 raises its query time $2.6\times$ and the solver's $74\times$. At the same latent dimension it is also more accurate than POD-LSPG and a quadratic manifold.

1 INTRODUCTION

Two lines of work aim to make partial differential equation (PDE) solutions fast to compute: neural operators and reduced-order models (ROMs), and both have advanced quickly. Neural operators (Li et al., 2021; Lu et al., 2021; Kovachki et al., 2023) now include attention-based models for general geometries (Wu et al., 2024; Luo et al., 2025) and pretrained PDE foundation models and scalable transformer backbones (Herde et al., 2024; Hao et al., 2024; Holzschuh et al., 2025), and are fast to evaluate once trained (Azizzadenesheli et al., 2024). In parallel, nonlinear-manifold ROMs (NM-ROMs) have moved from dense autoencoders (Lee and Carlberg, 2020; Y. Kim et al., 2022) to domain-decomposed (Diaz et al., 2024), neural-field (Chen et al., 2023; M. Kim et al., 2024; Berman and Peherstorfer, 2024) and hyper-reduced Navier–Stokes models (Romor et al., 2025), while keeping the projection of the governing equations used by linear projection-based ROMs (Benner et al., 2015; Carlberg et al., 2011). Hesthaven et al. (2026) survey these nonlinear methods for transport-dominated problems.

Two properties decide whether such a model can stand in for a full-order solver: whether accuracy can be traded for speed when the model is used, and how fast the cost of a given accuracy grows as the mesh is refined. Full-order methods (FOMs) with iterative solvers such as conjugate gradients (Hestenes and Stiefel, 1952) have the first property, since their tolerance and time step are set per query (each new problem instance to solve), but their cost grows quickly with the mesh. A trained neural operator gives one trade-off between accuracy and wall-clock time at a given mesh, and changing it in general requires retraining (de Hoop et al., 2022); comparisons against weak numerical baselines can also overstate their speed (McGreivy and Hakim, 2024). Linear ROMs can require many basis functions for moving fronts (Cohen and DeVore, 2015; Peherstorfer, 2022), and a POD basis built on one grid is usually tied to that grid. NM-ROMs are more compact, but their online

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.

nonlinear solve can offset the benefit of fewer unknowns. We ask: *can one trained model have both properties, an accuracy choice at query time and a cost that grows slowly with the mesh?*

We answer this question for viscous Burgers in 2D and incompressible Navier–Stokes in 3D, and evaluate the same framework on linear Poisson and heat equations. Our NM-ROM builds the solution from a *spatial bank*, a set of R functions of space, each a small neural network of the coordinates, whose weighted sum gives the solution; the weights come from a small nonlinear *head*. After training, the bank is ordered once by how much of the training data each of its functions carries. A query then chooses how many of the ordered functions to use, R', and whether to solve with the nonlinear head or in the linear span of those functions, without changing a learned weight. Because each bank function is a small network of the coordinates, the same trained network can be evaluated on any mesh. For Burgers, Poisson and heat we train it once on coarse meshes and use it unchanged up to 4096^2 in 2D and up to 256^3 in 3D, while the POD basis of our POD-LSPG baseline is rebuilt from new full-order solves for every mesh. For Navier–Stokes the bank is itself a POD basis, built at each mesh. All online coefficients come from the PDE residual, without access to the reference solution.

We report the following results, each from a single trained model per problem family. On 2D viscous Burgers at 4096^2, the accurate setting reaches 1.35% worst error at $10.7\times$ the speed of Newton–BiCGStab and the fast setting 3.11% at $27.5\times$, on 64 held-out test cases (Table 1). On 3D Navier–Stokes the nonlinear head reaches 0.21% at $8.01\times$ the speed of the pseudo-spectral solver at 96^3. On the linear Poisson and heat equations the ordered span is $108\times$ faster than conjugate gradients at 4096^2 (0.83% error) and $216\times$ faster than Crank–Nicolson CG at 2048^2 (0.13%).

Contributions.

1. **Speed that scales with the mesh on nonlinear PDEs:** On 2D Burgers up to 4096^2 and 3D Navier–Stokes, a frozen model is up to $27\times$ faster than the full-order solver at the finest meshes, and it is both more accurate and faster than four neural operators on Burgers at 1024^2 and 2048^2 and on Navier–Stokes at 64^3 and 96^3. At the same latent dimension it is more accurate than POD-LSPG, a quadratic manifold and a reproduced NM-ROM.
2. **Accuracy chosen at query time from one trained NM-ROM:** After training, we order the spatial bank once by importance. A query then chooses how many of the ordered bank functions to use (the width R') and whether to solve in their linear span or with the nonlinear head, without changing a learned weight. Widening the span raises both accuracy and cost, so one trained model gives a whole accuracy–cost ladder instead of a single operating point. Which setting is most accurate depends on the PDE: the full-width span for Burgers, Poisson and heat, and the nonlinear head for Navier–Stokes, where it is more accurate than every span width we tested.
3. **An architecture built for a fast, mesh-independent solve:** The decoder has no encoder: a bank of coordinate networks, zero on the Dirichlet boundary, multiplied by a small head with a linear skip. Each bank function can be evaluated at any single node, so the same trained bank serves every mesh and empirical quadrature visits only a few nodes; with every linear term precomputed, the matrix-free JAX solver's query time grows far more slowly with the mesh than the full-order solver's (§5.1).

2 Related Work

Where our method sits. Two families of methods make PDE solutions cheaper to compute, and they differ in whether they solve the PDE online. *Projection-based reduced-order models* build a trial space offline from full-order solutions and, for each query, solve the discretised PDE restricted to that space. *Neural operators* learn the map from inputs to solutions and return a forward pass of the trained network, without evaluating the PDE. Our method is a projection-based nonlinear-manifold ROM, solved by least-squares Petrov–Galerkin projection; we compare against both families.

Projection-based ROMs. Linear ROMs approximate solutions in a fixed subspace (Benner et al., 2015; Sirovich, 1987); moving fronts can require a large basis (Cohen and DeVore, 2015; Greif and Urban, 2019; Peherstorfer, 2022). Nonlinear-manifold ROMs replace the subspace by a learned manifold and keep the residual projection (Lee and Carlberg, 2020; Y. Kim et al., 2022; Romor et al., 2023; Diaz et al., 2024), including quadratic manifolds (Geelen et al., 2022; Barnett and Farhat, 2022), neural-network corrections to a linear basis (Barnett et al., 2023), adaptive bases

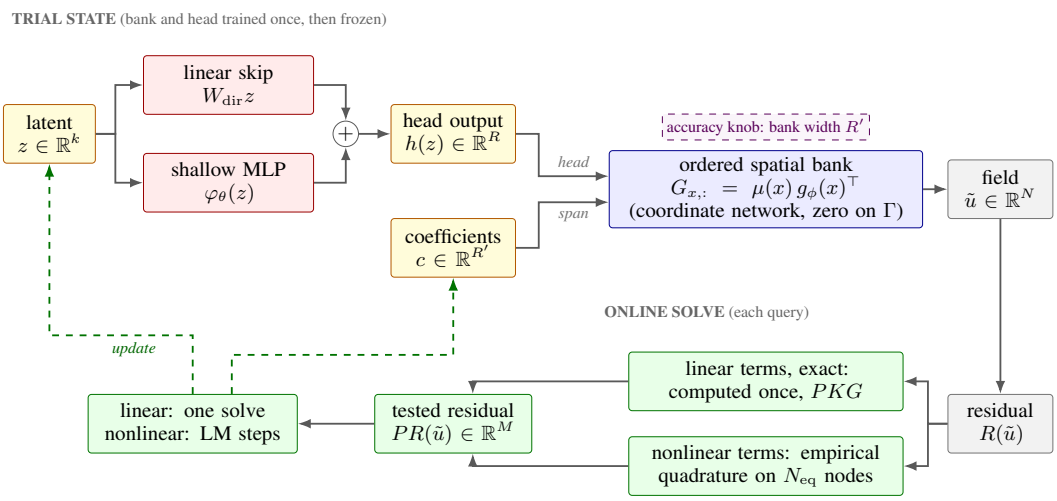

Figure 1: NM-ROM model and online solve. **Top, trained once and frozen:** the latent code z passes through a linear skip and a shallow SiLU network to give the bank coefficients $h(z)$, and the ordered coordinate-network bank, zero on Γ, turns them into the field $\tilde{u}$. A query uses the head, $\tilde{u} = Gh(z)$, or the span of the first R' ordered columns, $\tilde{u} = \widehat{G}_{R'} c$. **Bottom, each query:** the residual is tested against M fixed tests P, with linear terms from the precomputed PKG and advection from empirical quadrature on N_{eq} nodes. A linear PDE in the span is one least-squares solve; otherwise Levenberg–Marquardt steps update z or c. Dashed tags mark deployment choices.

(Peherstorfer and Willcox, 2015; Carlberg, 2015), hyper-reduced models of Navier–Stokes flows (Romor et al., 2025) and graph autoencoders on unstructured meshes (Magargal et al., 2025). These methods solve the PDE online, but a linear or quadratic basis is tied to the mesh it was built on, and an autoencoder decoder is evaluated at every node in each online iteration unless it is hyper-reduced. Latent-dynamics ROMs instead learn the reduced dynamics from data (Fries et al., 2022; He et al., 2023). Least-squares Petrov–Galerkin projection (Carlberg et al., 2011) and preassembly of reduced operators (Ştefănescu and Sandu, 2014; Weder et al., 2024) are established components that we reuse. Our closest relatives are neural-field ROMs (Chen et al., 2023; M. Kim et al., 2024; Berman and Peherstorfer, 2024), which also evaluate a coordinate network at individual nodes. We differ in three ways: our decoder separates space from the latent code (a bank times a head), so every linear term is precomputed exactly; the bank carries an exact Dirichlet factor; and the bank is ordered after training, so its nested widths give accuracy settings at query time from one trained model, as runtime-tunable networks do for vision and language models (Yu et al., 2019; Cai et al., 2020; Kusupati et al., 2022; Devvrit et al., 2024).

Neural operators. FNO (Li et al., 2021), DeepONet (Lu et al., 2021), U-Net (Ronneberger et al., 2015; Takamoto et al., 2022) and Transolver (Wu et al., 2024; Luo et al., 2025) learn solution maps, as do neural-field operators (Serrano et al., 2023) and pretrained PDE foundation models (Herde et al., 2024; Hao et al., 2024). Their resolution or inference procedure can change accuracy and cost; our accuracy setting instead changes the trial space in which the PDE residual is solved.

Hyper-reduction and baselines. Empirical quadrature (Hernández et al., 2017; Yano and Patera, 2019; Hernández et al., 2024) reduces nonlinear residual evaluation through non-negative weights; we retain that construction and test its error on the states the solver reaches. Because learned solvers are often compared with weak baselines (McGreivy and Hakim, 2024), we compare against named iterative full-order solvers.

3 Methodology

A full-order solve has one unknown per grid node, so its cost grows with the mesh size N. For each query we instead solve the same discretised PDE in a few unknowns, at a cost that barely depends on N. This needs three ingredients. First, a compact set of candidate solutions, the trial manifold (§3.1): the solution is a weighted sum of the functions in the spatial bank, and its weights are either produced

by the nonlinear head from a few latent unknowns or solved for directly on the first R' bank functions. Second, a way to choose the right candidate without knowing the answer (§3.2). Third, a way to evaluate the PDE without visiting all N nodes (§3.3): linear terms are computed once offline, and the nonlinear term is sampled at a few nodes (in three dimensions, it is evaluated with a precomputed quadratic tensor). Figure 1 shows the whole pipeline. Throughout, $u \in \mathbb{R}^N$ is the full-order discrete state on a uniform grid and $K \in \mathbb{R}^{N\times N}$ is the sparse finite-difference matrix of $-\Delta$.

3.1 Trial Manifold with Exact Dirichlet Enforcement

Solutions of one PDE family share their spatial structure, so a few learned functions of space can describe all of them; the nonlinear head lets their combination follow features, such as moving fronts, that a small fixed combination misses. We do not use an encoder. The decoder writes the state as a fixed set of spatial functions times coefficients produced by a small network,

$$\tilde{u}(z) = \mathcal{D}(z) = G\,h(z), \qquad G_{x,:} = \mu(x)\,g_\phi(x)^\top, \qquad \mu(x) = 16\,x_1(1-x_1)\,x_2(1-x_2). \quad (1)$$

Each column of the bank $G \in \mathbb{R}^{N\times R}$ is one spatial function. The function comes from a small network g_ϕ that takes the coordinates of a point and returns a value there, so a column is simply that network evaluated at every grid node. The head $h : \mathbb{R}^k \to \mathbb{R}^R$ takes the latent state z and returns one weight per column. Because μ is zero on the boundary Γ, every column of G is zero there too. The boundary values therefore never change when z changes, on any mesh, and the boundary equations do not enter the solve. The cube and the L-shaped domain define their own boundary factor (Appendix A), zero on their boundaries in the same way. Each query then solves for z (head) or c (span) by solving the PDE (§3.2).

Ordering the bank. Training does not rank the columns of G: any one of them may matter as much as any other, so keeping only the first R' columns would throw away useful ones at random. After training we therefore replace the columns by new combinations of them, sorted from most to least important,

$$\widehat{G} = G\,T. \quad (2)$$

The matrix T comes from a principal component analysis of the training fields written in the bank (Appendix A.1). The first column of $\widehat{G}$ is the single direction that best describes the training fields, the second bests the remaining directions, and so on. Keeping the first R' columns therefore gives the best R'-dimensional subspace of the bank for the training data. At full width nothing changes: $\widehat{G}$ spans the same space as G, and no weight is retrained. A query then solves in one of two ways,

$$\tilde{u} = G\,h(z), \;\; z \in \mathbb{R}^k \quad \text{(head)}, \qquad \tilde{u} = \widehat{G}_{R'}\,c, \;\; c \in \mathbb{R}^{R'} \quad \text{(span)}, \quad (3)$$

where $\widehat{G}_{R'}$ holds the first $R' \le R$ columns. The head is a nonlinear manifold with k unknowns; the span is a linear space with R' unknowns. The width R' is the accuracy knob. The columns are nested, so narrowing the span needs no retraining, and cheaper to solve and to convert back to a field.

3.2 Projecting the PDE onto the Manifold

At query time the solution is unknown but the PDE is known, so we choose the candidate that best satisfies the discretised equations; no solution data is needed at query time. Each PDE we consider gives a discretised residual $R(u)$ that vanishes at the discrete solution. We substitute the trial manifold into R and, instead of pulling the residual back through the decoder Jacobian, test it against M fixed vectors, the rows of $P \in \mathbb{R}^{M\times N}$. On the square these are the M lowest sine modes, which are eigenvectors of K. A head query solves

$$z^\star = \arg\min_z \; \tfrac{1}{2}\,\|P\,R\big(\tilde{u}(z)\big)\|_2^2; \quad (4)$$

a span query solves the same problem in c. We choose M with the width: $M \ge k$ for the head and $M \ge R'$ for the span ($M = 4R'$ for Burgers). This is least-squares Petrov–Galerkin projection (Carlberg et al., 2011; Lee and Carlberg, 2020) with fixed rather than state-dependent tests; each row of PR is also scaled by a fixed factor that puts the tests on a common scale (Appendix A); we leave this scaling out of the notation. We solve (4) by damped Gauss–Newton (Levenberg–Marquardt) (Marquardt, 1963),

$$\Delta z = -\big(J^\top J + \lambda\,\mathrm{diag}(J^\top J)\big)^{-1} J^\top P\,R\big(\tilde{u}(z)\big), \qquad J = P\,\frac{\partial R}{\partial u}\,J_{\mathcal{D}}(z), \quad (5)$$

and keep a step only if the residual decreases. Here $J_{\mathcal{D}}(z) = G\,J_h(z)$ for the head and $J_{\mathcal{D}} = \widehat{G}_{R'}$, a constant, for the span. Unlike $J_{\mathcal{D}}(z)^\top$, the tests P do not change with z, which is what lets every linear term be computed once (§3.3). For Poisson in the span, (4) is a linear least-squares problem solved once, and for heat in the span each step is linear; the iteration is needed only for the head and for the nonlinear PDEs.

Elliptic (Poisson). The Poisson residual is $R(u) = Ku - F$, where F is the discrete source. In the span it is linear in c,

$$P\,R\big(\widehat{G}_{R'}\,c\big) = \big(PK\widehat{G}_{R'}\big)\,c - PF, \tag{6}$$

so a query is one least-squares solve with a fixed $M \times R'$ matrix, factorised once per mesh and width, and nothing is iterated. With the head, (5) iterates in z with $J = PKG\,J_h(z)$, and only the head is re-evaluated.

Time-dependent problems. Heat (Crank–Nicolson), Burgers (backward Euler with a sign-upwind stencil) and Navier–Stokes (implicit midpoint on the periodic cube) each give a residual $R(u_{n+1})$ for one time step; their forms are in Appendix A. Each step solves

$$z_{n+1} = \arg\min_z \|P\,R\big(\tilde{u}(z)\big)\|_2, \qquad u_n = \tilde{u}(z_n), \tag{7}$$

by (5), warm-started from the previous step; the span solves the same problem in c_{n+1}. The heat residual is linear, so in the span the rollout applies one fixed $R' \times R'$ matrix per step. Burgers and Navier–Stokes add a nonlinear advection term $A(u)$, so both the head and the span iterate (5) at every step. For Navier–Stokes the tests P are divergence-free Fourier modes, so the pressure drops out of PR; because the vortices travel, the bank is evaluated in a frame centred on the flow whose shift is solved at each step (Appendix A.5).

3.3 Hyper-reduction

The manifold reduces the unknowns from N to k or R', but the tested residual $P\,R(\tilde{u})$ still visits every node, so each evaluation costs $O(NR)$, which would erase the wall-clock benefit. We remove this cost in two ways, one for linear terms and one for the nonlinear term.

Linear terms: computed once and exact. A linear term is tested once, offline. For the Laplacian,

$$PK\tilde{u} = \big(PKG\big)\,h(z), \qquad PKG \in \mathbb{R}^{M\times R}, \tag{8}$$

and the span uses $PK\widehat{G}_{R'}$, the first R' columns of $PKGT$. Online a linear term costs $O(MR)$ and is exact. For Poisson and heat this is the whole residual, so their reduced solves make no approximation beyond the trial space.

The nonlinear term: Empirical Quadrature. The advection terms of Burgers and Navier–Stokes cannot be computed once. We use Empirical Quadrature (Hernández et al., 2017; Yano and Patera, 2019) to select a non-negative per-node weight vector w, independent of the state, such that

$$PA(\tilde{u}) = \sum_{i=1}^{N} P_{:,i}\,A_i(\tilde{u}) \;\approx\; \sum_{i\in\mathcal{S}} w_i\,P_{:,i}\,A_i(\tilde{u}). \tag{9}$$

Letting $g_i^{(\ell)} = P_{:,i}A_i(\tilde{u}^{(\ell)}) \in \mathbb{R}^M$ denote the per-node contribution at a stored state $\tilde{u}^{(\ell)}$ that the solver reaches on training trajectories, the constraint at that state is $\sum_i w_i\,g_i^{(\ell)} = PA(\tilde{u}^{(\ell)})$. Stacking the constraints gives a matrix E and a vector b, and we solve

$$\min_{w\ge 0} \|E\,w - b\|_2 \tag{10}$$

by NNLS (Lawson and Hanson, 1974), which returns a sparse w with support $\mathcal{S}$ of size $N_{\text{eq}} \ll N$. Computing A_i needs $\tilde{u}$ at node i and its stencil neighbours $\mathcal{N}(i)$, so we evaluate the decoder only on $\mathcal{S}^+ = \mathcal{S} \cup \bigcup_{i\in\mathcal{S}} \mathcal{N}(i)$, with $|\mathcal{S}^+| \le 5N_{\text{eq}}$ in two dimensions and $7N_{\text{eq}}$ in three. Because each bank column is a coordinate network (§3.4), those rows of the bank are computed once and cached.

Cost. With linear terms computed once and advection on $\mathcal{S}^+$, one iteration of (5) costs

$$\mathcal{O}\big(MR\,(N_{\text{eq}} + k)\big), \tag{11}$$

with k replaced by R' in the span, plus the cost of the head and of the small normal-equation solve, none of which depends on N. Returning the field at an output time is one further $O(NR')$ product.

Checking the rule. Each rule is checked on the states the reduced solver reaches on held-out trajectories, against an error bar fixed in advance; every rule behind Table 1 passes (Appendix A.6).

3.4 Model Architecture

Three requirements of the solve shape the model in Figure 1. First, every head solve starts from a stored training code and descends along the head's Jacobian, so that Jacobian must retain a usable direction at all times; the linear skip guarantees this. Second, empirical quadrature keeps only a sparse set of nodes, which saves time only if the decoder can be evaluated at one node without the others –and one trained model must serve every mesh of a series; the coordinate-network bank meets both. Third, one trained model must offer several accuracy settings without retraining; the *ordered bank* provides this. Widths and training settings are in Appendix C.

Linear skip, for the latent solve. The head is a shallow network plus a linear map,

$$h(z) \;=\; W_{\mathrm{dir}} z \;+\; \varphi_\theta(z), \qquad \varphi_\theta(z) = W_3\,\sigma\big(W_2\,\sigma(W_1 z + b_1) + b_2\big) + b_3, \tag{12}$$

with σ the SiLU activation. Its Jacobian is $J_h(z) = W_{\mathrm{dir}} + J_\varphi(z)$. The first term does not depend on z, so the Levenberg–Marquardt step from a stored code always has a fixed linear direction to follow, while φ_θ supplies the curvature of the manifold. The head acts on the R bank coefficients, not on the N nodes, so its cost and its parameter count do not depend on the mesh.

Coordinate-network bank, for node-local evaluation. Each column of the bank is a coordinate network read at the node and multiplied by the boundary factor of (1),

$$G_{x,r} = \mu(x)\,g_{\phi,r}(x), \qquad g_\phi(x) = \mathrm{MLP}_\phi\big(\gamma(x)\big), \quad \gamma(x) = \big[\cos(2\pi\Omega x),\ \sin(2\pi\Omega x)\big], \tag{13}$$

where γ is a random-Fourier-feature embedding with a fixed random frequency matrix Ω (Tancik et al., 2020). Because g_ϕ reads a coordinate, a row of G is computed from its node alone. A quadrature rule therefore needs only the rows of $\mathcal{S}^+$, and the same trained network is sampled on every mesh. Two natural alternatives fail one of these tests. A convolutional or attention decoder couples its output nodes, so one node costs a pass over the whole grid, and a learned dense readout stores NR parameters and must be retrained for each mesh. The parameter count of g_ϕ does not depend on N.

Ordered bank, for deployment-time accuracy. The ordering (2) makes the settings nested, so one model built with R columns contains every smaller one, and the two solve modes of (3) are its two ends: the span is a linear ROM in a learned bank, and the head is a nonlinear-manifold ROM in k unknowns that can follow states no small span contains.

3.5 Implementation and Training

This subsection describes how a query runs on the GPU and how the model is trained. The matrix PKG is built once per mesh, so a linear term costs one small matrix–vector product online, and in the span the Jacobian is the fixed matrix $PK\widehat{G}_{R'}$. With the head, the remaining factor is the head Jacobian, supplied by forward-mode automatic differentiation (Bradbury et al., 2018): $J = PKG\,J_h(z)$ with $J_h v = \mathrm{JVP}(h, z, v)$. Each accepted step solves the small damped normal equation directly, and advection reads the cached stencil on the quadrature nodes (Appendix A.4). The ordered bank is stored in column blocks at the tested widths, so a query at width R' reads exactly R' columns. Training uses training data only: the coordinate network g_ϕ is fitted to the training states through (1) and frozen; the head is then fitted to the bank coefficients of those states, with one stored code per training state (an auto-decoder); last, the bank is ordered once by (2), with one QR and one SVD. Widths, cohorts and offline costs are in Appendix C.

4 Experimental Setup and Benchmark Problems

Problems and references. We test the method on PDEs of increasing difficulty: two linear problems, Poisson and heat, and two nonlinear ones, viscous Burgers and incompressible Navier–Stokes. In two dimensions we solve Poisson on the square, with an L-shaped domain as a check with a re-entrant corner, heat and Burgers; in three dimensions Poisson, heat and Burgers on the cube and Navier–Stokes on the periodic cube. Meshes and cohorts are in Table C.3, the sampled families in Table C.4 and the three-dimensional configurations in Table C.8.

Table 1: NM-ROM against the named full-order solver (FOM) at every problem and mesh. **Both NM-ROM columns of a row come from one frozen model**: *accurate* is the most accurate tested setting and *fast* the cheapest tested setting whose worst error is below 5 % (except $R' = 64$, 5.6 %, on the L-shape, Poisson on an L-shaped domain); settings and FOM tolerances are in Table C.5. Error is the worst relative L^2 error over the cases (%). Speedup is the FOM's median time divided by the NM-ROM's in the same GPU allocation, **bold** where the NM-ROM is faster; the FOM is a standard iterative solver (CG, CN–CG, Newton–BiCGStab or CNAB2) run at the loosest tested setting that is at least as accurate as the accurate setting; looser settings we tested were less accurate than the NM-ROM, so the FOM error is often below ours. Every row is on held-out test cases that were not used to choose the accurate setting.

Two-dimensional							*Three-dimensional*						
		Accurate		Fast		FOM			Accurate		Fast		FOM
Problem	Mesh	Err.	Speedup	Err.	Speedup	Err.	Problem	Mesh	Err.	Speedup	Err.	Speedup	Err.
Poisson	256^2	0.84	**18.2**×	4.07	**27.0**×	0.77	Poisson	32^3	0.23	**4.42**×	2.01	**5.25**×	0.16
	1024^2	0.83	**31.1**×	4.06	**82.0**×	0.33		64^3	0.23	**8.07**×	1.98	**11.8**×	0.11
	2048^2	0.83	**64.4**×	4.06	**221**×	0.22		128^3	0.23	**13.6**×	1.97	**28.5**×	0.075
	4096^2	0.83	**108**×	4.06	**372**×	0.58		256^3	0.23	**19.3**×	1.96	**47.6**×	0.17
L-shape	256^2	2.02	**32.1**×	5.61	**34.7**×	1.08	Heat	32^3	0.080	**10.9**×	1.11	**15.2**×	0.078
	512^2	2.00	**50.1**×	5.60	**58.5**×	0.74		64^3	0.082	**8.50**×	4.78	**18.0**×	0.081
	1024^2	1.99	**41.6**×	5.60	**49.4**×	1.77		128^3	0.083	**5.79**×	4.78	**20.9**×	0.082
	2048^2	1.99	**78.4**×	5.60	**97.5**×	1.18		256^3	0.083	**7.60**×	4.78	**28.0**×	0.082
Heat	1024^2	0.13	**105**×	1.08	**200**×	0.045	Burgers	64^3	1.01	0.21×	3.37	**2.08**×	0.92
	2048^2	0.13	**216**×	1.08	**512**×	0.045		128^3	1.46	0.57×	4.26	**5.10**×	1.39
	4096^2	0.13	**231**×	1.08	**491**×	0.054		256^3	2.20	**2.19**×	3.77	**5.60**×	1.52
Burgers	256^2	0.87	0.38×	3.85	**1.22**×	0.11	Navier–Stokes	32^3	0.21	0.93×	3.06	**3.56**×	0.19
	512^2	1.13	0.62×	4.10	**1.93**×	0.12		64^3	0.21	**2.92**×	3.06	**8.53**×	0.12
	1024^2	1.25	**1.51**×	4.25	**3.87**×	0.13		96^3	0.21	**8.01**×	3.24	**14.4**×	0.056
	2048^2	1.32	**3.39**×	3.06	**8.72**×	0.13							
	4096^2	1.35	**10.7**×	3.11	**27.5**×	0.14							

Accuracy. We report the worst relative L^2 error over the stated cases and requested times, in percent. Poisson uses the steady reference norm and heat the current reference norm; Burgers and Navier–Stokes use the initial-field norm over evolved times. Same-grid error measures the reduction error against the converged discrete solution. Settings for a test cohort are fixed before its cases are opened; all other cohorts are development cohorts.

Timing and speedup. Every accuracy–runtime pair comes from the same solver invocation, in double precision, with GPU burn-in, synchronised timing and medians of retained repetitions (protocol in Appendix C.1). GPU query time covers initialisation, the solve or rollout and the requested device outputs. Training and compilation are offline costs. Speedup is $S = T_{\mathrm{FOM}}/T_{\mathrm{method}}$ with both times from the same allocation. Solves that miss their stopping rule are reported as such and do not enter a speedup.

Baselines. Poisson and the L-shape are compared with conjugate gradients (CG), heat with Crank–Nicolson stepping whose linear solve is CG (CN–CG), Burgers with Newton–BiCGStab, and Navier–Stokes with the Fourier pseudo-spectral solver that generated its data (Crank–Nicolson diffusion, Adams–Bashforth advection; CNAB2). Each runs at its fastest tested setting at least as accurate as the accurate NM-ROM setting.

5 Numerical Experiments and Results

We ask three questions, on linear (Poisson, heat) and nonlinear (Burgers, Navier–Stokes) problems: how the NM-ROM compares with full-order solvers (§5.1, Table 1), with neural operators and other ROMs (§5.2, Table 2), and which setting to choose (§5.3).

5.1 Accuracy and Speed against Full-Order Solvers

Each row of Table 1 shows two settings of one trained model, an accurate one and a fast one. The fast setting beats the full-order solver on every problem and mesh; the accurate one beats it on every linear problem and on the finer meshes of the nonlinear ones.

Linear problems. On Poisson, the L-shape and heat both settings of a row are solves in the span of the ordered bank. The head is less accurate there (4.48 % against 0.83 % on Poisson at 4096^2, Table D.1): on a linear problem the best solve in the bank is the linear one, and it is also the cheapest. Because the reduced query does not visit the grid until it returns the field, the speedup grows with the mesh, to 108× over CG on Poisson at 4096^2 and 216× over CN–CG on heat at 2048^2. The accurate error is the same at every mesh because it is the bank's projection floor, the best fit of the same-grid solution in the span of the frozen bank; the reduced solve reaches that floor (0.83 % on Poisson), and the floor does not change once the solution is resolved.

Burgers. At coarse meshes the reduced solve is slower than Newton–BiCGStab, but its cost barely changes with the mesh. On the 64 held-out test cases the accurate setting (the span of 384) reaches 0.87–1.35 % (median 0.03 %) at 0.38–10.7× the speed of Newton–BiCGStab, and the fast setting (the span of 96 up to 1024^2 and of 128 beyond) 3.06–4.25 % at 1.22–27.5×; the fast setting is faster than the FOM at every mesh and the accurate one from 1024^2.

In three dimensions one frozen span model, trained on 32^3–128^3 fields and used unchanged at 256^3, reaches 2.20 % at 2.19× the speed of Newton–BiCGStab at 256^3, with settings chosen on 64 validation cases; advection uses a precomputed quadratic tensor, and the nonlinear head did not generalise on this family and is not used.

Navier–Stokes. Here the head earns its place. On the held-out test cases the head with $k = 8$ reaches 0.21 % at every mesh, at 0.93×, 2.92× and 8.01× the speed of CNAB2 at 32^3, 64^3 and 96^3. The span of the same bank is 33× less accurate at the same number of unknowns, and even the full rank-64 span reaches only 1.12 %: the moving vortices lie on a curved manifold that no small span contains. The error stays flat from 32^3 to 96^3 while the speedup rises.

Where the speedup comes from. On the linear problems the query does not iterate: it is one least-squares solve with a matrix factorised offline, and its time is mostly writing the $O(NR')$ output field (41.8 of 42.4 ms on the Poisson development cases at 4096^2, where CG needs about 5300 iterations over all nodes). On the nonlinear problems the iteration count does not grow with the mesh: about one Levenberg–Marquardt iteration per time step on Burgers and three Gauss–Newton sweeps on Navier–Stokes, each costing (11) (Table C.6). The full-order solvers sweep all N nodes in every iteration and, for CG, need more iterations as the mesh is refined (Saad, 2003). At the settings of Table 1, time grows as N^p with $p \approx 0.2$ for our model on Burgers and 0.3 on Navier–Stokes, against about 0.8 and 0.9 for the full-order solvers.

5.2 Comparison with Neural Operators and Other ROMs

We next compare with the two alternatives a practitioner would consider instead: neural operators and other reduced-order models. We train an FNO (Li et al., 2021), a U-Net (Ronneberger et al., 2015), a DeepONet (Lu et al., 2021) and a Transolver (Wu et al., 2024), widely used representatives of spectral, convolutional, branch–trunk and attention-based operators, on the evaluation mesh, on our model's training split (on Burgers, 128 trajectories of the same family) with a 3000 s wall budget per network, and time them with the NM-ROM and the full-order solver in one allocation per problem (Table 2).

Against neural operators, the NM-ROM wins as the mesh grows. On small meshes the operators are competitive, since a forward pass is cheap: at 256^2 on Burgers all four are faster than the NM-ROM, and at 32^3 on Navier–Stokes the FNO is faster than our accurate setting and more accurate than our fast one. As the mesh grows, the NM-ROM's advantage grows with it: its query does not visit the grid until it returns the field, while every operator reads and writes the full grid, and the FNO, U-Net and Transolver errors on Burgers grow 2–5× from 256^2 to 2048^2. From 1024^2 on, and on Navier–Stokes from 64^3, one NM-ROM setting is both more accurate and faster than all four

Table 2: Best NM-ROM against FNO, U-Net, Transolver and DeepONet at every measured (PDE, N) cell. Both NM-ROM columns come from the same trained model. Error and speedup as in Table 1, one full-order setting per cell; *italic* marks speedups below 1×. **Bold** marks an NM-ROM setting that is both more accurate and faster than every operator in the cell. Each operator family shows the size chosen on its validation cases; most operator runs stopped at the 3000 s wall budget (Table B.3). Timed separately from Table 1 (Appendix B).

		NM-ROM accurate		NM-ROM second setting		FNO		U-Net		Transolver		DeepONet	
PDE	N	Err. (%)	Speedup	Err. (%)	Speedup	Err. (%)	Speedup	Err. (%)	Speedup	Err. (%)	Speedup	Err. (%)	Speedup
Burgers 2D	256^2	—	—	4.10	*0.58*×	8.76	2.53×	5.98	1.83×	6.83	1.64×	64.2	4.79×
	1024^2	1.25	*0.24*×	**8.42**	**2.15**×	14.8	*0.74*×	11.5	1.38×	14.9	*0.55*×	68.2	1.44×
	2048^2	1.32	1.05×	**9.04**	**7.37**×	18.0	*0.31*×	22.6	1.42×	33.8	1.43×	67.1	1.96×
Heat 2D	1024^2	**0.13**	**87.4**×	—	—	3.59	1.07×	1.56	1.93×	2.57	6.22×	4.60	2.09×
	2048^2	**0.13**	**289**×	—	—	3.98	*0.93*×	2.76	2.76×	3.66	40.5×	10.9	5.11×
Navier–Stokes 3D	32^3	0.21	*0.94*×	3.24	1.68×	0.53	1.69×	0.68	*0.37*×	0.78	*0.34*×	51.9	1.52×
	64^3	**0.21**	**3.00**×	3.24	5.29×	1.76	2.07×	1.58	*0.75*×	1.80	*0.71*×	47.6	1.39×
	96^3	**0.21**	**8.01**×	3.24	14.4×	1.51	3.23×	2.29	1.44×	1.16	6.65×	51.4	2.26×

operators (Table 2, bold): on Burgers 8.42–9.04 % against 11.5–68.2 % for the operators, on heat 0.13 % against 1.56–10.9 %, and on Navier–Stokes 0.21 % against 1.16–51.4 %.

At the same number of unknowns, the learned manifold is far more accurate than prior ROMs. With 16 unknowns, a linear basis and a quadratic manifold cannot follow the moving fronts of Burgers, and both run slower than the full-order solver; our head follows them at a cost that does not grow with the mesh. On the 64 test cases at 1024^2 and 2048^2, the head with $k = 16$ reaches 8.42–9.04 % at 2.17–4.79× the speed of Newton–BiCGStab, against 34.1–41.7 % for POD-LSPG and a quadratic manifold (Geelen et al., 2022) (Table B.1). POD-LSPG approaches 1 % only with 256 or more unknowns, two orders of magnitude slower, and its basis is rebuilt from 128 new full-order trajectories at each mesh. A reproduced shallow masked-autoencoder NM-ROM (Y. Kim et al., 2022) did not reach a usable accuracy in our setting (145–175 %; Table B.1).

5.3 Which Knob to Turn

A user of the trained model can turn two knobs without retraining: the bank width R' and the choice between head and span (full ladders in Table D.1). The width changes what the model can represent, so it is the accuracy knob. On Poisson at 4096^2, narrowing the bank from $R' = 512$ to 32 raises the error 18.6× and moves the same model from 108× to 1136× the speed of CG on the test cases. On heat at 2048^2 the span runs from 0.13 % at 216× ($R' = 128$) to 1.08 % at 512× ($R' = 32$), and on Burgers at 4096^2 from 1.35 % at 10.7× ($R' = 384$) to 8.24 % at 44.5× ($R' = 64$). The choice between head and span depends on the problem: on the linear problems the span is both more accurate and cheaper (§5.1), while on Navier–Stokes the head is the accurate end, 33× more accurate than the span at the same number of unknowns for about twice the cost.

Limitations. (i) Except for Navier–Stokes, speedups are against named iterative full-order solvers at the stated tolerances. Fast direct and spectral solvers exist for the square and the cube and would be faster than CG there. (ii) The Navier–Stokes bank is a POD basis built at each mesh and evaluated in a moving frame, with a head trained at each mesh, and the family has moderate Reynolds numbers; we make no claim about turbulent flow. (iii) On the L-shape the available tests limit the span to width 128. (iv) Quadrature validation covers finitely many reached states and is not a global certificate, and we changed the check once after seeing results (§3.3). Reduced solves can converge to a wrong branch, and lower same-grid error does not remove discretisation error. (v) The evaluation cohorts are small (6–64 cases). Timings report medians without dispersion.

6 Conclusion and Future Work

We built an NM-ROM whose speedup grows with the mesh: it is up to 27.5× faster than the full-order solver on nonlinear problems and over 100× faster on linear ones. One trained model also offers several accuracy settings at query time, through the width of its ordered bank and the choice between head and span, whereas a trained neural operator gives one fixed point at a given mesh. Next are higher-Reynolds-number flows and a nonlinear head for three-dimensional Burgers.

ETHICS STATEMENT

This work develops numerical methods for solving partial differential equations. All data are synthetic, generated by numerical solvers from sampled parameters; no human subjects, personal data or released datasets are involved. We are not aware of harmful uses beyond those of general-purpose simulation software.

REPRODUCIBILITY STATEMENT

Every table and prose number is taken from versioned run records identifying source, checkpoint, settings, cohort, allocation and audit (Appendix C); failed settings are kept.

AI USE STATEMENT

AI coding agents (Claude, Codex) implemented the methods, baselines and pipelines, proposed and refined experiment designs, analysed and cross-checked results, and drafted and edited text. Every reported number comes from a recorded solver run (Appendix C). The authors reviewed this work and take responsibility for all content.

REFERENCES

K. Azizzadenesheli, N. Kovachki, Z. Li, M. Liu-Schiaffini, J. Kossaifi, and A. Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024.

J. Barnett and C. Farhat. Quadratic approximation manifold for mitigating the Kolmogorov barrier in nonlinear projection-based model order reduction. *Journal of Computational Physics*, 464:111348, 2022.

J. Barnett, C. Farhat, and Y. Maday. Neural-network-augmented projection-based model order reduction for mitigating the Kolmogorov barrier to reducibility. *Journal of Computational Physics*, 492:112420, 2023.

P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

J. Berman and B. Peherstorfer. CoLoRA: Continuous low-rank adaptation for reduced implicit neural modeling of parameterized partial differential equations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pages 3565–3583, 2024.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han. Once-for-All: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020.

K. Carlberg. Adaptive h-refinement for reduced-order models. *International Journal for Numerical Methods in Engineering*, 102(5):1192–1210, 2015.

K. Carlberg, C. Bou-Mosleh, and C. Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.

P. Y. Chen, J. Xiang, D. H. Cho, Y. Chang, G. A. Pershing, H. T. Maia, M. M. Chiaramonte, K. T. Carlberg, and E. Grinspun. CROM: Continuous reduced-order modeling of PDEs using implicit neural representations. In *International Conference on Learning Representations*, 2023.

A. Cohen and R. DeVore. Approximation of high-dimensional parametric PDEs. *Acta Numerica*, 24:1–159, 2015.

Devvrit, S. Kudugunta, A. Kusupati, T. Dettmers, K. Chen, I. Dhillon, Y. Tsvetkov, H. Hajishirzi, S. Kakade, A. Farhadi, and P. Jain. MatFormer: Nested transformer for elastic inference. In *Advances in Neural Information Processing Systems*, volume 37, pages 140535–140564, 2024.

A. N. Diaz, Y. Choi, and M. Heinkenschloss. A fast and accurate domain decomposition nonlinear manifold reduced order model. *Computer Methods in Applied Mechanics and Engineering*, 425:116943, 2024.

W. D. Fries, X. He, and Y. Choi. LaSDI: Parametric latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 399:115436, 2022.

R. Geelen, S. Wright, and K. Willcox. Operator inference for non-intrusive model reduction with quadratic manifolds. *Computer Methods in Applied Mechanics and Engineering*, 403:115717, 2022.

G. H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis*, 10(2):413–432, 1973.

C. Greif and K. Urban. Decay of the Kolmogorov N-width for wave problems. *Applied Mathematics Letters*, 96:216–222, 2019.

Z. Hao, C. Su, S. Liu, J. Berner, C. Ying, H. Su, A. Anandkumar, J. Song, and J. Zhu. DPOT: Auto-regressive denoising operator transformer for large-scale PDE pre-training. In *International Conference on Machine Learning*, pages 17616–17635, 2024.

X. He, Y. Choi, W. D. Fries, J. L. Belof, and J.-S. Chen. gLaSDI: Parametric physics-informed greedy latent space dynamics identification. *Journal of Computational Physics*, 489:112267, 2023.

M. Herde, B. Raonić, T. Rohner, R. Käppeli, R. Molinaro, E. de Bézenac, and S. Mishra. Poseidon: Efficient foundation models for PDEs. In *Advances in Neural Information Processing Systems*, volume 37, pages 72525–72624, 2024.

J. A. Hernández, M. A. Caicedo, and A. Ferrer. Dimensional hyper-reduction of nonlinear finite element models via empirical cubature. *Computer Methods in Applied Mechanics and Engineering*, 313:687–722, 2017.

J. A. Hernández, J. R. Bravo, and S. Ares de Parga. CECM: A continuous empirical cubature method with application to the dimensional hyperreduction of parameterized finite element models. *Computer Methods in Applied Mechanics and Engineering*, 418:116552, 2024.

M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

J. S. Hesthaven, B. Peherstorfer, and B. Unger. Nonlinear model reduction for transport-dominated problems. *arXiv preprint arXiv:2602.01397*, 2026.

B. Holzschuh, Q. Liu, G. Kohl, and N. Thuerey. PDE-Transformer: Efficient and versatile transformers for physics simulations. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

M. V. de Hoop, D. Z. Huang, E. Qian, and A. M. Stuart. The cost-accuracy trade-off in operator learning with neural networks. *Journal of Machine Learning*, 1(3):299–341, 2022.

Y. Kim, Y. Choi, D. Widemann, and T. Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.

M. Kim, T. Wen, K. Lee, and Y. Choi. Physics-informed reduced order model with conditional neural fields. *arXiv preprint arXiv:2412.05233*, 2024.

N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: learning maps between function spaces. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

A. Kusupati, G. Bhatt, A. Rege, M. Wallingford, A. Sinha, V. Ramanujan, W. Howard-Snyder, K. Chen, S. Kakade, P. Jain, and A. Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 30233–30249, 2022.

C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

K. Lee and K. T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

H. Luo, H. Wu, H. Zhou, L. Xing, Y. Di, J. Wang, and M. Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *International Conference on Machine Learning*, pages 41432–41449, 2025.

L. Magargal, P. Khodabakhshi, S. Rodriguez, J. Jaworski, and J. Michopoulos. Projection-based model-order reduction via graph autoencoders suited for unstructured meshes. *Data-Centric Engineering*, 6:e52, 2025.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

N. McGreivy and A. Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6(10):1256–1269, 2024.

B. Peherstorfer. Breaking the Kolmogorov barrier with nonlinear model reduction. *Notices of the American Mathematical Society*, 69(5):725–733, 2022.

B. Peherstorfer and K. Willcox. Online adaptive model reduction for nonlinear systems via low-rank updates. *SIAM Journal on Scientific Computing*, 37(4):A2123–A2150, 2015.

F. Romor, G. Stabile, and G. Rozza. Non-linear manifold reduced-order models with convolutional autoencoders and reduced over-collocation method. *Journal of Scientific Computing*, 94(3):74, 2023.

F. Romor, G. Stabile, and G. Rozza. Explicable hyper-reduced order models on nonlinearly approximated solution manifolds of compressible and incompressible Navier–Stokes equations. *Journal of Computational Physics*, 524:113729, 2025.

O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.

L. Serrano, L. Le Boudec, A. Kassaï Koupaï, T. X. Wang, Y. Yin, J.-N. Vittaut, and P. Gallinari. Operator learning with neural fields: Tackling PDEs on general geometries. In *Advances in Neural Information Processing Systems*, volume 36, pages 70581–70611, 2023.

L. Sirovich. Turbulence and the dynamics of coherent structures, Parts I–III. *Quarterly of Applied Mathematics*, 45(3):561–590, 1987.

R. Ştefănescu and A. Sandu. Efficient approximation of sparse Jacobians for time-implicit reduced order models. *arXiv preprint arXiv:1402.2018*, 2014.

M. Takamoto, T. Praditia, R. Leiteritz, D. MacKinlay, F. Alesiani, D. Pflüger, and M. Niepert. PDEBench: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 35, 2022.

M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33, 2020.

P. Weder, P. Schwerdtner, and B. Peherstorfer. Nonlinear model reduction with Neural Galerkin schemes on quadratic manifolds. *arXiv preprint arXiv:2412.17695*, 2024.

H. Wu, H. Luo, H. Wang, J. Wang, and M. Long. Transolver: A fast transformer solver for PDEs on general geometries. In *International Conference on Machine Learning*, 2024.

M. Yano and A. T. Patera. An LP empirical quadrature procedure for reduced basis treatment of parametrized nonlinear PDEs. *Computer Methods in Applied Mechanics and Engineering*, 344:1104–1123, 2019.

J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2019.

A METHOD DETAILS

This appendix gives the details of §3 that the main text leaves out: the ordering of the bank, the row scaling of each PDE, the Burgers stencil, the Navier–Stokes family, the quadrature fit and the solver constants. On the square the tests are eigenvectors of the Laplacian, $PK = \Lambda P$ with Λ diagonal, and every PDE scales the tested residual PR by the inverse of a diagonal matrix built from Λ. On the unit cube the tests are the M lowest discrete sine modes $\sqrt{8}\prod_i \sin(\pi k_i x_i)$, ordered by $|k|^2$ ($M = 512$ for Poisson, 1024 for heat and about $4R'$ for Burgers, completed to a full shell of equal $|k|^2$), and the boundary factor is $\mu(x) = 64\prod_i x_i(1 - x_i)$. On the L-shaped domain the tests are the $M = 257$ lowest eigenvectors of its five-point matrix, computed once per mesh by shift-invert Lanczos, and the boundary factor is $\mu(x) = 4\min\big(x_1,\, x_2,\, r,\, t,\, \|x - (\frac{1}{2}, \frac{1}{2})\|\big)$, with $r = 1 - x_1$ for $x_2 \le \frac{1}{2}$ and $\frac{1}{2} - x_1$ otherwise, and $t = 1 - x_2$ for $x_1 \le \frac{1}{2}$ and $\frac{1}{2} - x_2$ otherwise.

A.1 ORDERING THE BANK

The ordering is a principal component analysis of the training fields, done inside the space the bank spans. First we make the bank orthonormal with a thin QR factorisation, $G = Q_G R_G$, so that a field Ga has coordinates $R_G a$ in the orthonormal basis Q_G. For every training field u_i we take these coordinates and divide by $\|u_i\|$, so that every field counts equally, and stack the results as the rows of a matrix $\mathcal{A}$. Its singular value decomposition $\mathcal{A} = USV^\top$ gives the directions the training fields use, in the columns of V, sorted by the singular values in S. We set $T = R_G^{-1}V$, so that $\widehat{G} = GT = Q_G V$: its columns are orthonormal and sorted by importance, and by the Eckart–Young theorem its first R' columns are the best R'-dimensional subspace of the bank for the scaled training fields. Because T is invertible, the full-width bank gives the same trial states as before (to 10^{-10} in our runs), and the head is kept unchanged. The step uses training data only and has no trainable weights.

A.2 POISSON

The tested residual is scaled by Λ^{-1}. Because the retained tests are orthonormal and $Ku^\star = F$, the scaled residual $\Lambda^{-1}P(K\tilde{u} - F) = P(\tilde{u} - u^\star)$ is exactly the tested error, so the objective is the squared discrete L^2 error of the reduced state in the retained modes. In the span, the thin QR factorisation of $P\widehat{G}_{R'}$ is computed once per mesh and width, and a query is one matrix–vector product and one triangular solve. At $R' = R$ the span solve reaches the bank floor in the tested norm.

A.3 HEAT

The semi-discretisation $du/dt = -\kappa K u$, with diffusivity κ, is advanced by Crank–Nicolson with the residual

$$R(u_{n+1}) = \big(I + \tfrac{\Delta t\kappa}{2}K\big)u_{n+1} - \big(I - \tfrac{\Delta t\kappa}{2}K\big)u_n. \tag{14}$$

The tested Crank–Nicolson residual (14) is scaled by $(I + \frac{\Delta t\kappa}{2}\Lambda)^{-1}$, which turns each step into $P\tilde{u}_{n+1} = D\,P\tilde{u}_n$ with $D = \operatorname{diag}\big((1 - \frac{\Delta t\kappa}{2}\lambda_i)/(1 + \frac{\Delta t\kappa}{2}\lambda_i)\big)$: each tested mode is multiplied by its Crank–Nicolson factor. In the span the step is the linear recurrence

$$c_{n+1} = B_{R'}^{+} D\, B_{R'}\, c_n, \tag{15}$$

with $B_{R'} = P\widehat{G}_{R'}$ and c_0 fitted to the initial field. Because the step matrix $B_{R'}^{+} D\, B_{R'}$ is fixed, its powers at the output times are formed once offline, so a query fits c_0, applies one small matrix per output time and rebuilds the fields. This is exactly the per-step least-squares solution of the tested Crank–Nicolson residual at the full-order time step $\Delta t = 0.025$, with no further time-discretisation error, and every heat result in the paper is computed this way.

A.4 BURGERS

For $u_t + u(u_x + u_y) = \nu\Delta u$, with viscosity ν, backward Euler gives the residual

$$R(u_{n+1}) = u_{n+1} - u_n + \Delta t\big(A(u_{n+1}) + \nu K u_{n+1}\big), \tag{16}$$

where $A(u)$ is the advection term. The full-order problem uses the sign-upwind stencil

$$A(u)_{ij} \;=\; u_{ij}\left(\delta_x u + \delta_y u\right)_{ij}, \qquad (\delta_x u)_{ij} = \frac{1}{\Delta x}\begin{cases} u_{ij} - u_{i-1,j}, & u_{ij} > 0, \\ u_{i+1,j} - u_{ij}, & u_{ij} \le 0, \end{cases} \tag{17}$$

with δ_y analogous, zero ghost values on the walls and backward Euler in time. The tested residual (16) is scaled by $(I + \Delta t\,\nu\Lambda)^{-1}$, the modal form of the Helmholtz preconditioner the full-order Newton solver uses. Each step is warm-started from the previous coefficients, and a linear or quadratic extrapolation is used when its residual is smaller. At every mesh the step assembles the Jacobian analytically, clips the step, solves the damped normal equation by Cholesky factorisation and carries the damping between steps.

A.5 Navier–Stokes

For $u_t + (u\cdot\nabla)u = -\nabla p + \nu\Delta u$, $\nabla\cdot u = 0$, the implicit midpoint rule gives the residual

$$R(u_{n+1}) = u_{n+1} - u_n + \Delta t\big((\bar u\cdot\nabla)\bar u + \nabla\bar p - \nu\Delta\bar u\big), \qquad \bar u = \tfrac{1}{2}(u_n + u_{n+1}). \tag{18}$$

The unknown is the three-component, zero-mean velocity on the periodic unit cube; the sampled family is in Table C.4. The data and the full-order comparator come from CNAB2 with two-thirds dealiasing. The bank is the rank-64 POD basis of 768 training states (six frames of each of 128 trajectories) generated on the query mesh, each centred on its own energy centroid. It is ordered once by the eigenvectors of the second moment of the training coefficients, which gives the nested widths $R' \in \{64, 48, 32, 16, 8\}$; the centroid moves with the vortices, and a query solves for its three-component shift at every step together with the latent coordinates. All linear reduced operators are computed offline, and the advection term is evaluated with a precomputed quadratic tensor (§3.3). The head is trained as an auto-decoder on the centred bank coefficients of training states. Bank and head are built once per mesh (32^3, 64^3, 96^3) and frozen. The tests are the $M = 292$ lowest divergence-free Fourier modes. A query centres the initial field, projects it on the bank and, for the head, starts from a stored training code. It then takes 10 implicit-midpoint steps of $\Delta t = 0.02$, each solved by three damped Gauss–Newton sweeps with an analytic Jacobian, starting from a linear extrapolation of the last two steps. With the centring held fixed the same bank misses every case by far more than 5 %, so the centring is necessary.

A.6 Empirical quadrature

On one state, a rule's error is

$$\rho(u) = \frac{\|\sum_{i\in\mathcal{S}} w_i P_{:,i} A_i(u) - PA(u)\|_2}{\|PA(u)\|_2}, \qquad \rho_{\max} \le 0.116, \tag{19}$$

over reachable states. The bar was fixed before any check was run. A rule is confirmed when $\rho_{\max}$ stays below it first on one set of trajectories disjoint from the fit and the evaluation cases, then on a further set checked last. Backward Euler never evaluates advection at the initial state. The first registered version of the check included that state; we removed it after observing that it was the only state at which the widest span settings failed, and this is the only change we made to the check after seeing results. The fit (10) normalises each row, grows the support greedily by the Lawson–Hanson criterion with an exact non-negative refit after each addition, and stops at N_{eq} nodes. The fit states are states the dense solver reaches on fit trajectories; the test states come from trajectories disjoint from the fit and the evaluation cases. A deployed rule is its node indices, weights and the cached $N_{\mathrm{eq}} \times 5 \times R$ bank block. The Burgers rows of Table 1 use rules fitted this way.

A.7 Solver constants

With r_w the scaled tested residual and J its Jacobian, the stopping test is scale free, $\|J^\top r_w\|_2 \le \eta_{\mathrm{tol}}\|J\|_F\,\|r_w\|_2$, so one tolerance serves every mesh and width. The Poisson, L-shape and heat settings of Table 1 are span solves with no stopping test. Two-dimensional Burgers uses $\eta_{\mathrm{tol}} = 10^{-2}$ at 256^2–1024^2 and 10^{-3} at 2048^2 and 4096^2, with at most 600 attempts per step. Three-dimensional Burgers solves the first three steps to $\eta_{\mathrm{tol}} = 10^{-3}$ (at most 50 attempts) and then takes one sweep per step. Navier–Stokes takes a fixed three Gauss–Newton sweeps per step with fixed damping $\lambda = 10^{-6}$.

Damping starts at $\lambda_0 = 10^{-6}$ (Poisson, Burgers in two and three dimensions) or 10^{-4} (heat), is carried between time steps, is divided by 3 on acceptance (down to 10^{-12}) and multiplied by 10 on rejection (up to 10^{14}). The *bank floor* is the projection error onto the span of the bank, which no solve can beat.

B NEURAL OPERATORS

We trained an FNO (Li et al., 2021), a U-Net (Ronneberger et al., 2015), a DeepONet (Lu et al., 2021) and a Transolver (Wu et al., 2024) on Burgers 2D, heat 2D and Navier–Stokes 3D, each at the mesh it is evaluated on, with AdamW, one fixed seed per network (Table B.3) and a 3000 s wall budget per network, and scored them by the same metric as our model. They produce the requested output times in one pass, so their output times are fixed at training.

Cells of Table 2. Error is over evolved times for Burgers and all times for heat. Cells: Burgers 256^2, 1024^2 and 2048^2 (A100, A100 and H100 GPUs), the 64 test cases of Table 1, FOM Newton–BiCGStab tol 3×10^{-3}, Δt=0.005 (0.11, 0.13 and 0.14 %); heat 1024^2 and 2048^2 (A100 and H200 GPUs), the 16 test cases of Table 1, FOM CN–CG Δt=0.025, rtol 10^{-4} (0.045 %); Navier–Stokes 32^3 and 64^3, the 32 test cases of Table 1, FOM CNAB2 with 40 and 50 steps (0.19 and 0.12 %); Navier–Stokes 96^3, 32 test cases, FOM CNAB2 with 70 steps (0.056 %). In these jobs the Burgers accurate setting (span, $R' = 384$) ran on the general solver path, slower than the path of Table 1, and was not timed at 256^2; the second NM-ROM setting is the head with $k = 16$ on Burgers and the span $R' = 16$ on Navier–Stokes, and was not timed for heat. These cells were timed in separate runs from Table 1, so the same setting can show a different speedup in the two tables.

Table B.1: Best NM-ROM against other nonlinear-manifold and projection ROMs on Burgers 2D at matched latent dimension ($k = 16$ for every baseline and for our head). Error is the worst evolved same-grid relative L^2 error (%). Speedup as in Table 2: one full-order setting per cell, the fastest at least as accurate as the NM-ROM accurate setting, same allocation; *italic* below 1×; **bold** marks an NM-ROM setting more accurate and faster than every baseline. At 256^2 and 512^2 the cohort is 32 validation cases (never used for any choice) and every method, ours included, uses a dense residual, so no times are given; at 1024^2 and 2048^2 it is the 64 held-out test cases of Table 1. The shallow masked-autoencoder NM-LSPG of Y. Kim et al. (2022) was checked against its published result before use (Table C.7); its hyper-reduction was not reproduced and it was not run from 1024^2. The quadratic manifold (Geelen et al., 2022) was run at 1024^2 and 2048^2 only. Every latent size and the memory use are in Tables C.7 and B.2. This table was timed in a separate run from Table 2 (H200 at 2048^2, H100 there), so the same setting can show a different speedup.

		NM-ROM accurate		NM-ROM head, k=16		Kim et al. NM-LSPG		POD-LSPG		Quadratic manifold	
PDE	N	Err. (%)	Speedup	Err. (%)	Speedup	Err. (%)	Speedup	Err. (%)	Speedup	Err. (%)	Speedup
Burgers 2D	256^2	—	—	6.79	—	145.40	—	41.53	—	—	—
	512^2	—	—	7.72	—	174.99	—	42.01	—	—	—
	1024^2	1.25	*0.24*×	**8.42**	**2.17**×	—	—	41.6	*0.41*×	34.1	*0.101*×
	2048^2	1.32	*0.68*×	**9.04**	**4.79**×	—	—	41.7	*0.43*×	34.2	*0.096*×

Other reduced-order models at 1024^2. Table B.2 gives POD-LSPG and a quadratic manifold (Geelen et al., 2022) at several sizes on the six development cases, in a separate job from Table 2. Neither is both more accurate and faster than our setting. POD-LSPG needs 256 or more unknowns to approach 1 %; the quadratic manifold stays above 7 % up to 64.

Navier–Stokes cells. Both sides draw from the same 512 training trajectories. The operators train on 448 and choose their size on the other 64, the same split the head used. At 32^3 CNAB2 with 40 steps reaches 0.185 % on the test cases, below the accurate setting's 0.206 %, so the rule selects 40 steps.

How much data each side was given. On heat and Navier–Stokes both sides are trained on the same trajectories. On Burgers the operators use 128 trajectories, while our bank is fitted on 576 and the head on 4608 of the same family (Table C.1). Both sides draw initial conditions from the same family and are scored at the same output times. At 256^2 each operator target is a 4096-interval solution restricted to the 256-interval grid. At 1024^2 and 2048^2 the trajectories are recomputed on

Table B.2: Burgers 2D at 1024^2, one allocation (A100 80 GB, six development cases): worst and median evolved same-grid error, median GPU query time, and speedup over one full-order setting chosen on these development cases (Newton–BiCGStab, tol 3×10^{-3}, Δt=0.005, 70.5 ms, the fastest setting at least as accurate as the NM-ROM accurate setting); below $1\times$ the row is slower. FOM rows are Newton–BiCGStab settings; 1023^2 is the number of interior nodes.

Method	Unknowns	Worst (%)	Median (%)	GPU ms	Speedup
NM-ROM, span (R'=384)	384	0.21	0.065	318	0.22×
NM-ROM, head (k=16)	16	2.29	1.04	34.9	2.02×
POD-LSPG	16	29.3	22.1	178	0.40×
POD-LSPG	64	7.41	3.66	911	0.077×
POD-LSPG	256	1.10	0.42	12147	0.006×
POD-LSPG	512	0.44	0.053	47503	0.001×
Quadratic manifold	16	22.6	15.0	733	0.096×
Quadratic manifold	32	13.0	6.68	2547	0.028×
Quadratic manifold	64	7.26	3.58	8257	0.009×
FOM, tol 10^{-2}, Δt=0.01	1023^2	1.66	1.20	37.0	—
FOM, tol 3×10^{-3}, Δt=0.005	1023^2	0.048	0.038	70.5	—

the evaluation mesh with a tight Newton tolerance, so there the operators are trained on exactly the fields they are scored against, while our model, trained at 256^2, is used unchanged. An operator sees one input–output pair per trajectory, while the head is fitted per state.

C Experimental configuration and reproducibility

Tables C.3 and C.4 identify the two-dimensional studies and Table C.8 the three-dimensional ones. Within each study, training precedes evaluation, and the selected bank, its ordering and the head remain frozen. Deployment changes reduced coordinates, not network weights. Table C.1 gives the training configuration of every frozen model and Table C.2 its offline cost.

Full-order and solver details. The Burgers FOM is Newton's method with BiCGStab inner solves (at most 200 iterations, Jacobian–vector products by forward-mode differentiation) preconditioned by the exact inverse of the implicit Helmholtz operator $(I + \Delta t\,\nu\,\lambda)^{-1}$ (advection dropped); Newton stops when the residual norm falls below the nonlinear tolerance times the previous state's norm, or after 20 iterations, and always takes the full step (no line search or damping). Each named setting fixes Δt, the nonlinear and the linear tolerance.

C.1 Headline settings, times and timing protocol

The settings behind Table 1, all from one frozen model per problem with no retraining: Poisson (square and cube) and heat use the bank span at two widths ($R' = 512/128$ square, $128/32$ cube, $128/32$ heat 2D, $320/128$ heat 3D at 32^3 and $320/64$ above, Crank–Nicolson stepping); Navier–Stokes uses the head $k = 8$ and the span $R' = 16$ (the cheapest span setting below 5 %); the L-shape uses the span $R' = 128/64$ (the widest span its tests support is 128); Burgers uses the span $R' = 384$ at every mesh and $R' = 96$ (256^2–1024^2) or $R' = 128$ (2048^2, 4096^2). Burgers 3D uses one coordinate bank ($R = 512$, trained once on 32^3, 64^3 and 128^3 fields) with the span $R' = 512$ and $R' = 192$ ($R' = 256$ at 256^3), a fixed-sweep Levenberg–Marquardt solve and a precomputed quadratic tensor for the advection term (about $4R'$ tests); its settings were chosen by the same rule on 64 validation cases and its rows are 32 test cases, all on H200 GPUs. All Burgers 2D rows were timed on A100 80 GB GPUs. The three-dimensional Poisson rows use 64 test cases. The FOM setting can differ between meshes of a series.

Table C.5 gives the settings and measured times behind every row of Table 1. Every cost and error come from the same invocation. Timings use GPU warm-up and synchronisation, double precision and the highest matrix-multiplication precision; all repetitions, including outliers, enter the median, and no time is borrowed from another allocation. Requested full fields and initialisation are charged; host transfer is not.

Table B.3: Training of the neural operators of Table 2, one row per trained network shown there: parameters, epochs completed within the 3000 s wall budget, and why training stopped. Every network used AdamW, weight decay 10^{-4}, batch 8 (by gradient accumulation where memory required it), seed 20260914 and the checkpoint of its best validation epoch. Training sets: the training trajectories and split of our model for each problem (heat 512 with 16 validation, Navier–Stokes 448 with 64 validation). The Burgers 2048^2 FNO was stopped by host memory after 19 epochs.

Cell	Operator	Parameters	Epochs	Stopped by
Burgers 2D, 256^2	FNO	17,877,317	692	wall budget
	U-Net	17,462,021	908	wall budget
	Transolver	3,108,240	1628	wall budget
	DeepONet	2,569,349	537	early stopping
Burgers 2D, 1024^2	FNO	17,877,317	37	wall budget
	U-Net	7,763,461	223	wall budget
	Transolver	3,108,240	131	wall budget
	DeepONet	2,569,349	308	wall budget
Burgers 2D, 2048^2	FNO	17,877,317	19	host memory
	U-Net	7,763,461	62	wall budget
	Transolver	3,108,240	24	wall budget
	DeepONet	2,569,349	69	wall budget
Heat 2D, 1024^2	FNO	17,877,189	7	wall budget
	U-Net	7,763,173	71	wall budget
	Transolver	12,746,048	228	wall budget
	DeepONet	2,568,917	98	wall budget
Heat 2D, 2048^2	FNO	17,877,189	6	wall budget
	U-Net	7,763,173	23	wall budget
	Transolver	14,126,144	394	wall budget
	DeepONet	2,568,917	31	wall budget
Navier–Stokes 3D, 32^3	FNO	18,888,495	624	wall budget
	U-Net	22,581,327	458	wall budget
	Transolver	6,003,816	269	wall budget
	DeepONet	5,741,519	298	early stopping
Navier–Stokes 3D, 64^3	FNO	18,888,495	88	wall budget
	U-Net	22,581,327	80	wall budget
	Transolver	17,996,928	110	wall budget
	DeepONet	5,741,519	232	wall budget
Navier–Stokes 3D, 96^3	FNO	18,888,495	110	wall budget
	U-Net	22,581,327	62	wall budget
	Transolver	6,619,288	510	wall budget
	DeepONet	5,741,519	70	wall budget

Table C.1: Training configurations of the frozen checkpoints, transcribed from the committed configuration and code files of each study.

Model	Stage	Network	Optimiser, learning rate	Steps	Batch	Training data
Burgers 2D ($k = 16$, $R = 512$)	bank	128 Fourier features (scale 4), width 1024, 2 layers	AdamW, wd 10^{-5}, warm-up + cosine $10^{-3} \to 10^{-5}$	300000	all states $\times$ 4096 points	16384 states of 576 trajectories
	head	width 512, 2 layers	Adam, warm-up + cosine $10^{-3} \to 10^{-5}$	200000	4096 states	131072 states of 4608 trajectories
Poisson 2D ($k = 32$, $R = 512$)	bank	64 Fourier features (scale 4), width R, 2 layers	Adam, four phases at 10^{-3}, $3{\times}10^{-4}$, $3{\times}10^{-4}$, 10^{-4}	100000, 40000, 30000, 30000	64 sources $\times$ 4096 points	2611 fit / 461 validation sources
	head	width 128, 2 layers	Adam, 10^{-3}	150000 (full batch)	all	as bank
L-shape ($k = 16$, $R = 512$)	bank / head	64 Fourier features (scale 4), 2 layers; head width 128, 2 layers	as Poisson $k = 32$	100000, 40000, 30000, 30000; head 150000	64 sources $\times$ 4096 points	2611 fit / 461 validation sources
Poisson 3D ($k = 16$, $R = 128$)	bank / head	64 Fourier features (scale 1.5), widths 256; head width 256 + skip	Adam, cosine (floor 0.03); 10^{-3} bank, $5{\times}10^{-4}$ head	150000 / 100000	64 states $\times$ 2048 points / 128 states	512 training / 16 validation fields
Heat 2D ($k = 8$, $R = 128$)	bank / head	32 Fourier features (scale 1.5), width 256; head width 256	Adam, clip 1, exact coefficient refit every 10000 steps; 10^{-3} bank, $5{\times}10^{-4}$ head	80000 / 80000	64 states $\times$ 2048 points / 128 states	512 training / 16 validation draws at 64^2
Heat 3D ($k = 32$, $R = 320$)	bank / head	128 Fourier features (scale 2), width 512, 3 layers; head width 512, 2 layers + skip	bank: Adam, clip 1, warm-up + cosine $10^{-3} \to 10^{-5}$; head: Adam, cosine from $5{\times}10^{-4}$ (floor 0.03)	40000 (step 32000 kept) / 120000	512 states on the full 32^3 grid / 256 states	2048 training / 256 validation trajectories (6 times each)
Burgers 3D ($R = 512$, span only)	bank	128 Fourier features (scale 2), width 1024, 3 layers	Adam, clip 1, warm-up + cosine $10^{-3} \to 10^{-5}$	12000 (step 7000 kept)	128 states $\times$ 32768 points per mesh	18432 states of 1536 trajectories at each of 32^3, 64^3, 128^3; 96 validation trajectories
Navier–Stokes 3D ($k = 8$, $R = 64$)	bank	POD of centred snapshots, built per mesh (no trained weights)	—	—	—	768 centred states of 128 trajectories
	head	width 256, 2 layers + skip, one per mesh	Adam, clip 1, warm-up + cosine $10^{-3} \to 10^{-5}$	100000 (full batch)	all	9408 training / 1344 validation states of 512 trajectories

Table C.2: Offline cost of each deployed model, in seconds of wall-clock time on one A100 or H200 GPU (the heat orderings on a workstation GPU). Setup is the per-mesh assembly of the tests and reduced operators, over the range of meshes. Dashes mark costs not recorded separately.

Problem	Bank	Head	Ordering	Setup per mesh
Poisson 2D	434	141	4.0	33.6 at 4096^2
Poisson, L-shape	375	98	79	869 at 2048^2 (eigenvectors 830)
Poisson 3D	101	54	4.5	10.7–48.0
Heat 2D	56	1113	14	16–260
Heat 3D	808	94	96	6.5–333
Burgers 2D	10,685	339	34	—
Burgers 3D	2072	—	—	14–279
Navier–Stokes 3D	—	95	—	—

Table C.3: Problem specification: meshes, time stepping and cohorts.

PDE	equation, domain, boundary	meshes	time stepping	cohorts
Burgers 2D	$u_t + u(u_x + u_y) = \nu\Delta u, (0,1)^2, u\|_{\partial\Omega} = 0$	256^2–4096^2	$\Delta t = 0.005$, backward Euler, sign-upwind	6 development cases; 32 validation; 64 test at 256^2–4096^2
Poisson 2D	$-\Delta u = f, (0,1)^2, u\|_{\partial\Omega} = 0$	256^2–4096^2	none (elliptic)	12 development / 32 test sources
Heat 2D, wide bank	$u_t = \kappa\Delta u, (0,1)^2$	1024^2–4096^2	Crank–Nicolson, $\Delta t = 0.025$	16 validation cases (selection); 16 test cases
Poisson, L-shape	$-\Delta u = f, (0,1)^2 \setminus [\frac{1}{2},1)^2$	256^2–2048^2	none	3072 / 256 / 32 / 32 sources (train / selection / development / test)

Table C.4: Sampled problem families.

PDE	equation and boundary	sampled family
Burgers 2D	$u_t + u(u_x + u_y) = \nu\Delta u$ on $(0,1)^2$, $u = 0$ on $\partial\Omega$	$u_0 = a\exp(-\lvert x-c\rvert^2/2w^2)$, $c_i \sim U(0.15, 0.85)$, $w \sim U(0.05, 0.20)$, $a \sim U(0.5, 2)$; $\nu \sim \log U(0.01, 0.1)$; outputs $t \in \{0.05, \ldots, 0.25\}$
Burgers 3D	$u_t + u(u_x + u_y + u_z) = \nu\Delta u$ on $(0,1)^3$, $u = 0$ on $\partial\Omega$	$u_0 \propto a\,\mu(x)\sum_{b\le B}\rho_b\exp(-\lvert x-c_b\rvert^2/2w_b^2)$, $\mu = 64\prod_i x_i(1-x_i)$, $B \in \{1,2,3\}$, $c_{b,i} \sim U(0.2, 0.8)$, $w_b \sim U(0.10, 0.20)$, $\rho_b \sim U(0.5, 1)$, peak normalised to $a \sim U(0.5, 2)$; $\nu \sim \log U(0.01, 0.1)$; backward Euler, sign-upwind; outputs $t \in \{0.05, \ldots, 0.25\}$
Poisson 2D	$-\Delta u = f$ on $(0,1)^2$, $u = 0$ on $\partial\Omega$	$f = a\exp(-\lvert x-c\rvert^2/2w^2)$, $c_i \sim U(0.15, 0.85)$, $w \sim \log U(0.02, 0.1)$, $a \sim U(0.5, 2)$
Poisson, L-shape	same source family on $(0,1)^2 \setminus [\frac{1}{2}, 1)^2$	as above; sources centred in the removed quadrant rejected
Heat 2D	$u_t = \kappa\Delta u$ on $(0,1)^2$, $u = 0$ on $\partial\Omega$, $\kappa = 0.02$ fixed	$u_0 = a\,16x_1(1-x_1)x_2(1-x_2)\exp(-\lvert x-c\rvert^2/2w^2)$, $c_i \sim U(0.35, 0.65)$, $w \sim U(0.10, 0.15)$, $a \sim U(0.8, 1.2)$; outputs $t \in \{0, 0.1, \ldots, 0.5\}$; Crank–Nicolson $\Delta t = 0.025$
Poisson 3D	$-\Delta u = f$ on $(0,1)^3$, $u = 0$ on $\partial\Omega$	$f = a\,64\prod_i x_i(1-x_i)\exp(-\lvert x-c\rvert^2/2w^2)$, $c_i \sim U(0.35, 0.65)$, $w \sim U(0.10, 0.15)$, $a \sim U(0.8, 1.2)$
Heat 3D	$u_t = \kappa\Delta u$ on $(0,1)^3$, $u = 0$ on $\partial\Omega$, $\kappa = 0.02$ fixed	$u_0 = a\prod_i 4x_i(1-x_i)\exp(-\lvert x-c\rvert^2/2w^2)$, $c_i \sim U(0.35, 0.65)$, $w \sim U(0.10, 0.15)$, $a \sim U(0.8, 1.2)$; outputs $t \in \{0, 0.1, \ldots, 0.5\}$; Crank–Nicolson $\Delta t = 0.025$
Navier–Stokes 3D	$u_t + (u\cdot\nabla)u = -\nabla p + \nu\Delta u$, $\nabla\cdot u = 0$ on the periodic cube $[0,1)^3$	u_0 the curl of two potentials $\exp\big(2\sum_i(\cos 2\pi(x_i - c_i) - 1)\big)$ along z and $(1,1,0)/\sqrt{2}$, amplitude 0.2; centre $c \sim U(0,1)^3$, separation $\sim U(0.12, 0.24)$ along $(1,1,1)/\sqrt{3}$, second strength $\sim U(0.6, 1.4)$; $\nu \sim \log U(0.002, 0.01)$; reference CNAB2 $\Delta t = 0.001$; outputs $t \in \{0, 0.04, \ldots, 0.2\}$

The Poisson and three-dimensional heat CG comparators are unpreconditioned; heat CG is warm-started from the previous time level.

Table C.5: Supporting data for Table 1, including the full-order setting the rule selected for each row: the two settings of each frozen model, median GPU query times (ms) and status. "EQ" is the empirical quadrature rule of Appendix A.6; "tensor" is the precomputed quadratic advection tensor used in three dimensions. With the host transfer of the full field included, the accurate and fast Poisson 2D speedups are 7.05× and 7.72× at 256^2 and 46.3× and 66.2× at 4096^2, and the accurate Poisson 3D speedups are 2.78–3.63× at 64^3–256^3. Heat errors are measured against the exact time evolution on the same grid, so the NM-ROM and CN–CG errors both include the Crank–Nicolson time error.

Problem	Mesh	Accurate	Fast	Accurate ms	Fast ms	FOM setting	FOM ms	Status
Poisson 2D	256^2	span, R'=512	span, R'=128	0.56	0.38	CG, rtol 3×10^{-2}	10.3	test
Poisson 2D	1024^2	span, R'=512	span, R'=128	3.10	1.18	CG, rtol 3×10^{-2}	96.4	test
Poisson 2D	2048^2	span, R'=512	span, R'=128	11.66	3.40	CG, rtol 3×10^{-2}	751.0	test
Poisson 2D	4096^2	span, R'=512	span, R'=128	41.92	12.11	CG, rtol 10^{-1}	4508.8	test
Poisson, L-shape 2D	256^2	span, R'=128	span, R'=64	0.32	0.30	CG, rtol 10^{-2}	10.3	test
Poisson, L-shape 2D	512^2	span, R'=128	span, R'=64	0.52	0.45	CG, rtol 10^{-2}	26.2	test
Poisson, L-shape 2D	1024^2	span, R'=128	span, R'=64	2.05	1.73	CG, rtol 3×10^{-2}	85.2	test
Poisson, L-shape 2D	2048^2	span, R'=128	span, R'=64	3.81	3.06	CG, rtol 3×10^{-2}	298.5	test
Heat 2D	1024^2	span, R'=128	span, R'=32	0.64	0.34	CN–CG, Δt=0.025, rtol 10^{-4}	67.00	test
Heat 2D	2048^2	span, R'=128	span, R'=32	1.96	0.83	CN–CG, Δt=0.025, rtol 10^{-4}	424.37	test
Heat 2D	4096^2	span, R'=128	span, R'=32	8.42	3.97	CN–CG, Δt=0.025, rtol 10^{-3}	1945.97	test
Burgers 2D	256^2	span, R'=384, M=1536, EQ	span, R'=96, EQ	45.2	14.2	Newton–BiCGStab, tol 3×10^{-3}	17.4	test
Burgers 2D	512^2	span, R'=384, M=1536, EQ	span, R'=96, EQ	46.7	15.1	Newton–BiCGStab, tol 3×10^{-3}	29.1	test
Burgers 2D	1024^2	span, R'=384, M=1536, EQ	span, R'=96, EQ	49.1	19.1	Newton–BiCGStab, tol 3×10^{-3}	74.1	test
Burgers 2D	2048^2	span, R'=384, M=1536, EQ	span, R'=128, EQ	90.4	35.2	Newton–BiCGStab, tol 3×10^{-3}	306.8	test
Burgers 2D	4096^2	span, R'=384, M=1536, EQ	span, R'=128, EQ	119.4	46.6	Newton–BiCGStab, tol 3×10^{-3}	1282.5	test
Poisson 3D	32^3	span, R'=128	span, R'=32	0.32	0.27	CG, rtol 10^{-2}	1.41	test
Poisson 3D	64^3	span, R'=128	span, R'=32	0.44	0.30	CG, rtol 10^{-2}	3.57	test
Poisson 3D	128^3	span, R'=128	span, R'=32	0.89	0.42	CG, rtol 10^{-2}	12.1	test
Poisson 3D	256^3	span, R'=128	span, R'=32	6.16	2.50	CG, rtol 3×10^{-2}	118.8	test
Heat 3D	32^3	span, R'=320	span, R'=128	0.31	0.22	CN–CG, Δt=0.025, rtol 10^{-4}	3.38	test
Heat 3D	64^3	span, R'=320	span, R'=64	0.64	0.30	CN–CG, Δt=0.025, rtol 10^{-4}	5.42	test
Heat 3D	128^3	span, R'=320	span, R'=64	2.99	0.83	CN–CG, Δt=0.025, rtol 10^{-4}	17.33	test
Heat 3D	256^3	span, R'=320	span, R'=64	23.61	6.41	CN–CG, Δt=0.025, rtol 10^{-4}	179.42	test
Burgers 3D	64^3	span, R'=512, Δt=0.005, tensor	span, R'=192, Δt=0.01, tensor	81.11	8.28	Newton–BiCGStab, Δt=0.005, tol 10^{-3}	17.26	test
Burgers 3D	128^3	span, R'=512, Δt=0.005, tensor	span, R'=192, Δt=0.01, tensor	85.00	9.54	Newton–BiCGStab, Δt=0.01, tol 10^{-3}	48.68	test
Burgers 3D	256^3	span, R'=512, Δt=0.01, tensor	span, R'=256, Δt=0.01, tensor	90.70	35.50	Newton–BiCGStab, Δt=0.01, tol 10^{-2}	198.69	test
Navier–Stokes 3D	32^3	head, k=8	bank span, R'=16, Δt=0.04, 2 sweeps	5.76	1.50	CNAB2 (spectral), 40 steps	5.33	test
Navier–Stokes 3D	64^3	head, k=8	bank span, R'=16, Δt=0.04, 2 sweeps	8.13	2.78	CNAB2 (spectral), 50 steps	23.72	test
Navier–Stokes 3D	96^3	head, k=8	bank span, R'=16, Δt=0.02, 3 sweeps	15.82	8.83	CNAB2 (spectral), 70 steps	126.78	test

Training/evaluation separation (Burgers). The 4608 training trajectories of the Burgers checkpoint were regenerated from their seeds and compared with every evaluation cohort (the 32 validation cases of Table B.1, the 64 test cases and the 6 development cases): no cohort shares a parameter vector, or any single parameter value, with the training set.

D TUNABILITY LADDERS

Table D.1 gives the bank-width ladders behind §5.3.

Table C.6: Solver iterations per query and where the query time goes. On Poisson and heat the query is a direct solve; *input* is projecting the forcing or fitting the initial state, *solve* the reduced solve including all time steps, and *output* rebuilding the field at the output times. These stage times come from separately compiled stages in the same jobs, so they do not add exactly to the query time. Burgers: Levenberg–Marquardt iterations per query, median over the cases (largest number in one step in brackets): in two dimensions over 50 time steps of the six development cases, every step stopped on the convergence test; in three dimensions over 50 ($\Delta t = 0.005$) or 25 ($\Delta t = 0.01$) steps of the 32 test cases. Navier–Stokes: 10 time steps of three Gauss–Newton sweeps each, plus 12 Gauss–Newton sweeps that fit the head's starting code to the projected initial field, starting from a stored training code. The Poisson 2D, Burgers 2D and Navier–Stokes rows are development runs, so their query times differ from Table C.5; the Burgers 2D 4096^2 row ran on an H200 GPU and the others on A100 GPUs, which is why it is faster than 2048^2. Times in ms.

Problem	Mesh	Setting	Iterations	Input	Solve	Output	Query
Poisson 2D	4096^2	span R'=512	0	0.99	<0.01	41.79	42.39
		span R'=128	0	0.74	<0.01	12.52	12.54
Heat 2D	4096^2	span R'=128	0	1.34	0.07	6.74	8.42
		span R'=48	0	1.10	0.07	3.09	4.74
Heat 3D	256^3	span R'=320	0	11.88	0.08	12.28	23.61
		span R'=128	0	5.81	0.08	5.42	10.68
Burgers 2D	256^2	span R'=384	52 (4)				45.71
		span R'=128	38.5 (4)				17.44
	512^2	span R'=384	52 (4)				45.91
		span R'=128	38.5 (4)				17.39
	1024^2	span R'=384	54 (5)				49.52
		span R'=128	51.5 (5)				21.31
	2048^2	span R'=384	54 (5)				90.09
		span R'=128	51.5 (5)				35.15
	4096^2	span R'=384	54 (5)				59.60
		span R'=128	51.5 (5)				24.06
Burgers 3D	64^3	span R'=512	51 (3)				81.11
		span R'=192	27 (5)				8.28
	128^3	span R'=512	51 (3)				85.00
		span R'=192	26 (5)				9.54
	256^3	span R'=512	28 (5)				90.70
		span R'=256	27 (5)				35.50
Navier–Stokes 3D	32^3	head k=8	30 + 12				6.26
		span R'=16	30				3.65
	64^3	head k=8	30 + 12				8.19
		span R'=16	30				4.64
	96^3	head k=8	30 + 12				15.96
		span R'=16	30				9.16

Table C.7: Supporting data for Table B.1: its 256^2 and 512^2 rows. Our rows use the unoptimised dense reference path (no empirical quadrature, stopping tolerance 10^{-6}), so no times are compared. The shallow masked-autoencoder NM-LSPG of Y. Kim et al. (2022) reproduced its published result on their own 2D Burgers benchmark (their Sec. 6.2) on the third of three planned attempts (sigmoid activation; median $1.45\,\%$ against a $1.5\,\%$ bar; published $<1\,\%$). Its published encoder width ($2n$) needs 136 GB to train at 256^2 and 2183 GB at 512^2, beyond the 77 and 135 GB devices; at 512^2 its encoder, capped at width 4096, trained only 229–231 epochs in its 4800 s budget, so those errors partly reflect a training-time limit. The data-matched row is trained on our bank's 576 trajectories.

Mesh	Method	k	Worst (%)	Median (%)	MB
256^2	Kim et al. NM-LSPG	8	163.93	43.34	2480
256^2	Kim et al. NM-LSPG	16	145.40	43.87	2567
256^2	Kim et al. NM-LSPG	32	127.51	43.61	2742
256^2	Kim et al. NM-LSPG, data-matched	16	120.33	38.71	2567
256^2	POD-LSPG	8	55.52	22.72	18
256^2	POD-LSPG	16	41.53	10.45	30
256^2	POD-LSPG	32	24.78	5.65	55
256^2	This work, head ($k = 16$)	16	6.79	0.66	625
512^2	Kim et al. NM-LSPG	8	175.71	55.48	9959
512^2	Kim et al. NM-LSPG	16	174.99	58.56	10511
512^2	Kim et al. NM-LSPG	32	252.33	56.72	10879
512^2	POD-LSPG	8	55.82	24.08	71
512^2	POD-LSPG	16	42.01	13.83	122
512^2	POD-LSPG	32	25.63	7.74	222
512^2	This work, head ($k = 16$)	16	7.72	0.69	2368

Table C.8: Three-dimensional configurations, read from the run records. Domains are the unit cube with homogeneous Dirichlet data, except Navier–Stokes, which is periodic.

Problem	Mesh	Cases	Reduced model	Named FOM	Residual
Poisson 3D	32^3–256^3	64 test	span $R' \in \{16, \ldots, 128\}$	CG, rtol 10^{-2}, no preconditioner	exact (linear)
Heat 3D	32^3–256^3	64 test (all times)	span $R' \in \{320, \ldots, 32\}$ ($R = 320$)	CN–CG, $\Delta t = 0.025$, rtol 10^{-4} (the rule's choice at every mesh)	exact (linear)
Burgers 3D	64^3, 128^3, 256^3	64 validation (settings); 32 test	coordinate bank $R = 512$; span $R' \in \{512, \ldots, 64\}$, $\Delta t \in \{0.005, 0.01\}$	Newton–BiCGStab, setting per row by the rule	precomputed quadratic tensor, checked on reached states
Navier–Stokes 3D	32^3, 64^3, 96^3 periodic	16 development / 32 test cases	POD bank $R = 64$ built per mesh, in a moving frame; head $k = 8$; span $R' \in \{64, 48, 32, 16, 8\}$	CNAB2 (spectral), 40–70 steps (per row by the rule)	precomputed quadratic tensor

Table D.1: Bank-width tunability at the finest mesh. Each block is one frozen model, and within a cohort every speedup divides the median time of *one* full-order setting (the *FOM* row, the fastest at least as accurate as the block's most accurate setting, in the same allocation) by the setting's time (ms). Error is the worst same-grid relative L^2 error (%), over evolved times for Navier–Stokes and Burgers and over all times for heat. Test cases were never used for selection and share the development settings; for Burgers 3D the development column is the 64 validation cases on which its settings were chosen, and only the two selected widths were run on the test cases. The Burgers 2D test cases ran on an A100 GPU and the development cases on an H200 GPU. Heat is shown at 2048^2.

	Development cohort			Test cohort		
Setting	Err. (%)	ms	Speedup	Err. (%)	ms	Speedup
Poisson 2D, 4096^2 *(12 development / 32 test cases)*						
Head, k=32, R'=512	3.15	43.4	103×	4.48	43.0	105×
Bank span, R'=512	0.74	42.4	106×	0.83	41.9	108×
Bank span, R'=384	0.77	32.5	138×	1.23	32.0	141×
Bank span, R'=256	0.94	22.6	199×	2.05	22.0	205×
Bank span, R'=128	2.31	12.5	358×	4.06	12.1	372×
Bank span, R'=64	5.33	7.1	631×	6.61	6.7	672×
Bank span, R'=32	9.17	4.4	1022×	15.5	4.0	1136×
FOM: CG, rtol 10^{-1}	0.45	4490		0.58	4509	
Navier–Stokes 3D, moving frame, 96^3 *(16 development / 32 test cases)*						
Head, k=8	0.15	16.0	6.67×	0.21	15.8	7.79×
Bank span, R'=64	0.62	15.4	6.92×	1.12	15.3	8.05×
Bank span, R'=48	1.00	12.9	8.27×	1.26	12.6	9.76×
Bank span, R'=32	1.26	11.6	9.14×	1.46	11.5	10.7×
Bank span, R'=16	2.96	9.2	11.6×	3.24	9.2	13.4×
Bank span, R'=8	5.55	8.2	13.0×	6.91	8.1	15.2×
FOM: CNAB2 (spectral), 60 / 70 steps	0.063	106		0.056	123	
Burgers 2D, 4096^2 *(6 development / 64 test cases)*						
Head, k=16, R'=512	2.41	41.5	12.6×	—	—	—
Bank span, R'=512 (quadrature rule failed its check)	0.30	77.6	6.74×	—	—	—
Bank span, R'=384	0.22	59.6	8.78×	1.35	119	10.7×
Bank span, R'=256	0.74	42.2	12.4×	—	—	—
Bank span, R'=128	1.89	24.1	21.7×	3.11	46.6	27.5×
Bank span, R'=64	4.83	16.1	32.6×	8.24	28.8	44.5×
Bank span, R'=32	10.7	12.1	43.2×	—	—	—
FOM: Newton–BiCGStab, tol. 3×10^{-3}	0.050	523		0.14	1282.5	
Burgers 3D, Δt=0.01, 256^3 *(64 validation / 32 test cases)*						
Bank span, R'=512	2.43	89.7	2.22×	2.20	90.7	2.19×
Bank span, R'=384	3.10	58.4	3.41×	—	—	—
Bank span, R'=256	3.84	34.0	5.86×	3.77	35.5	5.60×
Bank span, R'=192	5.65	25.7	7.74×	—	—	—
Bank span, R'=128	10.5	18.0	11.1×	—	—	—
Bank span, R'=96	14.4	13.9	14.3×	—	—	—
Bank span, R'=64	21.9	9.5	20.9×	—	—	—
FOM: Newton–BiCGStab, Δt=0.01, tol. 10^{-2}	1.27	199		1.52	199	
Heat 2D, Crank–Nicolson, Δt=0.025, 2048^2 *(16 test cases)*						
Head, k=8, R'=128	—	—	—	1.36	31.5	13.5×
Bank span, R'=128	—	—	—	0.13	1.96	216×
Bank span, R'=96	—	—	—	0.15	1.63	261×
Bank span, R'=64	—	—	—	0.19	1.28	332×
Bank span, R'=48	—	—	—	0.28	1.05	405×
Bank span, R'=32	—	—	—	1.08	0.83	512×
Bank span, R'=16	—	—	—	8.68	0.62	687×
FOM: CN–CG, $\Delta t = 0.025$, rtol 10^{-4}	—	—	—	0.045	424	