LEARNING LARGE-STEP FLUID DYNAMICS WITH NAVIER–STOKES STRUCTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

While the Navier–Stokes equations provide a continuous-time description of fluid motion, learning discrete-time fluid dynamics directly from observations remains highly challenging. Over large time intervals, fluid evolution becomes strongly nonlinear, making it difficult for purely data-driven neural networks to recover the underlying dynamical mechanisms. Consequently, existing methods may fit observed trajectories well with low per-step errors, yet still suffer from rapidly accumulating errors during autoregressive rollout and often generalize poorly beyond the training distribution. We introduce NeNS^2, a neural surrogate for incompressible flow that learns large-step dynamics with Navier–Stokes structure. On rectangular domains with periodic or free-slip boundaries, NeNS^2 represents the flow using analytical eigenfunctions of the Stokes operator, thereby satisfying incompressibility and boundary conditions by construction. In this modal basis, the viscous term is naturally diagonalized. Guided by the algebraic structure of the equations, we parameterize the remaining nonlinear dynamics with a low-rank skew-symmetric bilinear operator. This operator is learned directly from finite-time state pairs, without requiring physics-informed losses, while inherently preserving the kinetic-energy conservation property of the nonlinear term. On in-domain benchmarks, NeNS^2 reduces held-out relative vorticity error by approximately 30%, compared with the strongest evaluated baselines. Moreover, although trained only on randomized initial conditions, NeNS^2 exhibits strong out-of-distribution generalization, achieving the lowest errors on independently generated leapfrogging-vortex and Taylor-vortex dynamics.

1 INTRODUCTION

Learning accurate surrogates for the incompressible Navier–Stokes equations offers a way to reduce the cost of repeated flow simulations. Although neural operators and Transformer models have made substantial progress in learning PDE solution maps from data (Li et al., 2021; Raonic et al., 2023; Holzschuh et al., 2025), accurately learning fluid evolution remains challenging, especially over large prediction intervals. The advective term in the Navier–Stokes equations is quadratic in velocity, but integrating these interactions over time generally produces a finite-time map with higher-order nonlinear dependence on the initial state. Over large intervals, this accumulated nonlinear evolution makes it difficult to infer the underlying dynamical mechanisms from observed state transitions alone. A useful surrogate must nevertheless remain accurate over long rollouts beyond the training horizon. The challenge is further amplified in autoregressive prediction, where each prediction becomes the next input, allowing errors to alter subsequent transport and accumulate over time. Thus, existing data-driven surrogates may require large data or costly long-horizon objectives, yet still struggle with long-horizon accuracy and generalization to flow configurations outside the training distribution.

Encoding known Navier–Stokes structure into the surrogate rules out candidate models that violate physical constraints, thereby narrowing the search space for learning. Because these constraints also apply beyond the training trajectories, they provide an inductive bias for rollout accuracy and generalization. A suitable choice of coordinates makes such constraints explicit. For example, eigenfunction fluid simulation represents velocity as a weighted sum of divergence-free spatial modes and derives equations for the time-dependent weights by projecting the fluid equations onto these modes (De Witt et al., 2012). In this representation, fluid evolution becomes a set of interactions between modal coefficients. Previous studies (Peherstorfer & Willcox, 2016; Benner et al., 2021) have confirm that modal interactions can also be learned from data, with a reduced-order model

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.

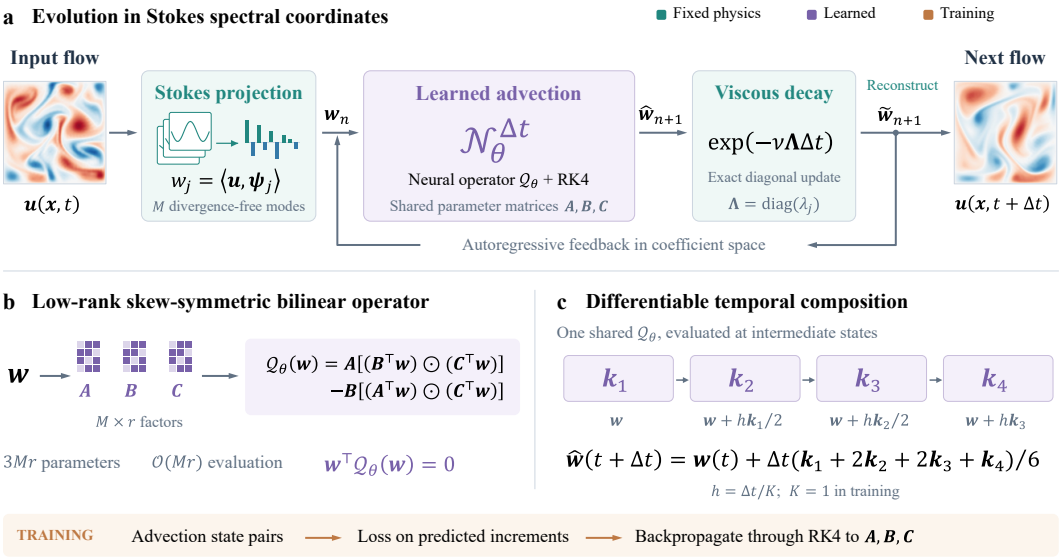


Figure 1: **NeNS^2 overview.** **(a)** Projection onto a Stokes eigenbasis gives modal coefficients. Learned advection and exact modal viscous decay advance the state, which is reconstructed as a velocity field and fed into the next step. **(b)** Three low-rank factors parameterize the nonlinear dynamics with $w^\top Q_\theta(w) = 0$. **(c)** RK4 stages share these factors, and the finite-time increment loss is differentiated through their composition. Global normalization is omitted here.

(ROM) specified. These observations inspire us to develop a neural surrogate for large-step fluid dynamics in an ROM manner.

We introduce NeNS^2, a neural surrogate that learns large-step fluid dynamics by composing a shared interaction law with Navier–Stokes structure (Fig. 1). On rectangular domains with periodic or free-slip boundaries, analytical eigenfunctions of the Stokes operator represent the velocity field, satisfying incompressibility and boundary conditions by construction. This basis also diagonalizes the viscous term, allowing exact modal decay to be handled separately. We parameterize the remaining nonlinear dynamics with a *low-rank skew-symmetric bilinear operator*. The factorization preserves the kinetic-energy conservation property of nonlinear advection for every choice of learned weights, while using $\mathcal{O}(Mr)$ parameters and operations per evaluation for M retained modes and rank r. This makes structured modal interactions tractable without storing a dense quadratic tensor.

To connect this interaction law to finite-time observations, we train through a differentiable fourth-order Runge–Kutta (RK4) scheme. Its stages reuse the same operator at intermediate states, composing quadratic interactions into an approximation of the nonlinear large-step map. Finite-time advection state pairs supervise this entire evolution, without requiring instantaneous derivatives or physics-informed losses. We evaluate NeNS^2 on incompressible-flow benchmarks. Relative to the strongest evaluated neural baselines, it reduces held-out relative vorticity error by approximately 30%. Although trained only on randomized initial conditions, NeNS^2 also achieves the lowest errors on independently generated leapfrogging-vortex and Taylor-vortex dynamics outside the training distribution. Ablations validate the architecture of our model, demonstrating that the predictive gains cannot be achieved by generic neural operators trained on the same modal coefficients.

2 Fluid Dynamics in Stokes Spectral Coordinates

We consider the incompressible Navier–Stokes equations on the rectangular domain:

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} = -\boldsymbol{\nabla} p + \nu \Delta \boldsymbol{u} + \boldsymbol{f}, \qquad \boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \tag{1}$$

where $\boldsymbol{u}(\boldsymbol{x}, t)$ is the velocity field, $p(\boldsymbol{x}, t)$ is the pressure, $\nu > 0$ is the kinematic viscosity, and $\boldsymbol{f}(\boldsymbol{x}, t)$ is the external force. Throughout this work, we equip the same rectangular domain with one of two boundary conditions. In the periodic setting, opposite faces are identified, so that Ω represents a flat torus. In the free-slip setting, the velocity satisfies $u_n = 0$ and $\partial_n \boldsymbol{u}_\tau = 0$ on each boundary face, imposing both zero normal velocity and zero tangential shear. Our goal is to rewrite Eq. (1) as a finite-dimensional dynamical system in a basis adapted to incompressibility and either boundary condition.

Stokes Eigenbasis. To eliminate the pressure from the state representation, we apply the divergence-free projection and work with the Stokes operator $\mathcal{A} = -\mathbb{P}\Delta$, where $\mathbb{P}$ denotes the Leray projector onto admissible divergence-free vector fields (Temam, 2024). The periodic and free-slip settings have an especially useful property: the vector Laplacian maps the operator domain into the admissible divergence-free space. Consequently,

$$\mathcal{A}\boldsymbol{u} = -\Delta\boldsymbol{u} \tag{2}$$

on the domain of $\mathcal{A}$; in other words, the Stokes operator coincides with the componentwise vector Laplacian. Let $\{\boldsymbol{\psi}_j\}_{j\geq 1}$ be an $L^2(\Omega)$-orthonormal family of eigenfunctions of $\mathcal{A}$,

$$\mathcal{A}\boldsymbol{\psi}_j = \lambda_j\boldsymbol{\psi}_j, \qquad 0 \leq \lambda_1 \leq \lambda_2 \leq \cdots. \tag{3}$$

In the equivalent pressure formulation, $-\Delta\boldsymbol{\psi}_j + \nabla\pi_j = \lambda_j\boldsymbol{\psi}_j$, the auxiliary modal pressure π_j may therefore be chosen constant and its gradient vanishes.

The same structure gives an analytic eigenbasis. In the periodic setting, for $\boldsymbol{k} \in \mathbb{Z}^d$ let $\kappa_i = 2\pi k_i/\ell_i$. The modes are divergence-free Fourier fields $\boldsymbol{\psi}_{\boldsymbol{k},\boldsymbol{a}}(\boldsymbol{x}) = \boldsymbol{a}e^{\mathrm{i}\boldsymbol{\kappa}\cdot\boldsymbol{x}}$ with $\boldsymbol{\kappa}\cdot\boldsymbol{a} = 0$ and eigenvalue $\lambda_{\boldsymbol{k}} = |\boldsymbol{\kappa}|^2$ (or their real-valued sine–cosine counterparts). In the free-slip setting, for $\boldsymbol{k} \in \mathbb{N}_0^d$ instead let $\kappa_i = \pi k_i/\ell_i$. Up to normalization, the modes have components

$$[\boldsymbol{\psi}_{\boldsymbol{k},\boldsymbol{a}}(\boldsymbol{x})]_i = a_i \sin(\kappa_i x_i) \prod_{j\neq i} \cos(\kappa_j x_j), \qquad \boldsymbol{\kappa}\cdot\boldsymbol{a} = 0, \tag{4}$$

with eigenvalue $\lambda_{\boldsymbol{k}} = |\boldsymbol{\kappa}|^2$, retaining only nonzero fields. The sine and cosine factors enforce the normal and tangential free-slip conditions, respectively, while the polarization constraint enforces incompressibility. Hence the basis, its eigenvalues, and its normalization can all be constructed without a numerical eigensolve (Orszag, 1971; De Witt et al., 2012). See Appendix A for details. The self-adjoint, nonnegative Laplacian also makes the viscous term exactly diagonal in these coordinates. The periodic experiments use the zero-mean subspace, excluding spatially constant modes.

Truncated Spectral ROM. Expanding the velocity field in the Stokes basis gives

$$\boldsymbol{u}(\boldsymbol{x}, t) = \sum_{j=1}^{\infty} w_j(t)\boldsymbol{\psi}_j(\boldsymbol{x}). \tag{5}$$

In practice we build a reduced-order model (ROM) by retaining only the first M modes (Hussaini & Zang, 1986). The truncation is performed in increasing order of Stokes eigenvalue λ_j, which in both boundary settings orders the basis from low to high spatial frequency. Indeed, the analytic formulas above identify λ_j with the squared magnitude of the corresponding (periodic or sine–cosine) wavenumber. Collecting the retained coefficients into $\boldsymbol{w}(t) = [w_1(t), \ldots, w_M(t)]^\top \in \mathbb{R}^M$ yields a finite-dimensional state representation. As shown in Appendix B.1, Projecting Eq. (1) onto the truncated Stokes basis converts the PDE into an ODE system for the spectral coefficients (De Witt et al., 2012),

$$\dot{w}_i = -\nu\lambda_i w_i + \sum_{j=1}^{M}\sum_{k=1}^{M} b_{ijk} w_j w_k + g_i(t), \qquad i = 1, \ldots, M, \tag{6}$$

where $b_{ijk} = -\langle\boldsymbol{\psi}_i, (\boldsymbol{\psi}_j\cdot\nabla)\boldsymbol{\psi}_k\rangle$ encodes the quadratic advection interaction and $g_i(t) = \langle\boldsymbol{\psi}_i, \boldsymbol{f}(\cdot, t)\rangle$ is the projected forcing. Equivalently, in vector form,

$$\dot{\boldsymbol{w}} = -\nu\boldsymbol{\Lambda}\boldsymbol{w} + \mathcal{B}(\boldsymbol{w}, \boldsymbol{w}) + \boldsymbol{g}(t), \tag{7}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_M)$ and the bilinear map $\mathcal{B}$ is defined componentwise by $[\mathcal{B}(\boldsymbol{w}, \boldsymbol{w})]_i = \sum_{j,k=1}^{M} b_{ijk} w_j w_k$. This ROM is the starting point for our learning problem. In particular, when viscosity and forcing are omitted, the dynamics reduce to a purely quadratic interaction law on $\boldsymbol{w}$ with kinetic energy conserved (see Appendix B.2). This structure is exploited by our parameterization.

3 Learning a Skew-Symmetric Bilinear Operator

Our goal is to learn a neural evolution operator $\mathcal{N}_\theta^{\Delta t} : \mathbb{R}^M \to \mathbb{R}^M$ that directly predicts the modal advection update over a large time interval Δt, with viscous diffusion applied analytically (see Appendix B.5) through operator splitting:

$$\widehat{\boldsymbol{w}}(t + \Delta t) = \mathcal{N}_\theta^{\Delta t}\left(\boldsymbol{w}(t)\right), \qquad \widetilde{\boldsymbol{w}}(t + \Delta t) = \exp(-\nu\boldsymbol{\Lambda}\Delta t)\,\widehat{\boldsymbol{w}}(t + \Delta t). \tag{8}$$

Rather than parameterizing the finite-time advection as an unconstrained network, we construct it from a learnable instantaneous dynamics, $\dot{\boldsymbol{w}} = \mathcal{Q}_\theta(\boldsymbol{w}, \boldsymbol{w})$, that explicitly encodes the quadratic

interaction structure and energy-conserving symmetry of modal fluid advection. We then embed $\mathcal{Q}_\theta$ in a differentiable RK4 integrator and define the resulting finite-time map as $\mathcal{N}_\theta^{\Delta t}$.

3.1 Learnable Low-Rank Skew-Symmetric Bilinear Dynamics

The quadratic interaction law of the reduced dynamics naturally admits a bilinear representation. We therefore model the instantaneous evolution through a learnable bilinear map $\mathcal{Q}_\theta : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}^M$, such that evaluating the map twice on the same state, $\mathcal{Q}_\theta(\boldsymbol{w}, \boldsymbol{w})$, recovers the quadratic dependence required by modal advection dynamics.

A general bilinear map is represented by a dense third-order interaction tensor and therefore requires $\mathcal{O}(M^3)$ parameters. Such a representation quickly becomes impractical as the number of retained modes increases. Instead, we first consider the rank-r factorization

$$\mathcal{Q}_\theta^{\mathrm{raw}}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{A}\left[(\boldsymbol{B}^\top \boldsymbol{u}) \odot (\boldsymbol{C}^\top \boldsymbol{v})\right], \tag{9}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{M \times r}$ are trainable parameter matrices and r denotes the interaction rank. The superscript raw denotes the factorization before antisymmetrization. The factorization reduces both the number of learnable parameters and the evaluation cost from their dense counterparts to $\mathcal{O}(Mr)$.

Because the Stokes basis is orthonormal, the kinetic energy of the modal state is $E(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2/2$. Energy conservation therefore requires

$$\frac{\mathrm{d}}{\mathrm{d}t} E(\boldsymbol{w}) = \boldsymbol{w}^\top \mathcal{Q}_\theta(\boldsymbol{w}, \boldsymbol{w}) = 0 \tag{10}$$

for every $\boldsymbol{w}$. We enforce this property by antisymmetrizing the low-rank factorization and define our learnable bilinear dynamics as

$$\mathcal{Q}_\theta(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{A}\left[(\boldsymbol{B}^\top \boldsymbol{u}) \odot (\boldsymbol{C}^\top \boldsymbol{v})\right] - \boldsymbol{B}\left[(\boldsymbol{A}^\top \boldsymbol{u}) \odot (\boldsymbol{C}^\top \boldsymbol{v})\right]. \tag{11}$$

As proved in Appendix B.3, every member of the learnable model family satisfies the instantaneous energy-conservation constraint by construction. No conservation loss is required during training.

3.2 From Learnable Instantaneous Dynamics to a Large-Step Neural Operator

While $\mathcal{Q}_\theta$ parameterizes the instantaneous vector field, the quantity to be learned from data is the finite-time evolution over Δt. The learned trajectory starts from $\widehat{\boldsymbol{w}}(t) = \boldsymbol{w}(t)$ and formally satisfies

$$\widehat{\boldsymbol{w}}(t + \Delta t) = \boldsymbol{w}(t) + \int_t^{t+\Delta t} \mathcal{Q}_\theta\left(\widehat{\boldsymbol{w}}(\tau), \widehat{\boldsymbol{w}}(\tau)\right) \mathrm{d}\tau. \tag{12}$$

Because $\widehat{\boldsymbol{w}}(\tau)$ changes throughout the interval, evaluating the learned bilinear dynamics only at the initial state is insufficient for approximating this finite-time map (see Appendix B.4). We therefore construct the large-step neural operator $\mathcal{N}_\theta^{\Delta t}$ by integrating the learnable vector field with a differentiable RK4 scheme. Given an input modal state $\boldsymbol{w}(t)$, its four RK4 stages are

$$\begin{aligned}
\boldsymbol{k}_1 &= \mathcal{Q}_\theta\left(\boldsymbol{w}(t), \boldsymbol{w}(t)\right), & \boldsymbol{k}_2 &= \mathcal{Q}_\theta\left(\boldsymbol{w}(t) + \frac{\Delta t}{2}\boldsymbol{k}_1,\, \boldsymbol{w}(t) + \frac{\Delta t}{2}\boldsymbol{k}_1\right), \\
\boldsymbol{k}_3 &= \mathcal{Q}_\theta\left(\boldsymbol{w}(t) + \frac{\Delta t}{2}\boldsymbol{k}_2,\, \boldsymbol{w}(t) + \frac{\Delta t}{2}\boldsymbol{k}_2\right), & \boldsymbol{k}_4 &= \mathcal{Q}_\theta\left(\boldsymbol{w}(t) + \Delta t\, \boldsymbol{k}_3,\, \boldsymbol{w}(t) + \Delta t\, \boldsymbol{k}_3\right).
\end{aligned} \tag{13}$$

The resulting large-step prediction is

$$\widehat{\boldsymbol{w}}(t + \Delta t) = \mathcal{N}_\theta^{\Delta t}\big(\boldsymbol{w}(t)\big) = \boldsymbol{w}(t) + \frac{\Delta t}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right). \tag{14}$$

Importantly, RK4 is part of the neural computation graph. The same trainable bilinear operator $\mathcal{Q}_\theta$ is repeatedly evaluated at the intermediate states, and gradients of the large-step prediction propagate through all RK4 stages to the parameter matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$. Given finite-time advection training pairs $(\boldsymbol{w}(t), \boldsymbol{y}(t{+}\Delta t))$, where $\boldsymbol{y}$ is the reference advection target before viscous decay, the parameters are optimized end-to-end through the predicted increment,

$$\theta^\star = \arg\min_\theta \sum_i \mathcal{L}\left(\mathcal{N}_\theta^{\Delta t}\big(\boldsymbol{w}_i(t)\big) - \boldsymbol{w}_i(t), \boldsymbol{y}_i(t + \Delta t) - \boldsymbol{w}_i(t)\right), \tag{15}$$

where $\mathcal{L}$ is the mode-weighted relative squared error of the advection increment as detailed in Appendix D.1. Other normalization and optimization details are also given in Appendix D.

4 EXPERIMENTS

We evaluate NeNS^2 through a comprehensive set of experiments on both free-slip and periodic flows, with dataset construction and training settings detailed in Appendices C and D and evaluation protocols and reproducibility details in Appendix E. We first compare its prediction accuracy against representative neural baselines under consistent training and evaluation settings (Sec. 4.1), where NeNS^2 achieves the best overall accuracy. We then conduct ablation studies to assess the contributions of its key components (Sec. 4.2), confirming that the proposed design choices are important for both prediction accuracy and long-term stability. In addition, we compare NeNS^2 with the traditional spectral solver under the same computational resources (Sec. 4.3), demonstrating higher inference efficiency. Finally, we examine its generalization capability on challenging out-of-distribution flow configurations that are entirely absent from the training data (Sec. 4.4), where NeNS^2 maintains strong predictive accuracy and produces flow dynamics that closely match the ground truth.

4.1 COMPARISON WITH LEARNED DYNAMICS MODELS

We compare NeNS^2 with U-Net (Ronneberger et al., 2015), CNO (Raonic et al., 2023), FNO (Li et al., 2021), PDE-Transformer (PDE-T) (Holzschuh et al., 2025), and LNO (Wang & Wang, 2024). For a fair comparison, all methods are trained and evaluated on the same data splits under matched GPU-hour budgets. We compare each baseline under a parameter budget comparable to NeNS^2, while for PDE-T and LNO we additionally report results using the configurations recommended in their original papers. We evaluate all methods under two settings: an in-distribution (ID) setting using held-out test data drawn from the training distribution, and an out-of-distribution (OOD) setting consisting of structured vortex configurations with dynamics qualitatively different from the randomized training flows. Baseline configurations are detailed in Appendix E.3, and the construction of the OOD test cases is described in Appendix E.2.

Prediction accuracy is evaluated under closed-loop autoregressive rollout using the relative vorticity error. For trajectory i at rollout time t, we define

$$e_\omega^{(i)}(t) = \frac{\|\widetilde{\omega}^{(i)}(t) - \omega^{(i)}(t)\|_2}{\|\omega^{(i)}(t)\|_2}, \tag{16}$$

where $\widetilde{\omega}^{(i)}(t)$ and $\omega^{(i)}(t)$ denote the predicted and ground-truth vorticity fields, respectively. For NeNS^2, $\widetilde{\omega}$ is reconstructed after the analytic viscous update. We report the mean relative vorticity error over all evaluated trajectories and rollout steps (Appendix E.4).

4.1.1 FREE-SLIP FLOW

On the free-slip flow, NeNS^2 consistently achieves the lowest prediction error among all learning-based baselines over long autoregressive rollouts, as shown in Table 1. As shown in Figure 3, the advantage becomes substantially more pronounced on the out-of-distribution leapfrogging case, where NeNS^2 remains accurate while competing methods exhibit much larger accumulated errors or become unstable. These results indicate that the structured modal dynamics improve not only in-distribution prediction accuracy, but also generalize more reliably to qualitatively different flows.

4.1.2 PERIODIC FLOW

As shown in Table 1, a similar trend is observed for periodic flows, where NeNS^2 again achieves the best overall prediction accuracy on both held-out trajectories and the out-of-distribution Taylor-vortex scenes. Beyond the quantitative improvement, NeNS^2 better captures the underlying flow behavior: in the vortex-merging case, it reproduces the correct merging dynamics, whereas PDE-T predicts separation, as visualized in Fig. 3. Several baselines also exhibit severe error accumulation or numerical instability over long rollouts, while NeNS^2 remains stable throughout the evaluation. Together, these results demonstrate that the proposed structured dynamics generalize across boundary conditions and remain reliable over long-term evolution.

4.2 ABLATION AND STRUCTURAL ANALYSIS

We conduct ablation studies on the free-slip scenarios to isolate the design choices in NeNS^2, with results summarized in Table 2. First, we assess the skew-symmetric constraint by replacing the proposed bilinear operator with a parameter-matched unconstrained counterpart. This consistently degrades accuracy on both in-distribution and leapfrogging-OOD cases. Second, we examine the temporal integration scheme by replacing RK4 with a single Euler step. The resulting model exhibits

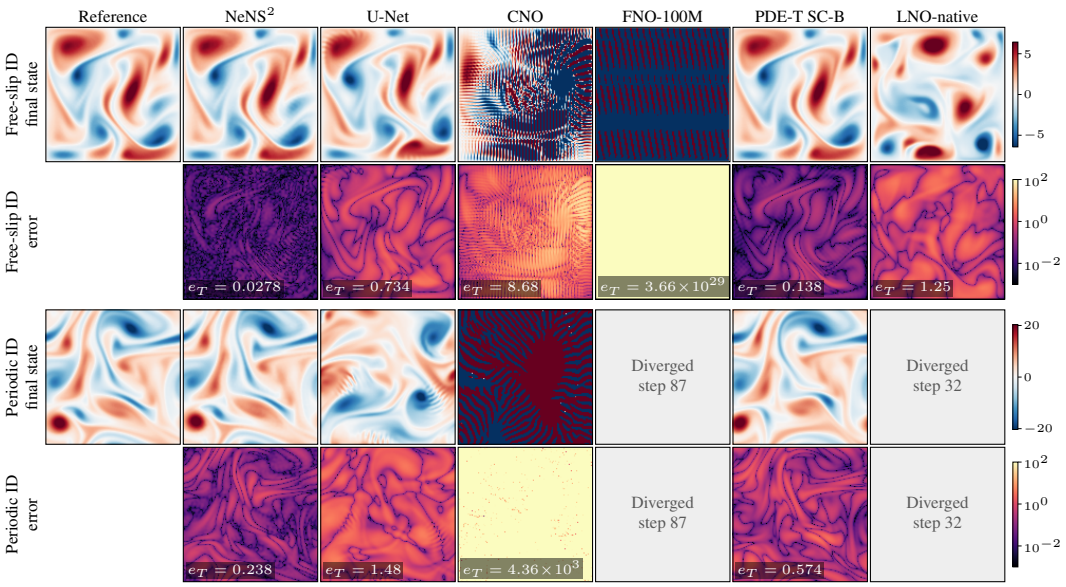


Figure 2: **Comparison on held-out trajectories.** This figure illustrates qualitative comparison of final-state predictions on representative held-out trajectories. NeNS^2 produces predictions that most closely match the reference solutions in both free-slip and periodic settings, with visibly lower spatial error than the competing methods. The top two rows show the free-slip case and the bottom two rows show the periodic case. For each boundary condition, the first row compares the reference and predicted vorticity fields at final state, while the second row shows the corresponding normalized pointwise error $|\widetilde{\omega} - \omega| / \mathrm{RMS}(\omega)$ on a logarithmic scale. All methods share the same color scale within each boundary condition. The inset value e_T denotes the relative L^2 error at the final time. Gray panels indicate diverged rollouts, with the first non-finite step reported. For methods with multiple evaluated variants, only the best-performing variant is shown for clarity and conciseness.

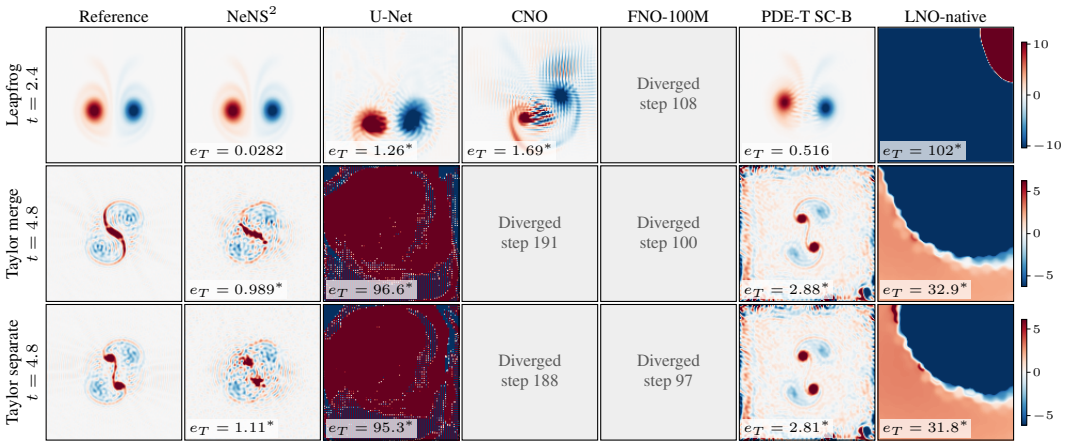


Figure 3: **Comparison on out-of-distributon cases**. This figure illustrates qualitative comparison of final-state predictions on the three out-of-distribution cases. NeNS^2 most closely matches the reference solutions across all three cases, while the competing methods exhibit substantially larger errors or numerical divergence. Notably, even PDE-T SC-B, the strongest competing variant despite its substantially larger parameter count, shows a clear deviation from the reference in the leapfrogging case and predicts separation rather than the correct merging behavior in the Taylor-vortex merging case. Within each row, all methods are shown using the same symmetric color scale centered at zero, determined from the corresponding reference field. The inset value e_T denotes the relative L^2 error at the final state, whereas Table 1 reports the average error over the full rollout. An asterisk indicates that the prediction is substantially clipped by the shared display range, and gray panels indicate diverged rollouts with the first non-finite step reported. For methods with multiple evaluated variants, only the best-performing variant is shown for clarity and conciseness.

Table 1: **Vorticity error.** Closed-loop relative vorticity error on in-distribution (ID) and out-of-distribution (OOD) flow prediction is reported. ID results are averaged over 200 held-out trajectories for each boundary condition; OOD evaluation uses leapfrogging for the free-slip setting and Taylor-vortex merging and separation for the periodic setting. PDE-T SC-100 is scaled to approximately match the parameter count of NeNS^2, while PDE-T SC-B uses the original Base configuration ($\sim$178M parameters), about $1.8\times$ larger than NeNS^2. PDE-T SC-B zero-shot directly evaluates the released pretrained model without training or fine-tuning on our data; PDE-T SC-B is trained from scratch; and PDE-T SC-B + pretrain is initialized from the released pretrained weights and then trained on our data. LNO-native follows the original configuration ($\sim$5M parameters), and LNO-100M scales the architecture to a parameter budget comparable to NeNS^2. Notably, even PDE-T SC-B, despite using substantially more parameters, remains less accurate than NeNS^2 across all reported settings. *diverged* indicates that the rollout becomes non-finite.

Model	Free-slip		Periodic		
	ID ↓	Leapfrog ↓	ID ↓	Taylor vortex	
				Merge ↓	Separate ↓
NeNS^2	**0.0276**	**0.0503**	**0.1304**	**0.1995**	**0.2002**
U-Net	0.2195	1.0036	1.1484	15.3323	15.6486
CNO	2.4094	1.0830	279.6206	*diverged*	*diverged*
FNO-100M	*diverged*	*diverged*	*diverged*	*diverged*	*diverged*
PDE-T SC-100	0.0477	0.4981	0.2036	1.1846	1.1817
PDE-T SC-B	0.0444	0.4830	0.1732	0.2175	0.2170
PDE-T SC-B + pretrain	0.0516	0.4352	0.2311	2.4201	2.4133
LNO-native	0.8281	46.0909	*diverged*	7.7236	7.6934
LNO-100M	*diverged*	*diverged*	2.0603	10.0064	10.2673
PDE-T SC-B zero-shot	1.0777	1.1087	1.1881	0.9181	0.9180

unstable long-term rollouts and eventually diverges. Third, we test whether the gains can be explained simply by operating in coefficient space by replacing the bilinear architecture with parameter-matched FNO1D and U-Net1D models on the same modal coefficients. Both alternatives are substantially less accurate, indicating that the bilinear parameterization provides an important inductive bias beyond the coefficient-space representation itself.

4.3 Efficiency against Classical Spectral Simulation

We compare the inference efficiency of NeNS^2 with the traditional spectral solver (De Witt et al., 2012) on a single NVIDIA A100 GPU; its numerical construction and timing protocol are detailed in Appendices C.1, C.2 and E.6. We report wall-clock time per large step for single-trajectory ($B = 1$) and batched ($B = 32$) execution, with corresponding rollout errors provided in Appendix E.6. As shown in Table 3, NeNS^2 consistently achieves substantially lower inference latency on both free-slip and periodic flows, with an even larger advantage under batched execution.

4.4 Long-Term Generalization

A central goal of NeNS^2 is to serve as a practical surrogate for flow simulation, which requires reliable long-term evolution beyond the training distribution. We therefore evaluate the frozen checkpoint on unseen flow configurations, and then further challenge it by introducing fixed solid obstacles that are absent from the training data. Passive dye is transported by each method's predicted velocity field to reveal accumulated differences in long-term material evolution.

Generalization to unseen flow configurations. Fig. 5 shows a layered rosette in a rotating flow and a patterned bunny in a shearing flow. Starting from identical modal states, NeNS^2 and the traditional spectral solver evolve independently. After 240 steps ($T = 4.8$), their rollouts remain closely aligned, with final velocity relative errors of 0.087% and 0.066%, and dye errors of 0.624% and 0.821%, respectively. The close agreement indicates accurate long-term evolution on previously unseen flow configurations.

Flow evolution with unseen geometric constraints. We further introduce stationary solid obstacles, with the same spectral boundary treatment applied to NeNS^2 and the traditional spectral solver. Fig. 4 shows a vortex pair deflecting around a single cylinder and another passing through a narrow two-cylinder gap. Despite these unseen geometric constraints, the learned rollouts remain visually closely aligned with the reference in both vortex motion and material transport.

Table 2: **Ablation study.** The structural components and coefficient-space architecture of NeNS^2 on the free-slip scenarios are investigated. All variants are evaluated using the same 120-step closed-loop relative vorticity error as in Table 1. ID results are averaged over 200 held-out trajectories, while OOD results are reported on the leapfrogging trajectory. All models contain approximately 100M parameters. The results show that both the skew-symmetric structure and RK4 integration are important to the accuracy and stability of NeNS^2, while generic coefficient-space networks perform substantially worse. *diverged* indicates that the rollout becomes non-finite.

Variant	ID ↓	Leapfrog OOD ↓
NeNS^2	**0.0276**	**0.0503**
No skew	0.0370	0.1124
No RK4 (Euler)	*diverged*	*diverged*
Coefficient FNO1D	1.1369	1.2961
Coefficient U-Net1D	1.2829	1.0992

Table 3: **Inference efficiency.** On both free-slip and periodic flows, We report the median wall-clock time required to advance one large time step on a single NVIDIA A100 GPU under matched computational settings. $B = 1$ denotes single-trajectory inference, while $B = 32$ reports the latency for advancing a batch of 32 trajectories. NeNS^2 is consistently faster than the traditional spectral solver in both settings, with particularly large gains under batched execution.

Setting	Method	$B = 1$ (ms)	$B = 32$ (ms)
free-slip	NeNS^2	**3.837**	**6.634**
	Classical	57.213	82.883
periodic	NeNS^2	**3.823**	**6.628**
	Classical	46.806	365.066

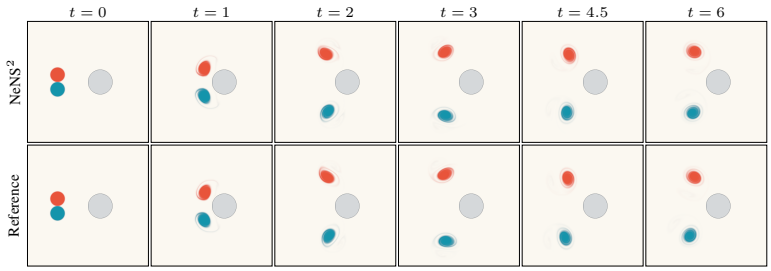
5 Related Work

Neural surrogates and neural operators. Neural surrogates for PDEs commonly learn evolution directly in discretized physical fields. U-Net (Ronneberger et al., 2015) provides a multiscale convolutional architecture, while neural operators such as FNO (Li et al., 2021), CNO (Raonic et al., 2023), LNO (Wang & Wang, 2024), and PDE-Transformer (Holzschuh et al., 2025) learn mappings between fields using spectral, convolutional, latent, or attention-based representations. These models form the principal data-driven baselines in our experiments. Other approaches learn corrections within a CFD solver (Kochkov et al., 2021) or add PDE constraints to operator learning (Li et al., 2024). Autoregressive PDE solvers have also used message passing (Brandstetter et al., 2022) and iterative refinement (Lippe et al., 2023) to improve long rollouts. NeNS^2 instead represents velocity in an admissible Stokes eigenbasis and learns the interactions among its modal coefficients, explicitly encoding incompressibility, boundary conditions, and the algebraic structure of fluid advection.

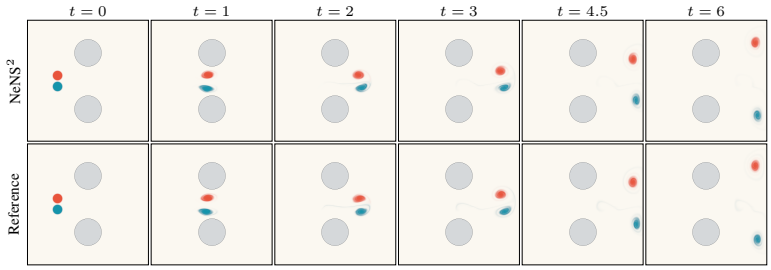
Projection-based ROMs and eigenfunction fluid simulation. Projection-based reduced-order models represent flow in a low-dimensional basis and evolve its coefficients by projecting the governing equations. POD constructs data-dependent modes from flow statistics (Berkooz et al., 2003); such bases have also enabled interactive reduced fluid simulation (Treuille et al., 2006). Spectral approaches instead use bases associated with the underlying differential operators. Eigenfunction fluid simulation derives modal dynamics using Laplacian eigenfunctions (De Witt et al., 2012); later work scales this representation with fast transforms (Cui et al., 2018) and applies it to fluid control (Chen et al., 2024). For the periodic and free-slip domains considered here, Stokes eigenfunctions satisfy the fluid constraints analytically and diagonalize viscosity. NeNS^2 retains this structured representation but learns a low-rank modal interaction law from data instead of evaluating precomputed Galerkin interactions.

Learning reduced dynamics and operator inference. Reduced dynamics can be learned either together with the representation or directly in prescribed reduced coordinates. Nonlinear manifold ROMs learn reduced trial spaces with autoencoders (Lee & Carlberg, 2020), while operator inference identifies polynomial reduced operators from trajectory data (Peherstorfer & Willcox, 2016), including structure-aware formulations for incompressible flow (Benner et al., 2021). NeNS^2 instead uses a fixed analytic basis and learns a factorized quadratic interaction law from finite-time state pairs. By differentiating through a shared RK4 evolution map, it connects finite-time observations to an instantaneous structured generator, closely relating the formulation to neural ODEs (Chen et al., 2018) while retaining the known modal form of Navier–Stokes advection.

Structure-preserving learned dynamics. Physical structure can be imposed directly on learned dynamical systems. Hamiltonian neural networks derive dynamics from a learned Hamiltonian (Greydanus et al., 2019), while recent reduced-order approaches impose energy-preserving structure on learned polynomial operators (Koike & Qian, 2024; Gkimisis et al., 2026). Learned solution operators have also targeted the long-time statistics of dissipative chaotic systems (Li et al., 2022). In particular, skew-symmetric parameterizations provide a natural way to preserve quadratic energy.



(a) Single cylinder;



(b) Centered channel.

Figure 4: **Geometric generalization.** We demonstrate flow evolution under unseen geometric constraints. For each example, the upper row shows the rollout produced by NeNS^2, while the lower row shows the corresponding evolution from the traditional spectral solver at matched times $t = 0, 1, 2, 3, 4.5, 6$. Panels (a) and (b) show a vortex pair deflecting around a single cylinder and passing through a narrow two-cylinder gap, respectively. Passive dye is advected independently by the velocity field produced by each method, with the same color mapping and spectral boundary treatment used throughout. Despite the introduction of obstacle geometries that are entirely absent from training, the learned rollouts remain visually closely aligned with the numerical reference, reproducing the same large-scale vortex motion and material transport.

NeNS^2 combines this principle with a low-rank bilinear factorization in Stokes coordinates, ensuring kinetic-energy conservation of the instantaneous advective dynamics while keeping modal interactions tractable at large dimension. The resulting structured generator is then composed through RK4 to learn large-step evolution from finite-time data.

6 Conclusion & Discussions

We presented NeNS^2, a neural surrogate that learns large-step incompressible fluid dynamics with Navier–Stokes structure. The Stokes eigenbasis enforces incompressibility and boundary conditions while diagonalizing viscosity, and a low-rank skew-symmetric bilinear operator captures quadratic modal interactions while preserving the kinetic-energy conservation property of instantaneous advection. A differentiable RK4 scheme composes this shared interaction law into a large-step evolution map, enabling end-to-end learning from finite-time state pairs without instantaneous derivative supervision or physics-informed losses. Across the evaluated benchmarks, NeNS^2 improves autoregressive rollout accuracy and generalizes from randomized training flows to independently generated vortex configurations outside the training distribution. These results suggest that encoding the structure of instantaneous fluid dynamics and learning through its temporal composition provides an effective inductive bias for recovering large-step evolution from observations.

The analytic Stokes basis currently limits the method to rectangular periodic and free-slip domains, and RK4 does not exactly preserve the continuous energy invariant at the discrete level. A direct extension to three dimensions is currently computationally demanding because the growing number of modes and interactions substantially increases computational and memory costs. More scalable structured interactions, geometry-adaptive bases, and structure-preserving integrators are promising directions for broader neural fluid simulation.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks: to implement methods and to assist with translation. **We have not used generative AI tools for these tasks:** to generate synthetic data set, to help develop theoretical models or conceptual frameworks, to formulate mathematical claims, to provide critical ingredients for proving mathematical claims, to assist in the writing of proofs, to design or provide feedback on research methodology or experiments, or to interpret results. **The following subtasks are not applicable to this work:** to proposing or refine hypotheses, to clean and reformat dataset, and to support qualitative and thematic data analysis. **Additionally, we used generative AI tools for the following purposes:** to create or modify scientific figures or images and to identify relevant literature. **We have reviewed all AI-assisted work.** The AI-generated images serve only as reference material. The versions included in the article are all manually designed. LLM-generated code was verified and tested for correctness by 3 authors. The related works found by AI were also verified for authenticity. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The reported results can be reproduced from the account of the model and experiments in the main text, together with the constructions in the appendix. Secs. 2 and 3 introduce NeNS^2 as a large-step surrogate in Stokes spectral coordinates, in which a low-rank skew-symmetric operator carries the modal interactions and a differentiable integrator advances them over a finite interval. The analytic construction of the eigenbasis, and the derivations that justify this parameterization, are given in Appendices A and B. Building on that formulation, Sec. 4 studies rollout accuracy, the role of the structural choices, computational efficiency, and generalization beyond the training distribution. Appendices C to E record how the periodic and free-slip trajectories are generated, how the model is trained and rolled out, and how those comparisons are evaluated. The source code will be released upon acceptance.

REFERENCES

Peter Benner, Pawan Goyal, Jan Heiland, and Igor Duff. Operator inference and physics-informed learning of low-dimensional models for incompressible flows. *ETNA - Electronic Transactions on Numerical Analysis*, 56:28–51, January 2021. doi: 10.1553/etna_vol56s28.

G Berkooz, P Holmes, and John Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25:539–575, 11 2003. doi: 10.1146/annurev.fl.25.010193.002543.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Yixin Chen, David Levin, and Timothy Langlois. Fluid control with laplacian eigenfunctions. In *ACM SIGGRAPH 2024 Conference Papers*, pp. 1–11, 2024.

Qiaodong Cui, Pradeep Sen, and Theodore Kim. Scalable laplacian eigenfluids. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

Tyler De Witt, Christian Lessig, and Eugene Fiume. Fluid simulation using laplacian eigenfunctions. *ACM Trans. Graph.*, 31(1), February 2012. doi: 10.1145/2077341.2077351.

Leonidas Gkimisis, Igor Pontes Duff, Pawan Goyal, and Peter Benner. On the representation of energy-preserving quadratic operators with application to operator inference. *Applied Mathematics Letters*, 173:109761, 2026. doi: 10.1016/j.aml.2025.109761.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Benjamin Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. Pde-transformer: Efficient and versatile transformers for physics simulations. In *International Conference on Machine Learning*, 2025.

M Yousuff Hussaini and Thomas A Zang. Spectral methods in fluid dynamics. Technical report, 1986.

Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

Tomoki Koike and Elizabeth Qian. Energy-preserving reduced operator inference for efficient design and control. In *AIAA SciTech 2024 Forum*, 2024. doi: 10.2514/6.2024-1012.

Kookjin Lee and Kevin T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020. doi: 10.1016/j.jcp.2019.108973.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Zongyi Li, Miguel Liu-Schiaffini, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Learning chaotic dynamics in dissipative systems. *Advances in Neural Information Processing Systems*, 35:16768–16781, 2022.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

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.

Steven A Orszag. Numerical simulation of incompressible flows within simple boundaries. i. galerkin (spectral) representations. *Studies in applied mathematics*, 50(4):293–327, 1971.

Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016. doi: 10.1016/j.cma.2016.03.025.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of pdes. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 77187–77200. Curran Associates, Inc., 2023. doi: 10.52202/075280-3376.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Cham, 2015. Springer International Publishing. doi: 10.1007/978-3-319-24574-4_28.

Roger Temam. *Navier–Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Society, 2024.

Adrien Treuille, Andrew Lewis, and Zoran Popović. Model reduction for real-time fluids. *ACM Transactions on Graphics (TOG)*, 25(3):826–834, 2006.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse pde problems. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 33085–33107. Curran Associates, Inc., 2024. doi: 10.52202/079017-1042.

A Analytic Construction of the Stokes Eigenbasis

This appendix gives the explicit basis used to represent the velocity field and the construction used to enumerate its modes in the two-dimensional experiments. Let $\Omega = \prod_{i=1}^{d}(0, \ell_i)$ with $d = 2$, and denote its area by $|\Omega| = \prod_i \ell_i$. The velocity inner product is

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{L^2} = \int_\Omega \boldsymbol{u}(\boldsymbol{x})^\top \boldsymbol{v}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \tag{17}$$

All modal amplitudes below refer to velocity modes normalized in this inner product. Consequently, for any expansion $\boldsymbol{u}_M = \sum_{m=1}^{M} w_m \boldsymbol{\psi}_m$, the represented kinetic energy is $\frac{1}{2}\|\boldsymbol{u}_M\|_{L^2}^2 = \frac{1}{2}\|\boldsymbol{w}\|_2^2$.

A.1 Periodic eigenmodes

For the periodic box, associate an integer wave vector $\boldsymbol{k} \in \mathbb{Z}^d$ with the physical wave vector

$$\boldsymbol{\kappa}_{\boldsymbol{k}} = \left(\frac{2\pi k_1}{\ell_1}, \ldots, \frac{2\pi k_d}{\ell_d} \right), \qquad \lambda_{\boldsymbol{k}} = |\boldsymbol{\kappa}_{\boldsymbol{k}}|^2. \tag{18}$$

We exclude $\boldsymbol{k} = \boldsymbol{0}$, corresponding to a constant mean flow, and retain a canonical half-space

$$\mathcal{H} = \{ \boldsymbol{k} \in \mathbb{Z}^d \setminus \{\boldsymbol{0}\} : \text{the first nonzero component of } \boldsymbol{k} \text{ is positive} \}. \tag{19}$$

Thus exactly one of $\boldsymbol{k}$ and $-\boldsymbol{k}$ belongs to $\mathcal{H}$. For each $\boldsymbol{k} \in \mathcal{H}$, choose an orthonormal family $\{\boldsymbol{p}_{\boldsymbol{k},q}\}_{q=1}^{d-1}$ spanning the transverse space

$$\boldsymbol{\kappa}_{\boldsymbol{k}}^\top \boldsymbol{p}_{\boldsymbol{k},q} = 0, \qquad \boldsymbol{p}_{\boldsymbol{k},q}^\top \boldsymbol{p}_{\boldsymbol{k},q'} = \delta_{qq'}. \tag{20}$$

In two dimensions a deterministic choice is

$$\boldsymbol{p}_{\boldsymbol{k},1} = \frac{1}{|\boldsymbol{\kappa}_{\boldsymbol{k}}|}(-\kappa_{\boldsymbol{k},2}, \kappa_{\boldsymbol{k},1})^\top. \tag{21}$$

The real, unit-L^2 modes associated with one wave–polarization pair are

$$\boldsymbol{\psi}_{\boldsymbol{k},q}^{\cos}(\boldsymbol{x}) = \sqrt{\frac{2}{|\Omega|}} \, \boldsymbol{p}_{\boldsymbol{k},q} \cos(\boldsymbol{\kappa}_{\boldsymbol{k}}^\top \boldsymbol{x}), \tag{22}$$

$$\boldsymbol{\psi}_{\boldsymbol{k},q}^{\sin}(\boldsymbol{x}) = \sqrt{\frac{2}{|\Omega|}} \, \boldsymbol{p}_{\boldsymbol{k},q} \sin(\boldsymbol{\kappa}_{\boldsymbol{k}}^\top \boldsymbol{x}). \tag{23}$$

They are divergence-free, periodic, mutually orthonormal, and satisfy $-\Delta\boldsymbol{\psi} = \lambda_{\boldsymbol{k}}\boldsymbol{\psi}$. Both phases therefore inherit the same eigenvalue. Degeneracies also occur between distinct wave vectors of the same physical magnitude.

The periodic data generator stores a complex half-space coefficient $z_{\boldsymbol{k},q}$ and reconstructs the corresponding real field as

$$z_{\boldsymbol{k},q} \boldsymbol{p}_{\boldsymbol{k},q} e^{\mathrm{i}\boldsymbol{\kappa}_{\boldsymbol{k}}^\top \boldsymbol{x}} + \overline{z_{\boldsymbol{k},q}} \boldsymbol{p}_{\boldsymbol{k},q} e^{-\mathrm{i}\boldsymbol{\kappa}_{\boldsymbol{k}}^\top \boldsymbol{x}}. \tag{24}$$

Comparing Eq. (24) with Eq. (23) gives the real unit-basis amplitudes

$$w_{\boldsymbol{k},q}^{\cos} = \sqrt{2|\Omega|}\, \Re z_{\boldsymbol{k},q}, \qquad w_{\boldsymbol{k},q}^{\sin} = -\sqrt{2|\Omega|}\, \Im z_{\boldsymbol{k},q}. \tag{25}$$

The inverse map is $z_{\boldsymbol{k},q} = (w_{\boldsymbol{k},q}^{\cos} - \mathrm{i} w_{\boldsymbol{k},q}^{\sin})/\sqrt{2|\Omega|}$.

Cosine and sine phases are stored consecutively. The experiment-specific numbers of complex amplitudes and real coordinates are given in Appendix C.4.

A.2 Free-slip eigenmodes

For free-slip boundaries, let $\boldsymbol{n} = (n_1, \ldots, n_d) \in \mathbb{N}_0^d$ and define

$$\kappa_i = \frac{\pi n_i}{\ell_i}, \qquad \lambda_{\boldsymbol{n}} = \sum_{i=1}^{d} \kappa_i^2. \tag{26}$$

Given a polarization $\boldsymbol{a} \in \mathbb{R}^d$, define

$$[\boldsymbol{\psi}_{\boldsymbol{n},\boldsymbol{a}}(\boldsymbol{x})]_i = a_i \sin(\kappa_i x_i) \prod_{j \neq i} \cos(\kappa_j x_j). \tag{27}$$

Direct differentiation yields

$$\boldsymbol{\nabla} \cdot \boldsymbol{\psi}_{\boldsymbol{n},\boldsymbol{a}} = (\boldsymbol{\kappa}^\top \boldsymbol{a}) \prod_{j=1}^{d} \cos(\kappa_j x_j). \tag{28}$$

Hence incompressibility is equivalent to $\boldsymbol{\kappa}^\top \boldsymbol{a} = 0$. On a face normal to axis i, the normal component contains $\sin(\kappa_i x_i)$ and vanishes. Every tangential component contains $\cos(\kappa_i x_i)$, whose normal derivative vanishes. The mode therefore satisfies $u_n = 0$ and $\partial_n \boldsymbol{u}_\tau = \boldsymbol{0}$. Applying the Laplacian to the separated product also gives $-\Delta \boldsymbol{\psi}_{\boldsymbol{n},\boldsymbol{a}} = \lambda_{\boldsymbol{n}} \boldsymbol{\psi}_{\boldsymbol{n},\boldsymbol{a}}$.

In two dimensions both indices must be positive. A unit-L^2 polarization is

$$\boldsymbol{a}_{\boldsymbol{n}}^{(2)} = \frac{2}{\sqrt{|\Omega|}} \frac{(-\kappa_2, \kappa_1)^\top}{\sqrt{\kappa_1^2 + \kappa_2^2}}. \tag{29}$$

Changing the overall sign gives the same basis convention up to the sign of its coefficient.

The normalization follows from integrating the squared sine and cosine factors, each of which contributes half the corresponding side length.

A.3 Enumeration and truncation

For the periodic basis, nonzero integer wave vectors in the canonical half-space are sorted by $\lambda_{\boldsymbol{k}}$, with a deterministic integer lexicographic rule breaking ties. Each wave vector contributes $d - 1$ polarizations, and its cosine and sine phases are consecutive in the real representation. For the free-slip basis, positive index pairs are likewise sorted by $\lambda_{\boldsymbol{n}}$, with one mode per pair and lexicographic tie-breaking.

A requested state dimension may cut an eigenvalue-degenerate shell to obtain exactly M coordinates. The mode indices, phase order, and eigenvalues are stored with each dataset so that reconstruction uses the same basis as data generation.

A.4 Velocity and vorticity reconstruction

Velocity reconstruction is the direct modal sum

$$\boldsymbol{u}_M(\boldsymbol{x}, t) = \sum_{m=1}^{M} w_m(t) \boldsymbol{\psi}_m(\boldsymbol{x}). \tag{30}$$

For periodic modes, spatial derivatives multiply a complex Fourier coefficient by $\mathrm{i}\boldsymbol{\kappa}$, or equivalently exchange the real sine and cosine phases with the appropriate sign. For free-slip modes, derivatives are taken analytically from the separated sine–cosine products. In two dimensions the scalar vorticity is $\omega = \partial_{x_1} u_2 - \partial_{x_2} u_1$. These analytic derivatives are used both for data generation and for grid-based evaluation, avoiding finite-difference error in the reported vorticity fields.

B Derivations and Structural Properties

B.1 Galerkin projection in a divergence-free basis

Let $V_M = \operatorname{span}\{\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_M\}$ be either the periodic or free-slip space constructed in Appendix A, and let $\boldsymbol{u}_M = \sum_j w_j \boldsymbol{\psi}_j$. Testing the incompressible Navier–Stokes equations against $\boldsymbol{\psi}_i$ gives

$$\langle \boldsymbol{\psi}_i, \partial_t \boldsymbol{u}_M \rangle + \langle \boldsymbol{\psi}_i, (\boldsymbol{u}_M \cdot \boldsymbol{\nabla}) \boldsymbol{u}_M \rangle = -\langle \boldsymbol{\psi}_i, \boldsymbol{\nabla} p \rangle + \nu \langle \boldsymbol{\psi}_i, \Delta \boldsymbol{u}_M \rangle + \langle \boldsymbol{\psi}_i, \boldsymbol{f} \rangle. \tag{31}$$

The pressure term vanishes because

$$\langle \boldsymbol{\psi}_i, \boldsymbol{\nabla} p \rangle = -\langle \boldsymbol{\nabla} \cdot \boldsymbol{\psi}_i, p \rangle + \int_{\partial\Omega} p \, \boldsymbol{\psi}_i \cdot \boldsymbol{n} \, \mathrm{d}S = 0. \tag{32}$$

For periodic boundaries the surface contributions cancel pairwise; for free-slip boundaries the normal component of every basis field vanishes. Orthonormality and $\Delta\boldsymbol{\psi}_i = -\lambda_i\boldsymbol{\psi}_i$ then yield

$$\dot{w}_i = -\nu\lambda_i w_i + \sum_{j,k=1}^{M} b_{ijk} w_j w_k + g_i, \quad b_{ijk} = -\langle \boldsymbol{\psi}_i, (\boldsymbol{\psi}_j \cdot \boldsymbol{\nabla})\boldsymbol{\psi}_k \rangle, \quad g_i = \langle \boldsymbol{\psi}_i, \boldsymbol{f} \rangle. \tag{33}$$

B.2 Skew-symmetric structure of incompressible advection

For any admissible divergence-free field $\boldsymbol{v}$ and fields $\boldsymbol{u}, \boldsymbol{z}$ obeying the same boundary convention, integration by parts gives

$$\begin{aligned} \langle \boldsymbol{z}, (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{u} \rangle = -\langle \boldsymbol{u}, (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{z} \rangle - \int_\Omega (\boldsymbol{\nabla} \cdot \boldsymbol{v})(\boldsymbol{u}^\top \boldsymbol{z}) \, \mathrm{d}\boldsymbol{x} \\ + \int_{\partial\Omega} (\boldsymbol{v} \cdot \boldsymbol{n})(\boldsymbol{u}^\top \boldsymbol{z}) \, \mathrm{d}S = -\langle \boldsymbol{u}, (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{z} \rangle. \end{aligned} \tag{34}$$

The last equality again follows from periodic cancellation or zero normal velocity. Substituting basis functions into Eq. (34) shows

$$b_{ijk} = -b_{kji}. \tag{35}$$

Consequently the inviscid, unforced Galerkin system conserves energy:

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{1}{2} \|\boldsymbol{w}\|_2^2 = \sum_{i,j,k} b_{ijk} w_i w_j w_k = 0, \tag{36}$$

because the product $w_i w_k$ is symmetric while b_{ijk} is antisymmetric under $i \leftrightarrow k$.

B.3 Low-rank skew-symmetric bilinear parameterization

The learned bilinear map $\mathcal{Q}_\theta$ can be expanded as

$$[\mathcal{Q}_\theta(\boldsymbol{u}, \boldsymbol{v})]_i = \sum_{j,k=1}^{M} Q_{ijk} u_j v_k, \qquad Q_{ijk} = \sum_{\alpha=1}^{r} (A_{i\alpha} B_{j\alpha} - B_{i\alpha} A_{j\alpha}) \, C_{k\alpha}. \tag{37}$$

The construction makes $Q_{ijk} = -Q_{jik}$. This convention places the transported argument first and the advecting argument second, reversing the input order of b_{ijk} in Eq. (33). In this ordering, the physical interaction tensor is b_{ikj}, which is antisymmetric in i, j and gives the same quadratic dynamics when both inputs equal $\boldsymbol{w}$. Therefore, for arbitrary $\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{v}$,

$$\langle \boldsymbol{z}, \mathcal{Q}_\theta(\boldsymbol{u}, \boldsymbol{v}) \rangle = -\langle \boldsymbol{u}, \mathcal{Q}_\theta(\boldsymbol{z}, \boldsymbol{v}) \rangle. \tag{38}$$

Taking all three arguments equal to $\boldsymbol{w}$ proves $\boldsymbol{w}^\top f_\theta(\boldsymbol{w}) = 0$ for $f_\theta(\boldsymbol{w}) = \mathcal{Q}_\theta(\boldsymbol{w}, \boldsymbol{w})$. Thus every continuous trajectory of the learned inviscid dynamics satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{1}{2} \|\boldsymbol{w}(t)\|_2^2 = 0 \tag{39}$$

without a penalty term or constrained optimization.

A dense quadratic vector field contains M^3 scalar entries before symmetry reductions and costs $\mathcal{O}(M^3)$ operations under direct contraction. The factorized model has $3Mr$ bias-free parameters. It evaluates matrix–vector products together with element-wise products and therefore costs $\mathcal{O}(Mr)$ arithmetic and storage per right-hand-side call. An RK4 step uses four such calls.

B.4 Expansion of the finite-time map

Although $f(\boldsymbol{w}) = \mathcal{Q}(\boldsymbol{w}, \boldsymbol{w})$ is quadratic, the exact flow is not a quadratic function of its initial condition. Its short-time expansion is

$$\begin{aligned} \Phi_h(\boldsymbol{w}) &= \boldsymbol{w} + h f(\boldsymbol{w}) + \frac{h^2}{2} Df(\boldsymbol{w}) f(\boldsymbol{w}) + \mathcal{O}(h^3) \\ &= \boldsymbol{w} + h\mathcal{Q}(\boldsymbol{w}, \boldsymbol{w}) + \frac{h^2}{2} \left[\mathcal{Q}(\mathcal{Q}(\boldsymbol{w}, \boldsymbol{w}), \boldsymbol{w}) + \mathcal{Q}(\boldsymbol{w}, \mathcal{Q}(\boldsymbol{w}, \boldsymbol{w})) \right] + \mathcal{O}(h^3). \end{aligned} \tag{40}$$

The second-order-in-time term is cubic in $\boldsymbol{w}$, and successive terms contain progressively higher polynomial degrees. Intermediate RK4 stages re-evaluate the same bilinear operator at evolved states, approximating these compositions without introducing any separate network.

For a sufficiently smooth vector field, classical RK4 has local state error $\mathcal{O}(h^5)$ and global error $\mathcal{O}(h^4)$. The learned skew-symmetric bilinear dynamics conserve energy exactly in continuous time, but classical explicit RK4 is not an energy-preserving integrator. Its one-step energy defect is therefore generally nonzero, although it is $\mathcal{O}(h^5)$ locally along a smooth bounded trajectory. Using K substeps replaces $h = \Delta t$ by $h = \Delta t/K$ and can reduce this numerical defect.

B.5 Exponential viscous splitting

The basis diagonalizes viscosity. Given the neural advection prediction $\widehat{\boldsymbol{w}}_{n+1}$, the state at the next time step is obtained by the exact viscous subflow

$$\widetilde{\boldsymbol{w}}_{n+1} = \exp(-\nu \boldsymbol{\Lambda} \Delta t)\widehat{\boldsymbol{w}}_{n+1}, \qquad [\widetilde{\boldsymbol{w}}_{n+1}]_j = e^{-\nu\lambda_j \Delta t}[\widehat{\boldsymbol{w}}_{n+1}]_j. \tag{41}$$

Since $0 < e^{-\nu\lambda_j \Delta t} \leq 1$, it is non-expansive in coefficient energy:

$$\|\widetilde{\boldsymbol{w}}_{n+1}\|_2^2 = \sum_j e^{-2\nu\lambda_j \Delta t}|\widehat{w}_{n+1,j}|^2 \leq \|\widehat{\boldsymbol{w}}_{n+1}\|_2^2. \tag{42}$$

The same formula applies to periodic cosine–sine pairs and free-slip modes, because in both cases λ_j is the corresponding Stokes eigenvalue.

C Dataset Generation

The two-dimensional datasets are generated as trajectories with aligned advection input–target pairs. This samples the evolving state distribution encountered during autoregressive rollout. Both boundary settings use unforced flow and the advection–viscosity splitting described below.

C.1 Periodic pseudospectral generator

The periodic generator stores one complex amplitude for each wave–polarization pair in the canonical Fourier half-space of Appendix A.1. At every right-hand-side evaluation it inserts each retained amplitude and its complex conjugate into the full Fourier array, reconstructs a real velocity field with an inverse FFT, and evaluates the rotational form of incompressible advection,

$$\partial_t \boldsymbol{u} = \mathbb{P}(\boldsymbol{u} \times \boldsymbol{\omega}), \qquad \boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}. \tag{43}$$

In two dimensions, scalar vorticity is embedded in the third direction, so $\boldsymbol{u} \times \boldsymbol{\omega} = (u_2\omega, -u_1\omega)$.

After transforming the nonlinear field back to Fourier space, every nonzero wave is projected with the Leray projector

$$\boldsymbol{P_k} = \boldsymbol{I} - \frac{\boldsymbol{\kappa_k}\boldsymbol{\kappa_k}^\top}{|\boldsymbol{\kappa_k}|^2}. \tag{44}$$

The modal time derivative is obtained by contracting the projected spectrum with the stored transverse polarizations. The FFT grid is endpoint-free. If K_i is the largest retained absolute integer wave number along axis i, the implementation uses at least $3K_i + 1$ points along that axis, together with a configured minimum. This realizes the usual three-halves padding requirement for the quadratic product when expressed relative to the unpadded two-sided spectral support.

The periodic reference solver uses adaptive DOP853 in float64. The FFT resolution and generation seed are given in Table 4.

C.2 Free-slip quadrature generator

The free-slip generator works directly with the modes given in Appendix A.2. For each coefficient state, it reconstructs velocity and analytic vorticity on a tensor-product grid, evaluates the same

rotational nonlinear field as in Eq. (43), and projects every vector component back onto its sine–cosine pattern. Projection uses a tensor-product trapezoidal quadrature including both domain endpoints. If $N_i^{\max}$ is the largest retained modal order on axis i, the nominal quadrature resolution is

$$Q_i = \max\{qN_i^{\max} + 1, Q_{\min}\}, \tag{45}$$

where the quadrature multiplier q and lower bound $Q_{\min}$ are recorded in the dataset configuration. The sine and cosine tables and their quadrature-weighted counterparts are precomputed. Reconstruction and projection then reduce to tensor contractions, with no numerical eigensolve or finite-difference derivative.

As in the periodic case, the projected system evolves only the first M resolved modes. The reference solver integrates this resolved Galerkin system and always truncate a larger state. The free-slip grid and reference integrator are given in Table 4.

C.3 Rollout construction

Let R be the number of independently initialized rollouts and S the number of adjacent transitions per rollout. Initial coefficients are sampled only in the first K_{init} eigenvalue-ordered modes; all remaining resolved coefficients start at zero. In the periodic complex representation, real and imaginary parts are sampled independently and scaled so the complex amplitude has the prescribed variance. In the free-slip representation, real coefficients are sampled directly. The distributions and scales used by the final datasets are reported in Table 4.

For every outer step, the generator first advances the inviscid resolved system over Δt, producing the advection target $\boldsymbol{y}_{n+1}$:

$$\boldsymbol{y}_{n+1} = \mathcal{Q}_{\text{ref}}^{\Delta t}(\boldsymbol{w}_n). \tag{46}$$

It stores $(\boldsymbol{w}_n, \boldsymbol{y}_{n+1})$ as the supervised advection pair, then applies the exact modal decay

$$\boldsymbol{w}_{n+1} = \exp(-\nu \boldsymbol{\Lambda}\, \Delta t)\boldsymbol{y}_{n+1} \tag{47}$$

before beginning the next step. Thus the target of row n is the pre-decay state, whereas the input of the next row is the post-decay state. This distinction ensures that the learned operator represents advection and that viscosity is applied exactly once during an autoregressive rollout.

The complete procedure is summarized below.

1. Generate and order the resolved periodic or free-slip basis.
2. Draw one low-frequency initial state using a seeded random-number generator.
3. For $n = 0, \ldots, S-1$, integrate the resolved inviscid system, store the advection pair, and exponentially damp the target to form the next rollout state.
4. Repeat for R independent initial states, producing RS aligned pairs in trajectory-major order.
5. Reserve complete trajectories for training, validation, and testing; never split individual rows from the same trajectory.

C.4 Dataset settings

Table 4 summarizes the dataset configurations. For periodic flow, both the reference solver and the learned model retain the same 2048 complex amplitudes, equivalent to $M = 4096$ real coordinates. No additional modes are evolved by the reference solver, so the learning target does not include the effects of omitted modes from a larger reference system. The free-slip dataset likewise uses $M = 4096$ real coefficients for both reference evolution and learning.

Each dataset records its basis indices, eigenvalues, coefficient conventions, trajectory boundaries, and generation settings. Basis reconstruction and sampled reference transitions are checked before training, and complete trajectories are assigned to disjoint training, validation, and test sets.

Table 4: **Data-generation settings.** The mode count M refers to real coordinates.

Setting	Periodic	Free-slip
Domain lengths	$(1, 1)$	$(1, 1)$
Resolved modes M	4096	4096
Initial active modes K_{init}	64	64
Rollouts R	8400	8334
Pairs per rollout S	120	120
Total pairs RS	1,008,000	1,000,080
Outer step Δt	0.02	0.02
Viscosity ν	3×10^{-4}	3×10^{-4}
Initial distribution	$\mathcal{N}(0, 0.05^2)$	$\mathcal{N}(0, 0.05^2)$
Spatial grid	154^2 FFT	433^2 quadrature
Reference integrator	adaptive DOP853	adaptive DOP853
Backend and precision	PyTorch/CUDA, float64	PyTorch/CUDA, float64
Random seed	1234	1234

D Training and Implementation Details

D.1 Normalization and training objective

Both boundary settings use adjacent advection pairs $(\boldsymbol{w}_n, \boldsymbol{y}_{n+1})$ constructed as in Appendix C.3, where $\boldsymbol{y}_{n+1}$ is the reference advection output before viscous decay. Define the physical coefficient increment

$$\Delta \boldsymbol{w}_n = \boldsymbol{y}_{n+1} - \boldsymbol{w}_n. \tag{48}$$

Only the training split is used to compute the global root-mean-square scales

$$s_x = \left(\frac{1}{N_{\text{tr}} M} \sum_{n,j} w_{n,j}^2 \right)^{1/2}, \qquad s_\Delta = \left(\frac{1}{N_{\text{tr}} M} \sum_{n,j} (\Delta w_{n,j})^2 \right)^{1/2}. \tag{49}$$

The model receives $\boldsymbol{x}_n = \boldsymbol{w}_n / s_x$. Its differentiable RK4 map acts in this normalized coordinate system, and the physical predicted increment is

$$\widehat{\Delta \boldsymbol{w}}_n = s_x \left(\mathcal{N}_\theta^{\Delta t}(\boldsymbol{x}_n) - \boldsymbol{x}_n \right). \tag{50}$$

Here $\mathcal{N}_\theta^{\Delta t}$ is the RK4 map from Sec. 3.2, applied in normalized coordinates. The physical advection map is therefore $\boldsymbol{w} \mapsto s_x \mathcal{N}_\theta^{\Delta t}(\boldsymbol{w}/s_x)$; the main-text equations omit this scalar normalization. Equivalently, the implementation returns an increment scaled by s_Δ and stores the ratio s_x/s_Δ in the model checkpoint.

Training minimizes a mode-weighted, per-sample relative squared error,

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \frac{\sum_{j=1}^{M} q_j (\widehat{\Delta w}_{n,j} - \Delta w_{n,j})^2}{\sum_{j=1}^{M} q_j (\Delta w_{n,j})^2 + \varepsilon}. \tag{51}$$

Here N is the number of training pairs in the loss average, and $\varepsilon > 0$ stabilizes the denominator. The target increment includes only advection; viscous decay is applied analytically during rollout. For the continuously unit-normalized periodic real Fourier basis, all modal weights are equal. For the free-slip basis, q_j is the diagonal Gram entry of mode j under the endpoint trapezoidal grid used for evaluation, normalized so that the mean weight is one. This corrects small differences between the continuous modal normalization and the discrete grid norm.

D.2 Coefficient representation and reconstruction

The low-rank skew-symmetric bilinear operator acts on real modal coefficients in both boundary settings. Periodic complex amplitudes are converted to consecutive cosine and sine amplitudes using Eq. (25), with the eigenvalue repeated for both phases. Free-slip coefficients are already real. The

Table 5: **NeNS^2 training settings.** Two rollout substeps are utilized in timing comparison.

Setting	Periodic	Free-slip
Modes M	4096	4096
Interaction rank r	8192	8192
Train/val./test trajectories	8000/200/200	7934/200/200
Training RK4 substeps	1	1
Rollout RK4 substeps	1 or 2	1 or 2
Per-device batch	2048	2048
Devices/global batch	4/8192	4/8192
Optimizer steps	30000	30000
Peak/minimum learning rate	$3 \times 10^{-4}/3 \times 10^{-5}$	$3 \times 10^{-4}/3 \times 10^{-5}$
Warmup steps	500	500
Weight decay	10^{-6}	10^{-6}
Gradient clip	1.0	1.0
Evaluation/checkpoint interval	500/5000	500/5000
Precision and hardware	FP32, 4×A100	FP32, 4×A100
Seed	42	42

basis ordering and eigenvalues are stored with the checkpoint to preserve the coordinate convention used during training.

Both settings use the same normalization, learned bilinear operator, RK4 integration, and exponential viscous decay. Velocity and vorticity are reconstructed on an endpoint-free periodic grid or an endpoint grid for free-slip flow, using the analytic derivatives in Appendix A.4.

D.3 Optimization

The three factor matrices are bias-free. The $\boldsymbol{C}$ factor is initialized to zero, so the initial right-hand side and one-step increment vanish, while $\boldsymbol{A}$ and $\boldsymbol{B}$ retain their standard random linear-layer initialization. Optimization uses AdamW, gradient-norm clipping, and a linear warmup followed by cosine decay,

$$\eta(s) = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left[1 + \cos\left(\pi \frac{s - s_{\text{warm}}}{s_{\max} - s_{\text{warm}}}\right)\right] \tag{52}$$

after the warmup interval. Training uses one differentiable RK4 substep at the dataset interval. At inference, the same learned operator can be integrated with a different number of substeps; the reported rollouts use one or two.

D.4 Autoregressive inference

Let $\widetilde{\boldsymbol{w}}_0 = \boldsymbol{w}_0$ be the reference initial state. For K RK4 substeps over each prediction interval, define

$$\mathcal{N}_{\theta,K}^{\Delta t} = \underbrace{\mathcal{N}_\theta^{\Delta t/K} \circ \cdots \circ \mathcal{N}_\theta^{\Delta t/K}}_{K \text{ substeps}}. \tag{53}$$

Thus $\mathcal{N}_{\theta,1}^{\Delta t} = \mathcal{N}_\theta^{\Delta t}$ is the training map. Inference repeats

$$\begin{aligned} \boldsymbol{x}_n &= \widetilde{\boldsymbol{w}}_n / s_x, \\ \widehat{\boldsymbol{w}}_{n+1} &= s_x \mathcal{N}_{\theta,K}^{\Delta t}(\boldsymbol{x}_n), \\ \widetilde{\boldsymbol{w}}_{n+1} &= \exp(-\nu \boldsymbol{\Lambda} \Delta t)\, \widehat{\boldsymbol{w}}_{n+1}. \end{aligned} \tag{54}$$

The reported rollouts use $K = 1$ or $K = 2$, with $K = 2$ in the timing comparison (Appendix E.6). Only the initial frame is taken from the reference trajectory. All later inputs are the model's own post-viscosity predictions. No coefficient projection, clipping, spectral filtering, energy correction, or ground-truth replacement is applied during the ID and vortex-OOD benchmark rollouts. The exploratory cylinder scenes in Sec. 4.4 additionally apply the boundary constraint described there.

E Evaluation Protocol and Reproducibility

E.1 Trajectory splits and model selection

Training, validation, and test sets contain disjoint complete trajectories, with the counts listed in Table 5. Normalization and optimizer updates use only training data. Hyperparameters and checkpoints are selected on validation trajectories, and test trajectories are evaluated after these choices are fixed. All methods use the same basis ordering and field reconstruction within each boundary setting.

E.2 Out-of-distribution vortex configurations

All vortex-OOD reference trajectories are generated independently of the learned models. Within each case, every model starts from the same projected initial field and evolves autoregressively.

Periodic Taylor vortices. The periodic out-of-distribution experiment uses a Taylor-vortex configuration with core radius 0.3, velocity scale $U = 1$, center separations 0.785 and 0.790 for the merging and separating cases, respectively, zero viscosity, reference time step 0.02, final time 4.8, and a 154×154 endpoint-free reference grid.

Free-slip leapfrogging vortices. The free-slip out-of-distribution benchmark uses vortex centers $(-0.253, 0.3)$, $(-0.1, 0.3)$, $(0.1, 0.3)$, and $(0.253, 0.3)$, signs $(+, +, -, -)$, core size 0.03, intensity 46, a 433×433 projection grid, viscosity 3×10^{-4}, time step 0.02, and a rollout horizon of 120 steps. The reference trajectory is generated independently of every learned model, then all models start from the same projected initial frame.

E.3 Controlled learned-dynamics baselines

The controlled neural comparison is performed separately on the free-slip and periodic two-dimensional flow benchmarks. Within each boundary setting, NeNS^2 and all baselines receive states derived from the same coefficient trajectories, use the same trajectory split, compute normalization from training data only, and are evaluated from identical held-out initial states. Grid models receive scalar vorticity reconstructed on the common evaluation grid: an endpoint grid for free slip and an endpoint-free Fourier grid for periodic flow. Their raw predictions are fed back autoregressively, without projection onto the divergence-free basis, clipping, filtering, pressure correction, or energy correction. The free-slip split is $7934/200/200$ trajectories and the periodic split is $8000/200/200$; both use $M = 4096$, $\Delta t = 0.02$, and a 120-step ID horizon.

The comparison contains the following model families.

- **U-Net**: a multiscale convolutional encoder–decoder following the PDE forecasting adaptation of the standard U-Net architecture.
- **CNO**: the official continuous-discrete consistent convolutional operator, used through a data and training adapter without changing its internal filtered-convolution blocks.
- **FNO**: the official Fourier Neural Operator implementation on the regular vorticity grid.
- **PDE-Transformer**: the official multiscale spatiotemporal Transformer. Scratch-trained, pretrained, and zero-shot variants are labeled separately in Table 1.
- **LNO**: the official latent neural operator with its physics-cross-attention encoder and coordinate decoder.

The primary controlled protocol uses one physical input frame and one-step autoregressive prediction. U-Net, CNO, and the controlled FNO predict a vorticity increment. LNO predicts the next vorticity state because short validation diagnostics showed that its delta formulation could collapse to persistence. The parameter-matched PDE-Transformer predicts a vorticity increment; its configuration is listed in Table 6. Native configurations are labeled separately in Table 1. Different normalized objectives are not compared through their raw training- loss magnitudes.

Two resource controls are reported separately. A parameter-matched comparison adjusts architecture width toward a common trainable-parameter budget. A compute-matched comparison gives each

model the same aggregate GPU-hour budget as NeNS^2. For every entry we report actual parameter count, optimizer updates, target-frame exposures, number and type of accelerators, precision, aggregate GPU-hours, peak allocated memory, and measured inference cost.

Table 6: **Controlled baseline configurations.** The listed free-slip capacities and compute budgets are mirrored by the periodic comparison; boundary-specific grids and target adapters follow the protocol above.

Model	Input/target	Capacity	Training configuration
NeNS^2	Coefficients $\rightarrow$ coefficient delta	$M = 4096$, $r = 8192$ Params: 100.66M	Batch: $2048 \times 4 = 8192$ Steps: 30,000 LR: $3 \times 10^{-4} \rightarrow 3 \times 10^{-5}$ Budget: 15.111 GPU-h
U-Net	One vorticity frame $\rightarrow$ vorticity delta	Widths: $115/230$ $460/920$ Bottleneck: 1840 Params: 100.19M	Batch: $128 \times 4 = 512$ Steps: GPU-hour capped LR: $3 \times 10^{-4} \rightarrow 3 \times 10^{-5}$ Budget: 15.111 GPU-h
CNO	One vorticity frame $\rightarrow$ vorticity delta	Levels: 3; residual blocks: $4/4/4$ Neck blocks: 6; widths: $188/184$ Params: 100.66M	Batch: $128 \times 4 = 512$ Steps: GPU-hour capped LR: $3 \times 10^{-4} \rightarrow 3 \times 10^{-5}$ Budget: 15.113 GPU-h
FNO	One vorticity frame $\rightarrow$ vorticity delta	Fourier modes: 48×48 Width/layers: $103/4$ Params: 101.91M	Batch: $128 \times 4 = 512$ Steps: GPU-hour capped LR: $3 \times 10^{-4} \rightarrow 3 \times 10^{-5}$ Budget: 15.111 GPU-h
PDE-Transformer	One frame as a two-time masked input $\rightarrow$ vorticity delta	SC-100 (scratch) Width/heads: $144/6$ Depth: $2/5/8/5/2$ Params: 102.03M	Batch: $4 \times 4 = 16$ Steps: GPU-hour capped LR: $10^{-4} \rightarrow 10^{-5}$ Budget: 15.111 GPU-h
LNO	One vorticity frame $\rightarrow$ next vorticity state	LNO-100M Tokens/width: $256/960$ Blocks/heads: $12/30$ Params: 99.92M	Batch: $4 \times 4 = 16$ LR: $3 \times 10^{-4} \rightarrow 3 \times 10^{-5}$ Decay: 3000 steps, then hold Clip: none; budget: 15.111 GPU-h

Learning-rate trials use the validation split only, under the same bounded search budget for each candidate within a model family. The selected value and selection metric are archived before a formal run begins. Best checkpoints are chosen on validation performance; latest checkpoints retain optimizer state. Test data are not read by the learning-rate sweep or checkpoint selection code.

E.4 METRICS

Let N_{test} be the number of held-out trajectories and T the number of predicted frames. For NeNS^2, $\widetilde{\boldsymbol{w}}$ denotes the rollout state after analytic viscous decay, and $\widetilde{\omega}$ is its reconstructed vorticity. We use the same notation for the final rollout fields of the baselines. The coefficient relative error is

$$e_w(t) = \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \frac{\|\widetilde{\boldsymbol{w}}_t^{(i)} - \boldsymbol{w}_t^{(i)}\|_2}{\|\boldsymbol{w}_t^{(i)}\|_2}. \tag{55}$$

The vorticity relative error $e_\omega(t)$ is defined analogously on the common evaluation grid. The headline rollout score gives equal weight to trajectories and time,

$$\overline{e}_\omega = \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \frac{1}{T} \sum_{t=1}^{T} \frac{\|\widetilde{\omega}_t^{(i)} - \omega_t^{(i)}\|_2}{\|\omega_t^{(i)}\|_2}. \tag{56}$$

We also report velocity relative error where a grid baseline permits a consistent velocity reconstruction.

For an orthonormal velocity basis, kinetic energy and enstrophy are

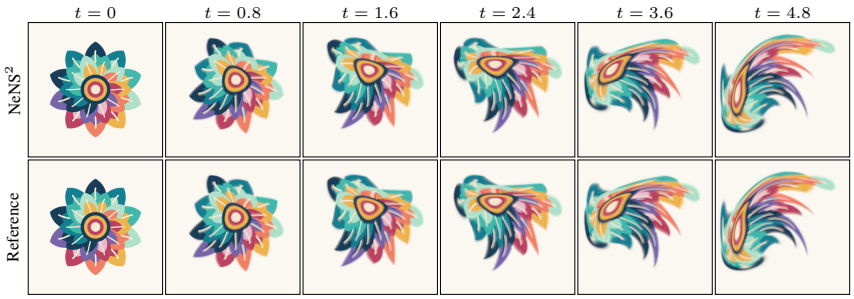
$$E(\boldsymbol{w}) = \frac{1}{2} \sum_j w_j^2, \qquad Z(\boldsymbol{w}) = \frac{1}{2} \sum_j \lambda_j w_j^2. \tag{57}$$

We report relative energy and enstrophy errors over time, together with the energy drift of inviscid tests when applicable. Frequency-resolved errors are computed by sorting the modes by increasing eigenvalue and dividing the resulting ordering into three equal-cardinality low-, middle-, and high-frequency bands.

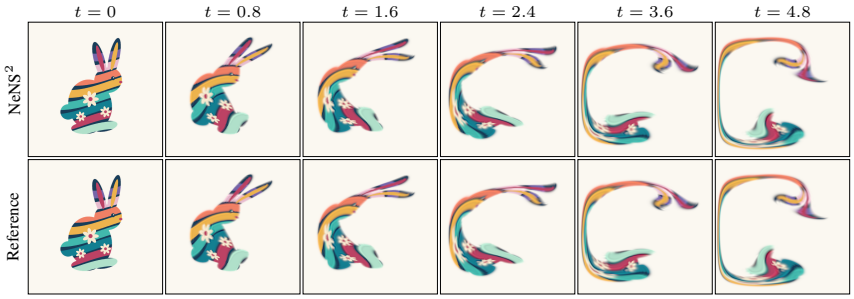
A trajectory that first produces a NaN or infinite state is marked unstable at that frame. We report its failure time and do not hide the failure by averaging only its finite prefix. Aggregate tables include the mean, median, and 95% trajectory-bootstrap confidence intervals computed from 10000 resamples with random seed 42.

E.5 Long-term generalization visualizations

Fig. 5 shows the matched-time passive-dye rollouts discussed in Sec. 4.4. The frozen model and the numerical reference start from the same modal state, but each advects the dye using its own predicted velocity field. The two examples use a layered rosette in a rotating flow and a patterned bunny in a shearing flow.



(a) Layered rosette;



(b) Patterned bunny.

Figure 5: **Long-term transport.** We demonstrate long-term passive-dye transport on unseen flow configurations. For each example, the upper row shows the rollout produced by NeNS^2, while the lower row shows the corresponding evolution from the traditional spectral solver. The six columns report matched times $t = 0, 0.8, 1.6, 2.4, 3.6, 4.8$, enabling direct comparison of the accumulated material deformation over the full rollout. Panels (a) and (b) correspond to the rosette and bunny examples, respectively. The same dye initialization and color mapping are used for both methods, and the dye is advected independently by each predicted velocity field. The close agreement throughout the rollout indicates that NeNS^2 preserves accurate long-term flow evolution and material transport even on flow configurations not encountered during training.

E.6 Timing protocol

Sec. 4.3 compares the latency of one prediction interval at batch sizes $B = 1$ and $B = 32$ on a single NVIDIA A100-SXM4-80GB. Both use the same 32 equidistantly selected test trajectories per boundary setting, with $M = 4096$, $\Delta t = 0.02$, and $\nu = 3 \times 10^{-4}$. Computation uses FP32 with TF32 and mixed precision disabled. Methods run sequentially in separate processes on the same GPU without competing workloads.

After 20 warmup steps at the requested batch size, we time ten independent one-step trials per initial state at $B = 1$ (320 samples), and ten trials on the whole set at $B = 32$ (10 samples). Each trial

starts from the same prescribed initial state or batch. Wall-clock timing is bracketed by CUDA synchronization, and the table reports the pooled median.

NeNS^2 uses two RK4 substeps at both batch sizes, including all eight evaluations of the learned bilinear operator per prediction interval. Normalization and viscosity updates are included. The classical solver uses RK4 with a CFL substep rule, whose cost is included. At $B = 32$, the maximum speed over all states determines a common substep count. These timings compare the stated integration settings without a search for equal prediction error. The batch controller also affects the classical solver's scaling with batch size.

Steady-state timing excludes checkpoint loading, solver construction, input transfer, I/O, error evaluation, and visualization. Peak memory includes PyTorch-allocated resident parameters or solver tensors and workspace, but excludes stored rollout frames and the CUDA context. Training cost is not included.

Periodic grid resolution. The classical periodic solver evaluates advection by FFT reconstruction, pointwise multiplication, and projection onto the retained modes, as in Appendix C.1. It uses CFL number 0.5 and periodic spacing L_i/N_i. For the main timing comparison, its grid is 433^2, matching the point count of the free-slip quadrature grid. Periodic nodes remain endpoint-free, whereas free-slip spacing is $L_i/(N_i - 1)$.

The 154^2 grid used for periodic data generation already resolves the retained quadratic interactions without aliasing. Increasing it to 433^2 does not add retained modes, but increases FFT cost and can increase the CFL substep count. For context, the classical solver on the original 154^2 grid takes 16.247 ms at $B = 1$ and 21.987 ms at $B = 32$. The main table therefore compares equal grid point counts, rather than the fastest grid choice for each solver. Both the reference dataset and the timed periodic methods evolve 2048 complex amplitudes, equivalent to 4096 real coordinates.

Rollout accuracy and memory. The timing configurations are also evaluated over 120-step rollouts on the same 32 trajectories at $B = 32$. Table 7 reports the deviation in total vorticity from its initial value, averaged equally over trajectories and predicted frames, together with peak allocated memory. This metric measures violation of vorticity conservation over the rollout. All four configurations remain finite on all 32 trajectories.

Table 7: **Rollout statistics.** We report rollout accuracy and peak allocated memory for the $B = 32$ timing configurations. Note that two-dimensional incompressible inviscid flow should conserve vorticity.

Setting	Method	Mean vorticity rel. L^2	Memory (MiB)
Free-slip	NeNS^2	2.777×10^{-2}	401.66
	Classical	5.924×10^{-6}	174.50
Periodic	NeNS^2	1.156×10^{-1}	401.66
	Classical	3.719×10^{-2}	372.59

E.7 Software and reproducibility

The experiments use Python 3.11, PyTorch 2.4.1, NumPy 1.26.4, and CUDA 12.4; training uses four NVIDIA A100 GPUs. Run records include dataset and basis metadata, trajectory splits, normalization scales, model and optimizer settings, random seeds, checkpoint-selection rules, source revisions, and hardware details. Per-trajectory predictions, failure flags, aggregate metrics, and timing samples are retained to support reproduction of the reported results.