Learning Autoregressive Neural Operators on Solver-Defined Physical Constraint Manifolds

Nannan Zhou
School of Computer Science
Sichuan University, Chengdu, China
National Key Laboratory of Fundamental Algorithms and Models for Engineering Numerical Simulation
Chengdu, China
zhounannan@stu.scu.edu.cn

Guanmeng Xian
School of Computer Science
Sichuan University, Chengdu, China
National Key Laboratory of Fundamental Algorithms and Models for Engineering Numerical Simulation
Chengdu, China
xianguanmeng@stu.scu.edu.cn

Ning Yang*
School of Computer Science
Sichuan University, Chengdu, China
National Key Laboratory of Fundamental Algorithms and Models for Engineering Numerical Simulation
Chengdu, China
yangning@scu.edu.cn

arXiv:submit/8375754 [cs.AI] 15 Nov 2026

Abstract

Autoregressive neural operators approximate time-dependent PDE solvers by repeatedly composing a learned transition, although the resulting states need not remain feasible under the discrete constraints of the numerical problem. We study this issue for incompressible flow, where each velocity state must satisfy solver-level incompressibility together with prescribed boundary values. We formulate autoregressive operator learning as a map between solver-defined affine feasible sets and introduce PCM-NO, a projected-residual realization of this feasible-transition principle. The neural backbone predicts a candidate state increment, a fixed tangent projection extracts its admissible component, and a metric retraction maps the advanced state to the prescribed feasible set. With exact affine constraint solves, the resulting transition is closed on the feasible set, while an affine identity separates admissible-update parameterization from state-level closure. With approximate solves, tangent-only residuals admit a cumulative upper bound, whereas per-step retraction resets feasibility to the residual achieved by the current solve. Experiments on periodic Navier–Stokes flow and lid-driven cavity flow demonstrate the superiority of PCM-NO in preserving solver-level constraints over long autoregressive rollouts and under parameter, boundary-value, and resolution shifts, while achieving lower rollout errors and better flow-specific diagnostics across both in-distribution and stress settings.

CCS Concepts

• **Computing methodologies** → **Machine learning**; • **Applied computing** → **Physics**.

*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, Washington, DC, USA
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Keywords

neural operators, scientific machine learning, computational fluid dynamics, incompressible flow, physical constraints, autoregressive PDE surrogates

ACM Reference Format:
Nannan Zhou, Guanmeng Xian, and Ning Yang. 2026. Learning Autoregressive Neural Operators on Solver-Defined Physical Constraint Manifolds. In . ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Neural operators provide reusable function-to-function surrogates for parametric PDE families and have become an effective approach to accelerating repeated scientific simulation [18, 22, 28]. For time-dependent systems, an autoregressive neural operator learns a short-time transition and obtains long-horizon predictions by repeatedly composing that transition [2, 24, 29]. This formulation mirrors the stepwise structure of numerical solvers and supports flexible prediction horizons across varying physical conditions. However, it also makes every predicted state part of the input encountered by subsequent transitions. Constraint violations that survive one step can therefore enter the remainder of the rollout and progressively affect its physical validity and predictive stability.

This issue is especially acute in incompressible flow, where preservation of local fluid volume requires a divergence-free velocity field, and a fixed discretization encodes this requirement together with prescribed boundary values as algebraic constraints on admissible velocity states [12]. Yet solver-generated training trajectories alone do not guarantee that a learned transition preserves this admissible set from one step to the next [23, 29]. Existing physics-aware approaches address constraint violations through soft-constraint objectives based on residual or conservation penalties [17, 23, 32], structure-specific representations [4, 8, 34], or post-hoc state projection [20]. These approaches have distinct limitations in autoregressive rollout. **Soft-constraint objectives** do not guarantee feasibility at every step, so residual violations can enter subsequent predictions and accumulate over time. **Structure-specific representations** encode selected constraints but lack composability, requiring specialized designs for joint conditions such as incompressibility and nonhomogeneous boundary values. **Post-hoc state projection** like FinalProj [20] restores output feasibility only after an unconstrained next-state prediction, without constraining the learned update before state advancement.

These limitations motivate the feasible-transition view of autoregressive operator learning. Rather than allowing the learned dynamics to evolve freely in the ambient state space, *we treat the solver-defined feasible set as the state space of the learned transition, so that every autoregressive prediction remains physically feasible at the solver level.* This set captures discrete kinematic requirements such as incompressibility and prescribed boundary values, while the neural operator learns the dynamics within the feasible set. Based on this idea, we propose **Physical-Constraint-Manifold Neural Operators** (**PCM-NO**), a unified transition-level formulation in which the solver-defined feasible set itself serves as the state space of the learned dynamics. PCM-NO realizes this formulation by coupling a backbone neural operator with two solver-defined components. The **tangent projection** filters the backbone's candidate residual update before state advancement, so that only directions compatible with the homogeneous constraints enter the rollout. The **metric retraction** then maps the advanced state to the feasible set associated with the current constraint target, resetting numerical constraint residuals at each step. The **advantages** of PCM-NO are threefold. First, its projection–retraction factorization makes per-step solver-level feasibility a structural property of the transition, independent of loss balancing and of whether the backbone learns the constraints from data. Second, the projected-residual realization exposes the admissible component entering the learned dynamics and removes constraint-incompatible directions before state advancement, providing an interpretable inductive bias beyond state-level correction alone. Third, PCM-NO provides a unified transition formulation whose parameter-free, solver-derived layers accommodate periodic incompressibility, joint incompressibility–boundary constraints, and step-dependent affine targets.

Our analysis separates exact feasible-state closure, admissible-update parameterization, and finite-solve behavior. Under exact affine constraints, the direct-state and projected-residual realizations belong to the same feasible-transition family and map feasible inputs to the prescribed affine set. The tangent layer explicitly exposes the admissible component entering the learned dynamics and removes its orthogonal constraint-incompatible component before state advancement. Under approximate solves, tangent-only residuals can accumulate across rollout steps, whereas retraction resets the state residual to the accuracy achieved by the current solve and accommodates changes in the affine target. These results characterize the geometry and numerical behavior of the PCM-NO transition. The experiments then evaluate its predictive accuracy, rollout stability, and generalization under stress.

Our contributions can be summarized as follows:

- We formulate autoregressive operator learning as a feasible-state transition and propose PCM-NO, which couples a neural candidate-update backbone with successive tangent projection and metric retraction to keep every rollout step feasible under solver-defined affine constraints.
- We establish exact feasible-state closure, characterize the affine relationship between tangent projection and retraction, and analyze orthogonal update filtering together with residual accumulation and reset under approximate constraint solves.
- Extensive experiments on periodic and boundary-constrained incompressible flows verify the superiority of PCM-NO.

2 Preliminaries

2.1 Time-Evolution Operator and Discrete Realization

At each time t, the state $u_t := u(\cdot, t) \in \mathcal{X}$ is a spatial field function over the physical domain. The time-Δt evolution operator $\mathcal{S}_{\Delta t}$ maps the field function u_t to the one at the next time level:

$$\mathcal{S}_{\Delta t}(u_t; \mu, g) = u_{t+\Delta t}, \tag{1}$$

where μ denotes physical parameters and g boundary or forcing data. At spatial resolution h, these fields are represented by discrete states $u_{h,k} \in \mathcal{X}_h$ at $t_k = t_0 + k\Delta t$, and the reference solver induces the discrete evolution operator

$$u_{h,k+1} = \mathcal{S}_{h,\Delta t}(u_{h,k}; \mu, g_h). \tag{2}$$

The learned transition is defined on this same discrete state representation, where the solver-level constraints are specified.

2.2 Feasible States

Let m_h be the number of scalar constraints. At a fixed discretization and conditioning, a discrete state $u_h \in \mathcal{X}_h$ is called *feasible* if it satisfies the solver-defined constraint system $C_h u_h = b_h$, where $C_h : \mathcal{X}_h \to \mathbb{R}^{m_h}$ is the discrete constraint operator, and $b_h \in \mathbb{R}^{m_h}$ contains the corresponding prescribed values. The set of all feasible states is

$$\mathcal{M}_h(b_h) = \{u_h \in \mathcal{X}_h : C_h u_h = b_h\}. \tag{3}$$

The operator C_h may combine multiple solver-defined constraints. For incompressible flow, it includes the discrete divergence operator D_h as the block encoding incompressibility, while the corresponding block of b_h is zero. Consequently, every feasible state $u_h \in \mathcal{M}_h(b_h)$ satisfies $D_h u_h = 0$, the discrete counterpart of a divergence-free velocity field. Both C_h and b_h are determined by the numerical problem and are not learnable.

We assume that C_h is linear and that $\mathcal{M}_h(b_h)$ is nonempty for every problem instance considered. Under these conditions, the feasible set is an affine translate of $\ker C_h$ and therefore an embedded affine manifold. We call the algebraic object in Eq. (3) the *solver-defined affine feasible set* and its geometric interpretation the *physical constraint manifold.*

2.3 Problem formulation

Let the i-th trajectory $U_h^{(i)} = \left(u_{h,0}^{(i)}, u_{h,1}^{(i)}, \ldots, u_{h,T_i}^{(i)}\right)$ be generated by the reference solver under physical parameters $\mu^{(i)}$, discretized boundary or forcing data $g_h^{(i)}$, and the corresponding constraint target $b_h^{(i)}$. The training set contains adjacent conditioned function pairs

$$\left\{\left((u_{h,k}^{(i)}, \mu^{(i)}, g_h^{(i)}),\ u_{h,k+1}^{(i)}\right) : i = 1, \ldots, n,\ k = 0, \ldots, T_i - 1\right\},$$

with $u_{h,k}^{(i)}, u_{h,k+1}^{(i)} \in \mathcal{M}_h(b_h^{(i)})$, where n is the number of training trajectories and T_i is the number of transitions in the i-th trajectory.

The goal is to learn a neural operator Φ_θ that approximates the solver-induced transition while preserving the feasible state set:

$$\begin{aligned} &\Phi_\theta(\cdot; \mu, g_h, h) : \mathcal{M}_h(b_h) \to \mathcal{M}_h(b_h), \\ &\Phi_\theta(u_{h,k}; \mu, g_h, h) \approx \mathcal{S}_{h,\Delta t}(u_{h,k}; \mu, g_h). \end{aligned} \tag{4}$$

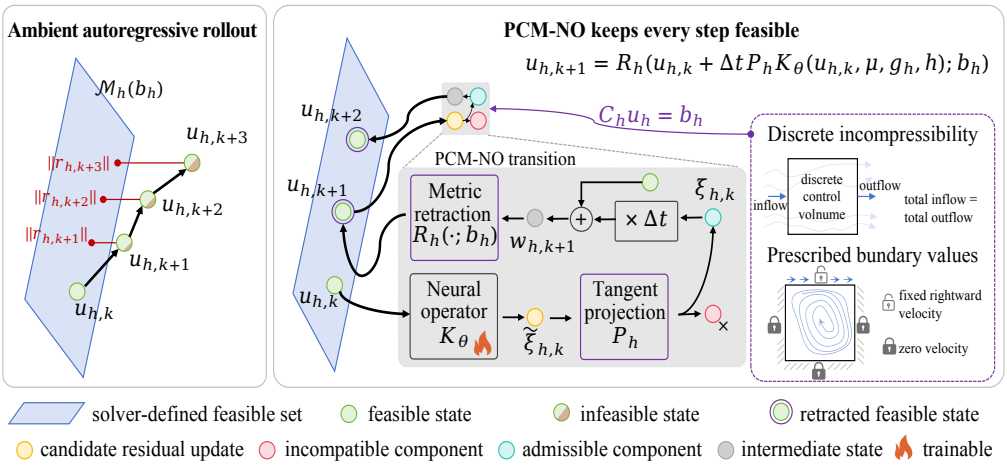


Figure 1: Overview of PCM-NO. An ambient autoregressive rollout can propagate constraint-incompatible components across steps. In PCM-NO, the backbone predicts a candidate update, P_h retains its admissible component, and R_h returns the advanced state to the prescribed affine feasible set. The right panel illustrates solver-defined constraints encoded by $C_h u_h = b_h$.

The first relation in Eq. (4) specifies a structural closure requirement for the learned transition, while the second relation defines the statistical approximation objective.

3 Method

3.1 Overview

Figure 1 contrasts an ambient autoregressive rollout with the feasible-state transition of PCM-NO and shows how its fixed layers are derived from solver-defined constraints. In the ambient rollout, each predicted state becomes the input to the next transition, so any constraint-incompatible component can be propagated and amplified across steps. PCM-NO instead factorizes each learned residual transition into a candidate update, tangent projection, state advancement, and metric retraction. This construction separates data-driven dynamics from solver-level constraint enforcement, allowing the backbone to learn within the feasible state space defined by $C_h u_h = b_h$. The same formulation therefore supports both periodic incompressibility and joint incompressibility–boundary constraints.

Given a feasible state $u_{h,k} \in \mathcal{M}_h(b_h)$, the backbone predicts a candidate update in the ambient discrete space. The tangent projection P_h retains its admissible component in $\ker C_h$ before state advancement, and the metric retraction R_h returns the advanced state to the affine feasible set associated with the current constraint target:

$$\begin{aligned}\Phi_\theta(u_{h,k}; \mu, g_h, h) &= \Big[R_h(\cdot; b_h) \circ \big(\mathrm{Id} + \Delta t\, P_h K_\theta(\cdot, \mu, g_h, h)\big)\Big](u_{h,k}) \\ &= R_h\big(u_{h,k} + \Delta t\, P_h K_\theta(u_{h,k}, \mu, g_h, h); b_h\big),\end{aligned} \tag{5}$$

where Id denotes the identity operator on the discrete state space $\mathcal{X}_h$. Note that only K_θ is trainable. The projection P_h filters the learned update, while the retraction R_h resets state-level feasibility at every rollout step. Both layers are constructed from the discrete constraint system $C_h u_h = b_h$, allowing the same transition-level formulation to accommodate constraints such as discrete incompressibility and prescribed boundary values.

3.2 Tangent projection and retraction

Let $W_h \succ 0$ define the discrete inner product $\langle a_h, b_h \rangle_{W_h} = a_h^\top W_h b_h$, $\|a_h\|_{W_h}^2 = a_h^\top W_h a_h$. For $u_h \in \mathcal{M}_h(b_h)$, let $\mathcal{U}_h := T_{u_h}\mathcal{M}_h(b_h)$ denote the tangent space of $\mathcal{M}_h(b_h)$ at u_h, namely the set of admissible directions along which the constraint value remains unchanged. Because C_h is linear,

$$C_h(u_h + \tau\xi_h) = b_h + \tau C_h \xi_h.$$

Hence $u_h + \tau\xi_h$ remains in $\mathcal{M}_h(b_h)$ precisely when $C_h\xi_h = 0$. The tangent space is therefore independent of the base point and satisfies

$$\mathcal{U}_h = \{\xi_h \in \mathcal{X}_h : C_h\xi_h = 0\} = \ker C_h, \quad u_h \in \mathcal{M}_h(b_h). \tag{6}$$

Tangent projection. Given a candidate update $v_h \in \mathcal{X}_h$, the tangent projection is the W_h-orthogonal projection onto $\mathcal{U}_h$:

$$P_h v_h = \arg\min_{\xi_h \in \mathcal{U}_h} \frac{1}{2}\|\xi_h - v_h\|_{W_h}^2. \tag{7}$$

Thus the projected update $P_h v_h \in \ker C_h$, which can be added to a feasible state without changing the constraint value.

Metric retraction. For an intermediate state $w_h \in \mathcal{X}_h$, we define the metric retraction R_h as the W_h-metric projection onto the affine feasible set:

$$R_h(w_h; b_h) = \arg\min_{u_h \in \mathcal{M}_h(b_h)} \frac{1}{2}\|u_h - w_h\|_{W_h}^2. \tag{8}$$

For a feasible base state $u_h \in \mathcal{M}_h(b_h)$ and a tangent increment $\delta u_h \in \mathcal{U}_h$, the corresponding local map

$$\mathcal{R}_{h,u_h}(\delta u_h) := R_h(u_h + \delta u_h; b_h)$$

satisfies $\mathcal{R}_{h,u_h}(0) = u_h$ and $\mathcal{R}_{h,u_h}(\delta u_h) = u_h + \delta u_h$.

3.3 PCM-NO Rollout Transition

We now specify the projected residual transition summarized in Figure 1. At time level k, the backbone produces a candidate update

$$\widetilde{\xi}_{h,k} = K_\theta(u_{h,k}, \mu, g_h, h), \qquad \widetilde{\xi}_{h,k} \in \mathcal{X}_h.$$

The tangent layer filters this update and advances the current state according to $\xi_{h,k} = P_h\widetilde{\xi}_{h,k}$ and $w_{h,k+1} = u_{h,k} + \Delta t\, \xi_{h,k}$. The complete PCM-NO transition is

$$\Phi_\theta(u_{h,k}; \mu, g_h, h) = R_h\big(u_{h,k} + \Delta t\, P_h K_\theta(u_{h,k}, \mu, g_h, h); b_h\big). \quad (9)$$

The backbone determines a candidate update $\widetilde{\xi}_{h,k}$, while only $\xi_{h,k} = P_h\widetilde{\xi}_{h,k}$ enters the rollout. If $u_{h,k} \in \mathcal{M}_h(b_h)$ and the tangent projection P_h is exact, then the intermediate state is already feasible, i.e., $w_{h,k+1} \in \mathcal{M}_h(b_h)$. In this case, the final retraction acts as the identity, $R_h(w_{h,k+1}; b_h) = w_{h,k+1}$, leaving a feasible intermediate state unchanged. When the tangent solve is approximate or the affine target changes between steps, R_h becomes active by resetting state-level feasibility and mapping the state to the prescribed next feasible set.

Relation to post-hoc state projection. The feasible-transition formulation subsumes post-hoc state projection [20] as a direct-state realization. Let $G_\theta : \mathcal{X}_h \to \mathcal{X}_h$ be an unconstrained next-state predictor and define $K^G_\theta(u_h) = \frac{G_\theta(u_h) - u_h}{\Delta t}$. For $u_h \in \mathcal{M}_h(b_h)$, the exact affine projection identity gives

$$\begin{aligned} R_h\big(G_\theta(u_h); b'_h\big) &= R_h\Big(u_h + \Delta t\, K^G_\theta(u_h); b'_h\Big) \\ &= R_h\Big(u_h + \Delta t\, P_h K^G_\theta(u_h); b'_h\Big). \end{aligned} \quad (10)$$

Thus, post-hoc state projection is the direct-state realization of the affine feasible-transition principle, while PCM-NO is its projected-residual realization. Both inherit exact closure, while PCM-NO makes the admissible update explicit.

Changing affine targets. When the constraint operator C_h remains fixed but its right-hand side changes from $b_{h,k}$ to $b_{h,k+1}$, the same tangent projection can be combined with a retraction onto the next feasible set:

$$\Phi_\theta(u_{h,k}; \mu, g_{h,k}, h) = R_h\big(u_{h,k} + \Delta t\, P_h K_\theta(u_{h,k}, \mu, g_{h,k}, h); b_{h,k+1}\big). \quad (11)$$

Thus, a state in $\mathcal{M}_h(b_{h,k})$ is mapped to a state in $\mathcal{M}_h(b_{h,k+1})$, while the homogeneous admissible direction space remains $\ker C_h$.

3.4 Training objective

Training minimizes prediction error over autoregressive rollout windows. Let H denote the training horizon and s the starting index of a window. For trajectory i, initialize $\widehat{u}^{(i,s)}_{h,0} = u^{(i)}_{h,s}$, and recursively apply

$$\widehat{u}^{(i,s)}_{h,r+1} = \Phi_\theta\Big(\widehat{u}^{(i,s)}_{h,r}; \mu^{(i)}, g^{(i)}_h, h\Big), \qquad r = 0, \dots, H-1.$$

The reported models are trained with

$$\mathcal{L}_{\text{train}}(\theta) = \sum_{i=1}^{n} \sum_{s=0}^{T_i - H} \sum_{r=1}^{H} \omega_r\, \ell_h\Big(\widehat{u}^{(i,s)}_{h,r}, u^{(i)}_{h,s+r}\Big), \quad (12)$$

where ℓ_h is a discrete field-level prediction loss and ω_r controls the relative contribution of rollout step r. Note that the feasibility is specified by the transition architecture in Eq. (9).

4 Structural Analysis

The analysis characterizes the solver-defined tangent projection and metric retraction, as well as the residual behavior of their numerical realizations. All proofs and derivations are given in Appendix A.

Proposition 1 (Exact feasible-state transition)**.** *Any feasible input $u_{h,k} \in \mathcal{M}_h(b_h)$ satisfies*

$$\Phi_\theta(u_{h,k}; \mu, g_h, h) \in \mathcal{M}_h(b_h). \quad (13)$$

If $\widehat{u}_{h,0} \in \mathcal{M}_h(b_h)$, then the exact autoregressive rollout remains in $\mathcal{M}_h(b_h)$ for every $k \geq 0$. More generally, when the target changes between steps,

$$u_{h,k} \in \mathcal{M}_h(b_{h,k}) \implies R_h\big(u_{h,k} + \Delta t\, P_h K_\theta(u_{h,k}); b_{h,k+1}\big) \in \mathcal{M}_h(b_{h,k+1}).$$

Remark 1 (Interpretation of exact closure). Proposition 1 shows that the exact PCM-NO transition maps a feasible state to the prescribed affine feasible set, including under step-dependent targets. This closure is determined by the fixed solver-defined constraint layers and is independent of the learned backbone and its parameters. Sections 5.2–5.4 will evaluate the corresponding numerical feasibility under in-distribution and shifted conditions.

Proposition 2 (Tangent filtering toward admissible increments)**.** *For any candidate increment $\widetilde{\delta u}_h \in \mathcal{X}_h$, with $\widetilde{\delta u}_h = \Delta t\, K_\theta(\cdot)$ in the PCM-NO transition, and any admissible target increment $\delta u^\star_h \in \mathcal{U}_h$,*

$$\begin{aligned} \|\widetilde{\delta u}_h - \delta u^\star_h\|^2_{W_h} &= \|P_h\widetilde{\delta u}_h - \delta u^\star_h\|^2_{W_h} \\ &\quad + \|(I - P_h)\widetilde{\delta u}_h\|^2_{W_h}. \end{aligned} \quad (14)$$

Consequently,

$$\|P_h\widetilde{\delta u}_h - \delta u^\star_h\|_{W_h} \leq \|\widetilde{\delta u}_h - \delta u^\star_h\|_{W_h}. \quad (15)$$

The target may be the difference between two states in the same affine feasible set or the tangent projection of a reference increment.

Remark 2 (Interpretation of tangent filtering). Proposition 2 shows that P_h removes exactly the W_h-orthogonal component outside $\mathcal{U}_h$ and cannot increase the distance to any admissible target increment. When two consecutive reference states satisfy the same affine target, $C_h(u_{h,k+1} - u_{h,k}) = 0$, so their increment lies in $\mathcal{U}_h$ up to the data-generation tolerance. Tangent filtering therefore preserves dynamics within the feasible set while removing constraint-incompatible components. Section 5.5 verifies this decomposition on realized model updates.

Assumption 1 (Approximate numerical constraint layers)**.** At rollout step k, the implemented tangent projection $\widehat{P}_{h,k}$ and metric retraction $\widehat{R}_{h,k}$ satisfy

$$\Big\|C_h\widehat{P}_{h,k}v_{h,k}\Big\| \leq \epsilon^P_{h,k}, \quad \Big\|C_h\widehat{R}_{h,k}(w_h; b_{h,k+1}) - b_{h,k+1}\Big\| \leq \epsilon^R_{h,k}.$$

Here, $\epsilon^P_{h,k} \geq 0$ and $\epsilon^R_{h,k} \geq 0$ denote the per-step residual bounds achieved by the numerical tangent-projection and metric-retraction solves, respectively, which may depend on the discretization, rollout step, and numerical solve budget.

Proposition 3 (Residual accumulation bound and per-step reset)**.** *Let $r_{h,k} = C_h \widehat{u}_{h,k} - b_h$ be the residual for a fixed target b_h. A tangent-only rollout $\widehat{u}^{\tan}_{h,k+1} = \widehat{u}^{\tan}_{h,k} + \Delta t\, \widehat{P}_{h,k} v_{h,k}$ satisfies*

$$\|r^{\tan}_{h,k}\| \le \|r^{\tan}_{h,0}\| + \Delta t \sum_{j=0}^{k-1} \epsilon^{P}_{h,j}. \tag{16}$$

The full transition with a final approximate metric retraction satisfies

$$\|C_h \widehat{u}^{\text{full}}_{h,k+1} - b_{h,k+1}\| \le \epsilon^{R}_{h,k}. \tag{17}$$

Remark 3 (Interpretation of residual propagation). Proposition 3 shows that residuals from approximate tangent projections can accumulate across rollout steps, whereas the metric retraction resets the state residual to the accuracy achieved by the current numerical solve. This behavior will be examined empirically in Section 5.5.

5 Experiments

5.1 Experimental setup

Datasets and evaluation settings. We use two datasets governed by the 2-dimensional incompressible Navier–Stokes equations. **Periodic NS** (**P-NS**) is forced periodic Kolmogorov flow on $[0, 2\pi]^2$, with $C_h = D_h$ and $b_h = 0$, where D_h is the periodic spectral divergence operator. **Lid-driven cavity flow** (**LC**) is defined on $[0, 1]^2$ with no-slip walls and a prescribed moving lid, using $C_h = [D_h; B_h]$ and $b_h = [0; g_h]$, where B_h extracts boundary degrees of freedom and g_h stores the wall and lid velocities. On these datasets, we evaluate **in-distribution rollout**, **long-horizon rollout**, and **stress** settings that vary Reynolds number, lid velocity, resolution, data budget, or projection tolerance without changing the underlying dataset definitions. Appendix C.2 describes the two constraint configurations and the construction of the stress conditions, while Table 5 in Appendix summarizes the data generation, discretization, and split statistics.

Baselines. All methods use FNO-based backbones with matched capacity. **FNO** is the unconstrained autoregressive reference [22]. **DivReg** follows the penalty-based treatment of incompressibility in physics-informed operator learning [23], adding a divergence penalty on P-NS and both divergence and prescribed-boundary penalties on LC. **PINO** augments the rollout objective with a dataset-specific discrete PDE residual, without instance-wise test-time optimization [23]. **ClawNO** is an external structure-preserving baseline evaluated only on P-NS using its FNO-based clawFNO realization [25], since its periodic potential representation does not directly encode the stacked LC constraint $C_h = [D_h; B_h]$. Its implementation details are provided in Appendix D.1. **FinalProj** is a representative post-hoc state-projection baseline, following the direct-state correction principle [20]. On P-NS, it uses a spectral Leray projection; on LC, it enforces the joint affine constraint $C_h = [D_h; B_h]$. **PCM-NO** instead predicts a candidate residual update, projects it onto $\ker C_h$, advances the state, and applies a final metric retraction.

Training and reporting. Unless stated otherwise, all methods use identical splits and autoregressive training windows of length $H = 4$ on 64^2 data with $Re \in [100, 500]$. LC additionally uses prescribed lid velocities in $U_{\text{lid}} \in [0.8, 1.2]$. Method-specific loss weights are fixed before test evaluation. The P-NS PINO residual weight is selected using validation rollouts, while the remaining soft-loss weights are fixed within each dataset. Complete settings are reported in Appendix C.3. Main test results are reported as mean ± standard deviation over three seeds. Constraint-solver diagnostics and end-to-end inference cost are reported in Appendices D.2 and D.3, respectively.

Metrics. Rollout **accuracy** is measured by RelL2_r, where r denotes the rollout horizon. We quantify solver-level **feasibility** using Div_r, the base-10 logarithm of the discrete incompressibility residual. On LC, we additionally report BC_r, defined analogously from the **boundary-condition residual**. We additionally report the **energy relative error** on P-NS and the **centerline profile error** $\mathcal{E}_{\text{prof}}$ on LC. Formal definitions and the aggregation protocol are provided in Appendix C.5.

5.2 In-Distribution Feasible Rollouts

On P-NS, FinalProj, ClawNO, and PCM-NO maintain divergence-free rollouts near the numerical floor of their constraint mechanisms, whereas the soft-constraint methods remain near -2 and FNO becomes non-finite before 50 steps. PCM-NO achieves the lowest RelL2_1, RelL2_{20}, and RelL2_{50}. Although FinalProj reaches a nearly identical Div_{50}, the shared feasibility floor controls for output constraint satisfaction and isolates the effect of transition parameterization. The lower rollout errors of PCM-NO therefore support the empirical value of its projected-residual realization. The corresponding tangent-filtering mechanism is examined directly in Section 5.5. Results in Appendix D.1 further show that PCM-NO substantially improves accuracy and energy error over ClawNO, while the runtime comparison quantifies the cost of the projection–retraction transition.

On LC, where $C_h = [D_h; B_h]$, PCM-NO maintains both incompressibility and the prescribed boundary values near the achieved solve floor while attaining the lowest rollout errors. The joint Div_{50}–BC_{50} behavior shows that the relevant feasible set is the complete affine constraint set and distinguishes hard solver-level feasibility from soft residual reduction.

5.3 Long-Horizon Rollout Stability

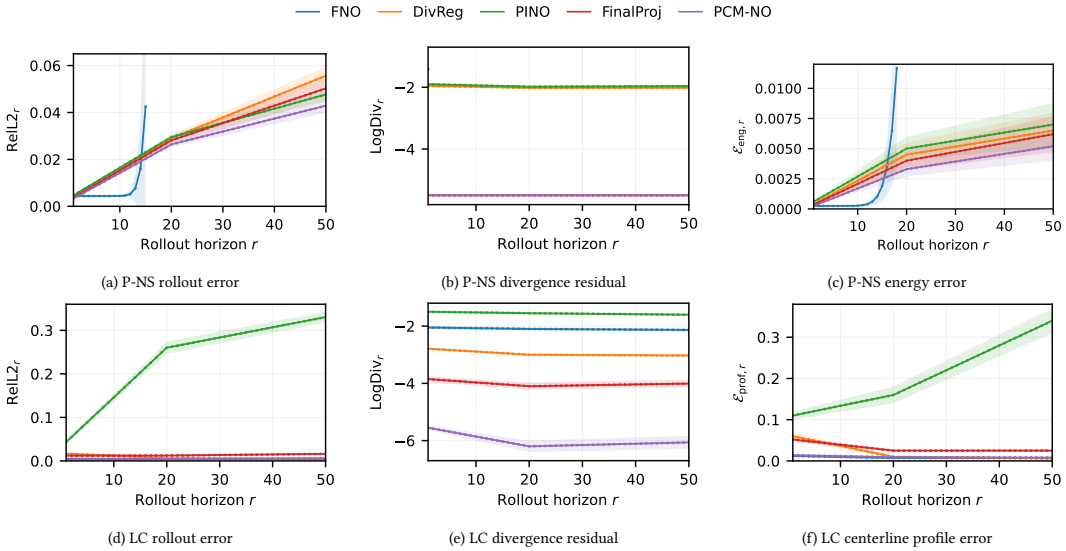
Figure 2 traces accuracy, feasibility, and flow-specific diagnostics across the rollout horizon. On P-NS, FinalProj and PCM-NO maintain projection-level incompressibility, while the unconstrained FNO rollout becomes non-finite. PCM-NO also gives the lowest long-horizon state and energy errors among the finite methods. On LC, it keeps both Div and BC near their numerical floors and substantially reduces the state and centerline-profile errors. The results show that the feasible transition remains effective under repeated composition and that its strongest predictive advantage appears when incompressibility and prescribed boundary values must be enforced jointly.

5.4 Generalization and Stress Diagnostics

We evaluate four stress settings: P-NS Reynolds-number extrapolation (**P-NS-Re**), LC Reynolds-number extrapolation (**LC-Re**), LC lid-velocity shift (**LC-Lid**), and P-NS resolution transfer (**P-NS-Grid**). Consistent with the pointwise closure property in Proposition 1, Table 2 shows that PCM-NO preserves near-numerical-floor feasibility beyond the training distribution. It maintains Div_{50} on

Dataset	Method	$\mathrm{RelL2}_1 \downarrow$	$\mathrm{RelL2}_{20} \downarrow$	$\mathrm{RelL2}_{50} \downarrow$	$\mathrm{Div}_{50} \downarrow$	$\mathrm{BC}_{50} \downarrow$
P-NS	FNO	0.0044 ± 0.0000	5.0484 ± 8.6469	N.F.	N.F.	–
	DivReg	0.0041 ± 0.0000	0.0290 ± 0.0015	0.0557 ± 0.0080	−2.0171 ± 0.0089	–
	PINO	0.0047 ± 0.0000	0.0295 ± 0.0003	0.0477 ± 0.0011	−1.9620 ± 0.0067	–
	ClawNO	0.0038 ± 0.0285	0.0266 ± 0.0015	0.9270 ± 0.3536	−5.3557 ± 0.0179	–
	FinalProj	0.0037 ± 0.0001	0.0281 ± 0.0006	0.0503 ± 0.0017	**−5.4981 ± 0.0001**	–
	PCM-NO	**0.0033 ± 0.0001**	**0.0264 ± 0.0002**	**0.0429 ± 0.0005**	−5.4978 ± 0.0002	–
LC	FNO	0.0042 ± 0.0005	0.0049 ± 0.0001	0.0058 ± 0.0018	−2.1344 ± 0.0524	−3.8770 ± 0.0219
	DivReg	0.0167 ± 0.0030	0.0063 ± 0.0017	0.0050 ± 0.0001	−3.0271 ± 0.0128	−3.9249 ± 0.0160
	PINO	0.0433 ± 0.0042	0.2601 ± 0.0142	0.3305 ± 0.0127	−1.6027 ± 0.0359	−1.4254 ± 0.0441
	FinalProj	0.0121 ± 0.0062	0.0123 ± 0.0023	0.0162 ± 0.0019	−4.0080 ± 0.1363	**−6.0000 ± 0.0000**
	PCM-NO	**0.0037 ± 0.0001**	**0.0043 ± 0.0002**	**0.0039 ± 0.0001**	**−6.0625 ± 0.2304**	**−6.0000 ± 0.0000**

Table 1: In-distribution rollout accuracy and feasibility. Div_{50} **uses spectral divergence on P-NS and interior MAC control volumes on LC;** BC_{50} **is reported only for LC. N.F. denotes a non-finite rollout before the stated horizon.**



(a) P-NS rollout error (b) P-NS divergence residual (c) P-NS energy error

(d) LC rollout error (e) LC divergence residual (f) LC centerline profile error

Figure 2: Long-rollout diagnostics. Shaded bands show the mean ± one standard deviation over three seeds. FNO curves are truncated once the corresponding rollout becomes non-finite in the P-NS accuracy and energy panels.

P-NS-Re, both Div_{50} and BC_{50} on LC-Re and LC-Lid, and near-feasibility after reconstructing the constraint operators on the 128^2 P-NS grid. Across all four settings, PCM-NO also achieves the lowest $\mathrm{RelL2}_{50}$ and flow-specific diagnostic error, whereas FinalProj reaches a comparable feasibility floor, so the lower state and flow-diagnostic errors of PCM-NO isolate the predictive advantage of the projected-residual realization under distribution shift. LC-Lid directly changes the affine target b_h, and P-NS-Grid evaluates target-grid transfer under reconstructed constraint operators in a setting where grid transfer, unseen-frequency extrapolation, and aliasing are closely coupled [36].

Figure 3 further shows that PCM-NO better preserves the P-NS vorticity organization and LC cavity circulation after 50 steps, consistent with Table 2. Detailed protocols and metric definitions are provided in Appendices C.2 and C.5.

5.5 Mechanism Diagnostics

Tangent filtering. Proposition 2 states that projection onto the discrete tangent space cannot increase the distance from a candidate update to an admissible target. We test this property on realized updates from the trained PCM-NO models by comparing raw and projected candidate updates with the projected reference increment. As shown in Table 3, projection consistently reduces this error on both datasets, while the normalized Pythagorean gap remains near numerical precision. The removed normal component also accounts for a substantial fraction of the candidate-update energy, confirming that the tangent layer actively filters constraint-incompatible directions. This diagnostic verifies the local mechanism in Proposition 2, while Section 5.6 evaluates its role in the complete transition.

Residual accumulation and reset. Figure 4 examines the behavior predicted by Proposition 3. In a controlled LC replay, candidate updates generated along the full PCM-NO rollout are projected once using the evaluation solve budget and applied to two branches. TangentOnly omits the final retraction, whereas Full retracts the same update sequence after every step, so the branches differ only in the state-level reset. TangentOnly accumulates incompressibility error, while Full remains near the achieved constraint-solve floor. Both preserve the prescribed boundary values because the projected updates satisfy the homogeneous boundary constraint, and the

Settings	Method	RelL2$_{50}$ ↓	Div$_{50}$ ↓	BC$_{50}$ ↓	Energy/Profile Err. ↓
P-NS-Re $Re \in \{700, 1000\}$	FNO	N.F.	N.F.	–	N.F.
	DivReg	0.0999 ± 0.0006	−1.9531 ± 0.0079	–	0.0091 ± 0.0051
	PINO	0.1091 ± 0.0055	−1.8891 ± 0.0054	–	0.0105 ± 0.0010
	FinalProj	0.1161 ± 0.0103	−5.5000 ± 0.0009	–	0.0275 ± 0.0205
	PCM-NO	**0.0964 ± 0.0010**	**−5.5006 ± 0.0005**	–	**0.0059 ± 0.0013**
LC-Re $Re \in \{750, 1000\}$	FNO	0.0255 ± 0.0050	−1.6761 ± 0.0374	−3.1378 ± 0.1074	0.1493 ± 0.0257
	DivReg	0.0373 ± 0.0129	−2.7939 ± 0.0618	−3.1870 ± 0.0736	0.2134 ± 0.0387
	PINO	0.2799 ± 0.0288	−1.5863 ± 0.1256	−1.5279 ± 0.0522	0.8063 ± 0.1889
	FinalProj	0.0550 ± 0.0115	−4.0374 ± 0.1134	**−6.0000 ± 0.0000**	0.3059 ± 0.0762
	PCM-NO	**0.0224 ± 0.0082**	**−6.1617 ± 0.2751**	**−6.0000 ± 0.0000**	**0.1025 ± 0.0324**
LC-Lid $U_{\text{lid}} = 1.5$	FNO	0.0596 ± 0.0148	−1.3433 ± 0.0264	−2.4675 ± 0.1289	0.1415 ± 0.0287
	DivReg	0.0839 ± 0.0135	−2.1814 ± 0.0759	−2.4631 ± 0.0400	0.1879 ± 0.0155
	PINO	0.3579 ± 0.0115	−1.4067 ± 0.0812	−1.2386 ± 0.0266	0.6772 ± 0.0122
	FinalProj	0.0601 ± 0.0363	−3.3582 ± 0.1269	**−6.0000 ± 0.0000**	0.1151 ± 0.0432
	PCM-NO	**0.0143 ± 0.0022**	**−5.9020 ± 0.3165**	**−6.0000 ± 0.0000**	**0.0311 ± 0.0031**
P-NS-Grid $64^2 \to 128^2$	FNO	N.F.	N.F.	–	N.F.
	DivReg	0.0511 ± 0.0009	−1.0807 ± 0.0588	–	0.0140 ± 0.0033
	PINO	0.1002 ± 0.0052	−1.2159 ± 0.0393	–	0.0080 ± 0.0009
	FinalProj	5.1918e3 ± 7.3833e3	−5.1451 ± 0.0240	–	1.1921e10 ± 2.0514e10
	PCM-NO	**0.0447 ± 0.0004**	**−5.1798 ± 0.0009**	–	**0.0075 ± 0.0007**

Table 2: Stress-set rollout results, using the in-distribution validation-selected checkpoints without adaptation. Reynolds-number rows pool the two test values. The last column reports energy error on P-NS and profile error on LC; N.F. denotes a non-finite rollout.

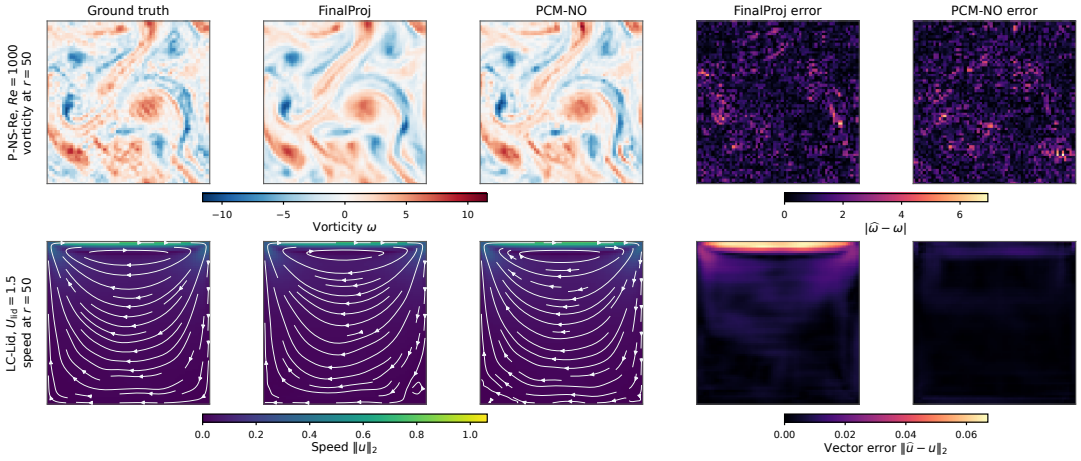


Figure 3: Representative 50-step stress predictions: P-NS Reynolds extrapolation at $Re = 1000$ and the LC lid shift at $U_{\text{lid}} = 1.5$. Rows share field and error color scales, respectively. Trajectories are selected by median ground-truth kinetic energy, independently of model error.

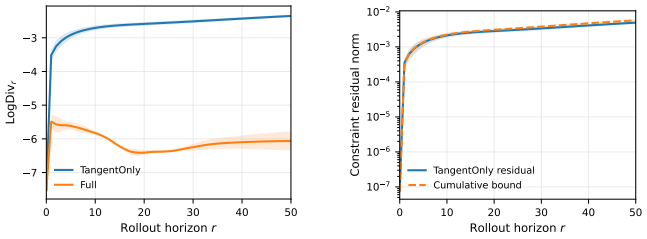
Dataset	Raw RelErr. ↓	Projected RelErr. ↓	Normal Frac. (%)	Pythag. Gap ↓
P-NS	0.8220 ± 0.0258	0.0607 ± 0.0014	39.75 ± 1.52	$(5.22 \pm 0.16) \times 10^{-8}$
LC	3.8898 ± 0.8964	0.0559 ± 0.0090	87.76 ± 3.19	$(5.96 \pm 0.37) \times 10^{-8}$

Table 3: Tangent-filtering diagnostic. Errors are measured against the projected reference increment; Normal Frac. is the removed candidate-update energy fraction, and Pythag. Gap is the normalized residual of Proposition 2.

tangent-only residual remains below the cumulative projection-residual bound. These results directly support the accumulation-versus-reset interpretation in Proposition 3.

5.6 Component Ablations

Having directly verified tangent filtering and residual reset, we next examine how tangent projection, state retraction, and the complete affine constraint affect the learned rollout. We conduct the ablation on LC because it activates both incompressibility and prescribed boundary constraints. We consider three structural variants. **TangentOnly** retains the residual-update parameterization and the tangent projection, but omits the final retraction: $u_{h,k+1} = u_{h,k} + \Delta t\, P_h K_\theta(u_{h,k}, \mu, g_h, h)$. **RetractOnly** removes the tangent projection and retracts the unconstrained residual update: $u_{h,k+1} = R_h\big(u_{h,k} + \Delta t\, K_\theta(u_{h,k}, \mu, g_h, h); b_h\big)$. **DivOnly** applies both fixed layers using the reduced constraint $C_h = D_h$, thereby removing the prescribed boundary constraint $B_h u_h = g_h$. The full



(a) Accumulation versus per-step reset.

(b) Realized residual and cumulative upper bound.

Figure 4: Controlled LC replay with identical projected updates.

Variant	RelL2$_{50}$ ↓	Div$_{50}$ ↓	BC$_{50}$ ↓	$\mathcal{E}_{\text{prof},50}$ ↓
TangentOnly	0.0156 ± 0.0021	−1.2424 ± 0.0587	**−6.0000 ± 0.0000**	0.0723 ± 0.0129
RetractOnly	0.0061 ± 0.0001	−2.7142 ± 0.0288	**−6.0000 ± 0.0000**	0.0208 ± 0.0007
DivOnly	0.0739 ± 0.0121	−2.7606 ± 0.1050	−1.8536 ± 0.0653	0.2075 ± 0.0269
PCM-NO	**0.0039 ± 0.0001**	**−6.0625 ± 0.2304**	**−6.0000 ± 0.0000**	**0.0088 ± 0.0027**

Table 4: LC component ablation with 50-step rollouts.

PCM-NO transition retains tangent projection, boundary-aware constraints, and state retraction.

Table 4 separates three numerical roles. TangentOnly retains the projected residual update but removes the state-level reset. Its increasing constraint residual is consistent with the accumulation mechanism in Proposition 3. RetractOnly applies the reset to an unconstrained residual update and therefore recovers much of the feasibility and accuracy. Relative to RetractOnly, PCM-NO reduces RelL2$_{50}$ by 36% and $\mathcal{E}_{\text{prof},50}$ by 58%, while improving Div$_{50}$ by 3.35 log units. These gains quantify the empirical benefit of the explicit projected-update parameterization under finite constraint solves. DivOnly removes the prescribed-boundary component from C_h, leading to substantial boundary violation and degraded profile accuracy. Together, the results support the complete affine constraint, the per-step feasibility reset, and the explicit admissible-update parameterization as complementary components of the implemented transition.

Computational implications. To place the accuracy–feasibility gains in a solver context, Appendix D.3 compares complete learned transitions with the corresponding reference solvers over the same physical horizon. At batch size one, PCM-NO remains 7.6× faster than the P-NS reference solver and 6.4× faster than the LC reference solver over a 50-step rollout. Constraint enforcement costs 1.37× the FNO rollout on P-NS, but 41.53× on LC, where the two iterative constraint solves dominate inference. The computational overhead of the solver-defined constraint layers is therefore discretization-dependent, motivating warm starts, preconditioning, and adaptive solve budgets for boundary-aware settings.

6 Related Work

Neural operators and temporal PDE surrogates. Neural operators learn mappings between function spaces through DeepONet, FNO, and their factorized, geometry-adaptive, convolutional, localized, mesh-based, graph, and transformer variants [14, 18, 21, 22, 27, 28, 31, 33, 39, 43]. Foundation models and benchmarks broaden the task scope [15, 37], while autoregressive, refinement, memory, and stability-oriented methods target long-horizon accuracy [2, 3, 24, 29]. These works do not generally require every composed transition to remain in a solver-defined feasible set.

Physics-aware and constrained dynamics. Physics-informed, conservation aware, divergence-free, and boundary-aware operators incorporate PDE structure through residual objectives, data-free or boundary-integral formulations, and specialized parameterizations [4, 5, 8, 9, 17, 23, 25, 26, 32, 34, 41, 42]. EquiNO imposes equilibrium through divergence-free POD bases, while CompNO composes learned convection and diffusion modules with a Poisson solver [7, 16]. Related structure-preserving methods use Hodge decompositions, conservative transport, port–Hamiltonian factorizations, hard-constrained generative sampling, or classical projection methods [1, 6, 12, 19, 30, 38, 40, 44]. Neural–numerical and explicitly constrained models similarly combine learned mappings with analytical or iterative structure [10, 11, 13, 35], and post-hoc projection can enforce divergence-free outputs [20]. PCM-NO instead defines an autoregressive map between solver-defined affine feasible sets through a projected residual parameterization, supporting joint constraints and nonhomogeneous targets while enabling analysis of exact closure and approximate-solve residual propagation.

7 Conclusion

We introduced PCM-NO, an autoregressive neural-operator construction whose transition is closed on a solver-defined affine constraint manifold. The backbone predicts a candidate update direction, while fixed tangent projection and state-retraction layers determine the admissible increment and the feasible output state. The structural analysis establishes exact feasible-state closure, orthogonal filtering of constraint-incompatible increment components, and residual accumulation and reset bounds for approximate numerical solves. Across periodic and boundary-constrained incompressible-flow settings, PCM-NO maintains discrete incompressibility and prescribed boundary values over long rollouts and under controlled parameter, boundary-value, and resolution shifts. The experiments also show consistent improvements in rollout accuracy and flow-specific diagnostics. Extending the construction to nonlinear, state-dependent, and geometry-dependent constraint sets is a natural direction for future work.

8 Limitations and Ethical Considerations

The current formulation is limited to fixed linear discrete constraints and nonempty affine feasible sets. The experiments cover two-dimensional incompressible flows with an FNO backbone, while three-dimensional systems, irregular geometries, and nonlinear or state-dependent constraints remain to be studied. The cost of the constraint layers is also discretization dependent, with the iterative LC solver introducing substantially greater overhead than the spectral P-NS implementation.

The datasets are synthetic and contain no human or personally identifiable information. Learned CFD surrogates may nevertheless produce plausible but inaccurate predictions under untested conditions. Safety-critical use therefore requires independent validation, error estimation, and a verified numerical solver as fallback.

9 Generative AI Usage

Generative AI tools were used to assist language editing, structural revision, and code drafting. The authors reviewed all generated content and independently verified the mathematical derivations, implementations, experimental protocols, and reported results. Generative AI was not treated as an author and did not make autonomous scientific decisions.

References

[1] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. 2006. Finite element exterior calculus, homological techniques, and applications. *Acta Numerica* 15 (2006), 1–155.

[2] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. 2022. Message Passing Neural PDE Solvers. In *International Conference on Learning Representations*.

[3] Ricardo Buitrago Ruiz, Tanya Marwah, Albert Gu, and Andrej Risteski. 2025. On the Benefits of Memory for Modeling Time-Dependent PDEs. In *International Conference on Learning Representations*.

[4] Jinbao Cheng, Jianguo Huang, Haoqin Wang, and Tao Zhou. 2026. Decoupled Divergence-Free Neural Networks Basis Method for Incompressible Fluid Problems. *arXiv preprint arXiv:2603.17906* (2026).

[5] Junho Choi, Teng-Yuan Chang, Namjung Kim, and Youngjoon Hong. 2025. A Data-Free Neural Operator Enabling Fast Inference of 2D and 3D Navier–Stokes Equations. *arXiv preprint arXiv:2510.23936* (2025).

[6] Alexandre Joel Chorin. 1968. Numerical solution of the Navier-Stokes equations. *Math. Comp.* 22, 104 (1968), 745–762.

[7] Hamidreza Eivazi, Jendrik-Alexander Tröger, Stefan Wittek, Stefan Hartmann, and Andreas Rausch. 2025. EquiNO: A Physics-Informed Neural Operator for Multiscale Simulations. *arXiv preprint arXiv:2504.07976* (2025).

[8] Luis Espath, Dmitry Kabanov, Jonas Kiessling, and Raúl Tempone. 2021. Statistical Learning for Fluid Flows: Sparse Fourier Divergence-Free Approximations. *arXiv preprint arXiv:2107.07633* (2021).

[9] Zhiwei Fang, Sifan Wang, and Paris Perdikaris. 2023. Learning Only on Boundaries: A Physics-Informed Neural Operator for Solving Parametric Partial Differential Equations in Complex Geometries. *arXiv preprint arXiv:2308.12939* (2023).

[10] Marc Finzi, Ke Alexander Wang, and Andrew Gordon Wilson. 2020. Simplifying Hamiltonian and Lagrangian Neural Networks via Explicit Constraints. In *Advances in Neural Information Processing Systems*, Vol. 33.

[11] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. 2019. Hamiltonian Neural Networks. In *Advances in Neural Information Processing Systems*.

[12] Jean-Luc Guermond, Peter Minev, and Jie Shen. 2006. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering* 195, 44–47 (2006), 6011–6045.

[13] Wenrui Hao, Xinliang Liu, and Yahong Yang. 2024. Newton Informed Neural Operator for Solving Nonlinear Partial Differential Equations. In *Advances in Neural Information Processing Systems*.

[14] Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. 2023. GNOT: A General Neural Operator Transformer for Operator Learning. In *Proceedings of the 40th International Conference on Machine Learning*.

[15] Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. 2024. POSEIDON: Efficient Foundation Models for PDEs. In *Advances in Neural Information Processing Systems*.

[16] Hamda Hmida, Hsiu-Wen Chang, and Youssef Mesri. 2026. Compositional Neural Operators for Multi-Dimensional Fluid Dynamics. In *International Conference on Learning Representations*.

[17] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. 2021. Physics-informed machine learning. *Nature Reviews Physics* 3, 6 (2021), 422–440.

[18] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. 2023. Neural Operator: Learning Maps Between Function Spaces. *Journal of Machine Learning Research* 24, 89 (2023), 1–97.

[19] Zishuo Lan, Junjie Li, Lei Wang, and Jincheng Wang. 2026. FluxNet: Learning Capacity-Constrained Local Transport Operators for Conservative and Bounded PDE Surrogates. In *Proceedings of the 43rd International Conference on Machine Learning*.

[20] Xigui Li, Hongwei Zhang, Ruoxi Jiang, Deshu Chen, Chensen Lin, Limei Han, Yuan Qi, Xin Guo, and Yuan Cheng. 2026. Project and Generate: Divergence-Free Neural Operators for Incompressible Flows. *arXiv preprint arXiv:2603.24500* (2026).

[21] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. 2023. Fourier Neural Operator with Learned Deformations for PDEs on General Geometries. *Journal of Machine Learning Research* 24, 388 (2023), 1–26.

[22] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. 2021. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*.

[23] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Andrew M. Stuart, Kamyar Azizzadenesheli, and Anima Anandkumar. 2022. Physics-Informed Neural Operator for Learning Partial Differential Equations. In *International Conference on Learning Representations*.

[24] Phillip Lippe, Bastiaan S. Veeling, Paris Perdikaris, Richard E. Turner, and Johannes Brandstetter. 2023. PDE-Refiner: Achieving Accurate Long Rollouts with Neural PDE Solvers. In *Advances in Neural Information Processing Systems*.

[25] Ning Liu, Yiming Fan, Xianyi Zeng, Milan Klöwer, Lu Zhang, and Yue Yu. 2024. Harnessing the Power of Neural Operators with Automatically Encoded Conservation Laws. In *Proceedings of the 41st International Conference on Machine Learning*.

[26] Ning Liu, Yue Yu, Huaiqian You, and Neeraj Tatikola. 2023. INO: Invariant Neural Operators for Learning Complex Physical Systems with Momentum Conservation. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*.

[27] Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. 2024. Neural Operators with Localized Integral and Differential Kernels. In *Proceedings of the 41st International Conference on Machine Learning*.

[28] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence* 3, 3 (2021), 218–229.

[29] Michael McCabe, Peter Harrington, Shashank Subramanian, and Jed Brown. 2023. Towards Stability of Autoregressive Neural Operators. *Transactions on Machine Learning Research* (2023).

[30] Satoshi Noguchi and Yoshinobu Kawahara. 2026. Mesh Field Theory: Port–Hamiltonian Formulation of Mesh-Based Physics. In *Proceedings of the 43rd International Conference on Machine Learning*.

[31] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. 2021. Learning Mesh-Based Simulation with Graph Networks. In *International Conference on Learning Representations*.

[32] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378 (2019), 686–707.

[33] Bogdan Raonić, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. 2023. Convolutional Neural Operators for Robust and Accurate Learning of PDEs. In *Advances in Neural Information Processing Systems*.

[34] Jack Richter-Powell, Yaron Lipman, and Ricky T. Q. Chen. 2022. Neural Conservation Laws: A Divergence-Free Perspective. In *Advances in Neural Information Processing Systems*.

[35] Alexander Rudikov, Vladimir Fanaskov, Ekaterina Muravleva, Yuri M. Laevsky, and Ivan Oseledets. 2024. Neural Operators Meet Conjugate Gradients: The FCG-NO Method for Efficient PDE Solving. In *Proceedings of the 41st International Conference on Machine Learning*.

[36] Mansi Sakarvadia, Kareem Hegazy, Amin Totounferoush, Kyle Chard, Yaoqing Yang, Ian Foster, and Michael W. Mahoney. 2026. The False Promise of Zero-Shot Super-Resolution in Machine-Learned Operators. In *The Fourteenth International Conference on Learning Representations*.

[37] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. 2022. PDEBench: An Extensive Benchmark for Scientific Machine Learning. In *Advances in Neural Information Processing Systems*.

[38] Roger Temam. 1969. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (II). *Archive for Rational Mechanics and Analysis* 33 (1969), 377–385.

[39] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. 2023. Factorized Fourier Neural Operators. In *International Conference on Learning Representations*.

[40] Utkarsh Utkarsh, Pengfei Cai, Alan Edelman, Rafael Gómez-Bombarelli, and Christopher Rackauckas. 2025. Physics-Constrained Flow Matching: Sampling Generative Models with Hard Constraints. In *Advances in Neural Information Processing Systems*, Vol. 38.

[41] Haixin Wang, Jiaxin Li, Anubhav Dwivedi, Kentaro Hara, and Tailin Wu. 2024. BENO: Boundary-Embedded Neural Operators for Elliptic PDEs. In *International Conference on Learning Representations*.

[42] Sifan Wang, Hanwen Wang, and Paris Perdikaris. 2021. Learning the Solution Operator of Parametric Partial Differential Equations with Physics-Informed DeepONets. *Science Advances* 7, 40 (2021), eabi8605.

[43] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. 2024. Transolver: A Fast Transformer Solver for PDEs on General Geometries. In *Proceedings of the 41st International Conference on Machine Learning.*
[44] Dongzhe Zheng, Tao Zhong, and Christine Allen-Blanchette. 2026. Topology-Preserving Neural Operator Learning via Hodge Decomposition. In *Proceedings of the 43rd International Conference on Machine Learning.*

A Proofs and Approximate Constraint Solves

This appendix proves the structural statements in Section 4 and relates approximate constraint layers to the achieved residuals of their linear systems.

A.1 Proof of Proposition 1

Let

$$A_h = C_h W_h^{-1} C_h^\top.$$

For every $z \in \mathbb{R}^{m_h}$,

$$z^\top A_h z = \|W_h^{-1/2} C_h^\top z\|_2^2.$$

Hence $\ker A_h = \ker C_h^\top$. Because A_h is symmetric,

$$\operatorname{range}(A_h) = (\ker A_h)^\perp = (\ker C_h^\top)^\perp = \operatorname{range}(C_h).$$

The KKT conditions for the tangent projection yield

$$P_h v_h = v_h - W_h^{-1} C_h^\top A_h^\dagger C_h v_h.$$

Since $C_h v_h \in \operatorname{range}(C_h) = \operatorname{range}(A_h)$,

$$\begin{aligned} C_h P_h v_h &= C_h v_h - A_h A_h^\dagger C_h v_h \\ &= 0. \end{aligned}$$

For the state projection,

$$R_h(w_h; b_h) = w_h - W_h^{-1} C_h^\top A_h^\dagger (C_h w_h - b_h).$$

The nonemptiness of $\mathcal{M}_h(b_h)$ implies $b_h \in \operatorname{range}(C_h)$, and therefore $C_h w_h - b_h \in \operatorname{range}(A_h)$. It follows that

$$\begin{aligned} C_h R_h(w_h; b_h) &= C_h w_h - A_h A_h^\dagger (C_h w_h - b_h) \\ &= b_h. \end{aligned}$$

Finally, if $u_{h,k} \in \mathcal{M}_h(b_h)$, then

$$\begin{aligned} C_h \left(u_{h,k} + \Delta t\, P_h K_\theta(u_{h,k}, \mu, g_h, h)\right) &= C_h u_{h,k} + \Delta t\, C_h P_h K_\theta(u_{h,k}, \mu, g_h, h) \\ &= b_h. \end{aligned}$$

The subsequent state projection preserves membership in $\mathcal{M}_h(b_h)$. Induction over k proves the rollout statement.

A.2 Derivation of Eq. (10)

Define

$$Q_h = W_h^{-1} C_h^\top A_h^\dagger, \qquad P_h = I - Q_h C_h.$$

For $u_h \in \mathcal{M}_h(b_h)$ and any feasible target value b_h',

$$\begin{aligned} R_h(u_h + \Delta t\, v_h; b_h') &= u_h + \Delta t\, v_h - Q_h\big(b_h + \Delta t\, C_h v_h - b_h'\big) \\ &= u_h + \Delta t\, P_h v_h - Q_h(b_h - b_h') \\ &= R_h(u_h + \Delta t\, P_h v_h; b_h'). \end{aligned}$$

Taking $b_h' = b_h$ gives

$$R_h(u_h + \Delta t\, v_h; b_h) = u_h + \Delta t\, P_h v_h,$$

which proves Eq. (10).

A.3 Proof of Proposition 2

The projection P_h is the W_h-orthogonal projection onto $\mathcal{U}_h = \ker C_h$. Hence $P_h \widetilde{\delta u}_h \in \mathcal{U}_h$, while $(I - P_h)\widetilde{\delta u}_h$ is W_h-orthogonal to $\mathcal{U}_h$. Since $\delta u_h^\star \in \mathcal{U}_h$,

$$\left\langle P_h \widetilde{\delta u}_h - \delta u_h^\star, (I - P_h)\widetilde{\delta u}_h \right\rangle_{W_h} = 0.$$

Therefore, the decomposition

$$\widetilde{\delta u}_h - \delta u_h^\star = \left(P_h \widetilde{\delta u}_h - \delta u_h^\star\right) + (I - P_h)\widetilde{\delta u}_h$$

has W_h-orthogonal components. Taking squared W_h-norms gives the Pythagorean identity in Proposition 2. Dropping the nonnegative normal-component term yields the stated inequality.

A.4 Approximate Solves and Proof of Proposition 3

Let

$$A_h = C_h W_h^{-1} C_h^\top.$$

The implementation may solve a damped normal equation

$$(A_h + \gamma_h I)\widehat{\lambda} = d_h$$

with $\gamma_h \geq 0$. This includes the undamped case $\gamma_h = 0$.

For tangent projection, set $d_{h,k}^P = C_h v_{h,k}$, define the achieved linear-system residual by

$$\rho_{h,k}^P = (A_h + \gamma_h I)\widehat{\lambda}_{h,k}^P - d_{h,k}^P,$$

and let

$$\widehat{P}_{h,k} v_{h,k} = v_{h,k} - W_h^{-1} C_h^\top \widehat{\lambda}_{h,k}^P.$$

Then

$$\begin{aligned} C_h \widehat{P}_{h,k} v_{h,k} &= d_{h,k}^P - A_h \widehat{\lambda}_{h,k}^P && (18) \\ &= \gamma_h \widehat{\lambda}_{h,k}^P - \rho_{h,k}^P. && (19) \end{aligned}$$

Thus one may take

$$\epsilon_{h,k}^P = \|\gamma_h \widehat{\lambda}_{h,k}^P - \rho_{h,k}^P\|,$$

or use the upper bound

$$\epsilon_{h,k}^P \leq \|\rho_{h,k}^P\| + \gamma_h \|\widehat{\lambda}_{h,k}^P\|.$$

For state projection, define

$$d_{h,k}^R = C_h w_h - b_{h,k+1},$$

$$\rho_{h,k}^R = (A_h + \gamma_h I)\widehat{\lambda}_{h,k}^R - d_{h,k}^R,$$

and

$$\widehat{R}_{h,k}(w_h; b_{h,k+1}) = w_h - W_h^{-1} C_h^\top \widehat{\lambda}_{h,k}^R.$$

The resulting feasibility residual is

$$C_h \widehat{R}_{h,k}(w_h; b_{h,k+1}) - b_{h,k+1} = \gamma_h \widehat{\lambda}_{h,k}^R - \rho_{h,k}^R.$$

Consequently,

$$\epsilon_{h,k}^R = \|\gamma_h \widehat{\lambda}_{h,k}^R - \rho_{h,k}^R\|$$

is valid. When $\gamma_h = 0$, these expressions reduce to the undamped residual identities.

For the tangent-only rollout,

$$r_{h,k+1}^{\tan} = r_{h,k}^{\tan} + \Delta t\, C_h \widehat{P}_{h,k} v_{h,k}.$$

Repeated application of the triangle inequality gives

$$\|r_{h,k}^{\tan}\| \le \|r_{h,0}^{\tan}\| + \Delta t \sum_{j=0}^{k-1} \epsilon_{h,j}^{P},$$

which proves Eq. (16). The full and retract-only transitions end each step with $\widehat{R}_{h,k}$, so Eq. (17) follows directly from Assumption 1.

B Discrete Constraint Operators

This appendix specifies the discrete operators used in the two constraint configurations. The same operators are used for data generation, projection or retraction, training-time rollout, and evaluation.

B.1 P-NS Spectral Constraint

For P-NS, $C_h = D_h$ is the periodic spectral divergence operator. Let G_h denote the corresponding spectral gradient. Under the discrete periodic inner product,

$$G_h = -D_h^{*_{W_h}},$$

so the Helmholtz projection

$$P_h = I - G_h (D_h G_h)^{\dagger} D_h \tag{20}$$

is the W_h-orthogonal projection onto $\ker D_h$. It is implemented in Fourier space, with the zero mode left unchanged. The same spectral divergence operator is used by the constraint layer and by the reported P-NS feasibility metric.

B.2 LC MAC-Grid Constraint

LC states are represented on a MAC-style finite-volume grid. Let $B_h u_h = g_h$ encode the no-slip walls and the moving lid, and let $D_h u_h = 0$ denote finite-volume incompressibility on the interior control volumes. Formally,

$$C_h = \begin{bmatrix} D_h \\ B_h \end{bmatrix}, \qquad b_h = \begin{bmatrix} 0 \\ g_h \end{bmatrix}.$$

The implementation enforces this stacked constraint through boundary lifting and a reduced-coordinate projection. Let $\ell_h(g_h)$ be the stored MAC field containing the prescribed wall and ghost values, and let

$$E_h : \mathcal{Z}_h \to \mathcal{X}_h$$

inject free-face variables into the full MAC state. Every state satisfying the boundary constraint has the unique representation

$$u_h = \ell_h(g_h) + E_h z_h, \qquad B_h E_h = 0, \qquad B_h \ell_h(g_h) = g_h. \tag{21}$$

Define

$$\overline{D}_h = D_h E_h, \qquad c_h(g_h) = -D_h \ell_h(g_h).$$

Then

$$D_h u_h = 0 \quad \Longleftrightarrow \quad \overline{D}_h z_h = c_h(g_h).$$

Thus the full constraint $C_h u_h = b_h$ is equivalent to a reduced divergence constraint on the free velocity faces.

Let

$$\overline{W}_h = E_h^{\top} W_h E_h.$$

The reduced tangent projection is

$$\overline{P}_h = I - \overline{W}_h^{-1} \overline{D}_h^{\top} \left(\overline{D}_h \overline{W}_h^{-1} \overline{D}_h^{\top} \right)^{\dagger} \overline{D}_h, \tag{22}$$

Setting	P-NS	LC
Domain	$[0, 2\pi]^2$ Dealiased pseudo-spectral	$[0, 1]^2$ MAC-grid projection
Generator	RK4 solver	solver
State representation	Velocity field	Staggered velocity field
Spatial resolution	64×64	64×64
Stored states per trajectory	56	56
One-step transitions per trajectory	55	55
Saved-state interval	0.05	0.006
Burn-in	8 saved intervals	4 saved intervals
Train/validation/test trajectories	500/100/200	500/100/200
Training Reynolds numbers	$Re \in [100, 500]$	$Re \in [100, 500]$
Training boundary data	Periodic	$U_{\text{lid}} \in [0.8, 1.2]$
Training rollout horizon	$H = 4$	$H = 4$
ID evaluation horizons	$r \in \{1, 20, 50\}$	$r \in \{1, 20, 50\}$

Table 5: Dataset generation and evaluation statistics. Spatial resolution refers to the active computational grid, and stored states include the initial state of each trajectory. All methods use identical trajectory splits within each dataset.

and the reduced state projection is

$$\overline{R}_h(z_h; c_h) = z_h - \overline{W}_h^{-1} \overline{D}_h^{\top} \left(\overline{D}_h \overline{W}_h^{-1} \overline{D}_h^{\top} \right)^{\dagger} \left(\overline{D}_h z_h - c_h \right). \tag{23}$$

The current MAC implementation uses the Euclidean metric on the free faces, so $\overline{W}_h = I$. Candidate increments are first assigned homogeneous boundary values and then projected with Eq. (22). State retraction first applies the boundary lifting and then solves Eq. (23). The boundary constraint is therefore imposed by construction, while the remaining divergence residual is controlled by the reduced normal-equation solve.

The exact formulas correspond to a converged undamped solve. The implemented finite-iteration CG solver uses a small damping parameter, and its residual behavior is covered by Proposition 3 and Appendix A.4. The Div values reported for LC are evaluated on the interior MAC control volumes.

C Experimental Details

C.1 Dataset Generation and Statistics

P-NS is generated as forced periodic incompressible flow on $[0, 2\pi]^2$ with $Re \in [100, 500]$. LC is generated on $[0, 1]^2$ using a MAC-grid projection solver with no-slip walls and $U_{\text{lid}} \in [0.8, 1.2]$. Within each dataset, all methods use the same training, validation, and test trajectories. Table 5 summarizes the data generation, discretization, and evaluation settings.

C.2 Constraint configurations and stress protocol

The experiments use two constraint configurations as the organizing axis for the empirical study. This organization follows the central claim of PCM-NO: the learned autoregressive transition should map a solver-defined feasible set to itself. Both datasets are generated from two-dimensional incompressible Navier–Stokes dynamics, but they induce different affine constraint sets. This separation allows us to distinguish divergence preservation from boundary-aware feasibility. The continuous equations below define the data-generating families, while the reported feasibility metrics and the projection–retraction layers are defined using the corresponding discrete solver operators.

P-NS. P-NS denotes periodic Navier–Stokes, instantiated as forced periodic Kolmogorov flow. Let $\Omega = \mathbb{T}^2_{2\pi}$ denote the two-dimensional torus. In continuous form, the P-NS data-generating family is

$$\partial_t u(x,t) + u(x,t)\cdot\nabla u(x,t) = -\nabla p(x,t) + \nu\Delta u(x,t) + f_{\mathrm{K}}(x),$$
$$\nabla\cdot u(x,t) = 0,$$

where $(x,t) \in \Omega \times (0,T]$. Equivalently, when represented on the periodic box $[0,2\pi)^2$, the velocity field satisfies

$$u(x + 2\pi e_i, t) = u(x,t), \qquad i = 1,2.$$

Here $u = (u_x, u_y)$ is the velocity field, p is the pressure, $\nu = 1/Re$ in our nondimensionalized data generation, and f_{K} is a fixed Kolmogorov forcing. At the discrete level, the feasible set is

$$\mathcal{M}_h(0) = \{u_h : D_h u_h = 0\},$$

so the only explicit constraint operator is the periodic spectral divergence D_h. Periodicity is built into the discretization, and no separate BC residual is reported. This configuration provides a clean test of divergence-free rollout, energy-based diagnostics, Reynolds-number extrapolation, and resolution transfer because the spectral projection and the energy diagnostic are defined on the same periodic grid.

LC. LC denotes lid-driven cavity flow. The *lid* is the top wall of the unit-square cavity. The left, right, and bottom walls are stationary no-slip walls, while the top lid moves horizontally with prescribed speed U_{lid}. The continuous data-generating equation is

$$\partial_t u + u\cdot\nabla u = -\nabla p + \nu\Delta u, \qquad \nabla\cdot u = 0, \qquad x \in [0,1]^2, \quad t > 0,$$

with boundary condition $u = (0,0)$ on the left, right, and bottom walls, $u = (U_{\mathrm{lid}}, 0)$ on the top lid. The resulting discrete feasible set is

$$\mathcal{M}_h(b_h) = \{u_h : D_h u_h = 0,\ B_h u_h = g_h\}, \qquad b_h = \begin{bmatrix} 0 \\ g_h \end{bmatrix}.$$

Equivalently, $C_h = [D_h; B_h]$ and $C_h u_h = b_h$. This setting is the main test of the general constraint formulation. It requires a rollout to preserve both incompressibility and prescribed boundary values, and it prevents the method from being interpreted as only a periodic Helmholtz projection. The profile error reported for LC uses the vertical and horizontal centerline velocity profiles, which are more informative for cavity flow than a spectral-energy diagnostic.

Evaluation-only stress diagnostics. The main stress diagnostics reuse the validation-selected checkpoints from the in-distribution experiments. Test data are regenerated under controlled changes in the physical or numerical condition, while the trained model, normalization statistics, backbone hyperparameters, and soft-loss weights are kept fixed. No stress-set tuning or instance-wise test-time fine-tuning is used. This protocol makes the stress results a direct test of out-of-distribution rollout behavior without additional adaptation.

Reynolds-number extrapolation. For P-NS, the training range is $Re \in [100, 500]$ and the stress test uses $Re \in \{700, 1000\}$. For LC, the stress test uses $Re \in \{750, 1000\}$, while the lid velocity remains in the training range $U_{\mathrm{lid}} \in [0.8, 1.2]$. Keeping the LC lid distribution fixed isolates the effect of Reynolds-number extrapolation from a boundary-value shift. Since higher Reynolds numbers correspond to smaller viscosity and stronger small-scale or boundary-layer effects, these tests probe whether the learned transition remains stable beyond the physical-parameter range seen during training.

Boundary-value shift on LC. The lid-shift experiment keeps $Re \in [100, 500]$ and evaluates at $U_{\mathrm{lid}} = 1.5$, outside the training range. This condition changes the right-hand side of the constraint system through $g_h(U_{\mathrm{lid}})$ and therefore changes the feasible set $\mathcal{M}_h(b_h)$ itself. It directly tests whether boundary lifting and boundary-aware retraction continue to impose $B_h\widehat{u}_h \approx g_h$ when the prescribed boundary value differs from those observed in training.

Resolution transfer. The resolution-transfer experiment is run on P-NS by training at 64^2 and evaluating at 128^2. Ground-truth trajectories are generated directly by the 128^2 solver, with no interpolation from coarse trajectories. The discrete operators D_h, P_h, and R_h are reconstructed on the target grid. Because cross-resolution evaluation can conflate resampling with extrapolation to unseen frequencies and expose aliasing [36], we use this setting only to test whether the feasible transition remains meaningful after target-grid reconstruction; we do not interpret it as evidence of general resolution invariance. LC resolution transfer is kept out of the main table because it couples mesh refinement with MAC-grid boundary projection and would add a separate implementation factor to the stress comparison.

Low-data stress. The low-data experiment is reported in the appendix because it changes the training set size. We retrain the methods with fixed fractions of the P-NS training trajectories and keep the validation and test sets unchanged. This diagnostic measures the data-efficiency effect of hard feasible transitions and does not replace the evaluation-only stress tests reported in the main table.

FinalProj within the feasible-transition formulation. FinalProj is the closest structural control and instantiates the direct-state realization in Eq. (10), while PCM-NO instantiates the projected-residual realization. Both return the predicted state to the same solver-defined feasible set, so comparable Div or BC floors control for state-level closure. Differences in $\mathrm{RelL2}_{50}$ and Energy/Profile error therefore quantify the empirical effect of transition parameterization, while Table 4 separately examines tangent projection and retraction under finite constraint solves.

C.3 Hyperparameter Settings

Table 6 lists the settings used in the paper-scale experiments. Architecture and optimizer choices are shared across methods within each dataset, and all method-specific loss weights are fixed before test evaluation.

For P-NS, the PINO residual weight is selected using validation rollouts from

$$\lambda_{\mathrm{pde}} \in \left\{10^{-5}, 3\times10^{-5}, 10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}\right\}.$$

The two strongest candidates are rerun over three seeds $\{0,1,2\}$. Their validation errors are 0.0460 ± 0.0008 for $\lambda_{\mathrm{pde}} = 10^{-5}$ and 0.0465 ± 0.0008 for $\lambda_{\mathrm{pde}} = 3\times10^{-5}$, so we select $\lambda_{\mathrm{pde}} = 10^{-5}$. The LC soft-loss weights are fixed before test evaluation and shared

across runs. PCM-NO uses only the rollout objective in Eq. (12) and introduces no auxiliary loss weight.

For LC, the iterative constraint solve uses 30 CG iterations during training and 120 iterations during evaluation, with evaluation tolerance 10^{-6} and damping 10^{-8}. P-NS uses the closed-form spectral Helmholtz projection implemented with FFTs.

C.4 Model and optimizer

All main experiments instantiate each method with the same four-layer FNO backbone, using width 48, 16 Fourier modes per spatial dimension, and GELU activations. The network predicts the next state directly for FNO, DivReg, PINO, and FinalProj, and predicts a candidate update for PCM-NO. We optimize all models with AdamW using learning rate 7×10^{-4}, weight decay 10^{-4}, and at most 100 epochs. The training batch size is 16 for P-NS and 8 for LC. All methods use autoregressive windows of length $H = 4$, and checkpoints are selected by validation rollout error. The complete training and method-specific loss settings are reported in Table 6.

C.5 Metric Definitions

This subsection defines the metrics used in Table 1, Figure 2, Table 2, and Table 4. For each random seed, a metric is first aggregated over all trajectories in the corresponding test set. The paper then reports the mean and standard deviation of the resulting per-seed values. When a stress setting pools multiple test conditions, such as two extrapolated Reynolds numbers, the per-seed metric is computed over the union of the corresponding test trajectories. Condition-wise results may additionally be reported in the appendix.

Unless otherwise stated, $\|\cdot\|_2$ denotes the grid-weighted discrete Euclidean norm over the relevant state, constraint, or profile degrees of freedom. We use the fixed numerical stabilizer $\epsilon = 10^{-6}$ throughout.

Rollout accuracy. At rollout horizon r, the relative state error is

$$\mathrm{RelL2}_r = \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \frac{\left\| \widehat{u}_{h,r}^{(i)} - u_{h,r}^{(i)} \right\|_2}{\left\| u_{h,r}^{(i)} \right\|_2 + \epsilon}. \tag{24}$$

Here, $u_{h,r}^{(i)}$ and $\widehat{u}_{h,r}^{(i)}$ denote the reference and predicted states, respectively, for test trajectory i. Table 1 reports $r \in \{1, 20, 50\}$, while Table 2 reports $\mathrm{RelL2}_{50}$. Figure 2 plots the same quantity as a function of the rollout horizon.

Constraint feasibility. Feasibility is evaluated using the discrete operators that define the corresponding affine constraint set. The incompressibility metric at rollout horizon r is

$$\mathrm{Div}_r = \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \log_{10} \left(\left\| D_h \widehat{u}_{h,r}^{(i)} \right\|_2 + \epsilon \right), \tag{25}$$

where D_h is the dataset-specific discrete divergence operator.

For LC, the prescribed-boundary residual is

$$\mathrm{BC}_r = \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \log_{10} \left(\left\| B_h \widehat{u}_{h,r}^{(i)} - g_h^{(i)} \right\|_2 + \epsilon \right), \tag{26}$$

where B_h extracts the constrained boundary degrees of freedom and $g_h^{(i)}$ contains the prescribed wall velocities for trajectory i. For P-NS, BC_r is omitted because periodicity is built into the discretization and the explicit constraint is $C_h = D_h$.

Equations (25) and (26) average the trajectory-wise logarithmic residuals, rather than taking the logarithm of an averaged residual. Lower and more negative values therefore indicate stronger constraint satisfaction. The logarithmic representation is used because FinalProj and PCM-NO can approach the numerical projection or retraction floor.

P-NS energy error. For P-NS, the flow-specific diagnostic is the relative kinetic-energy error. At rollout horizon r, it is defined as

$$\mathcal{E}_{\mathrm{eng},r} = \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \frac{\left| E\left(\widehat{u}_{h,r}^{(i)}\right) - E\left(u_{h,r}^{(i)}\right) \right|}{\left| E\left(u_{h,r}^{(i)}\right) \right| + \epsilon}, \tag{27}$$

where the discrete kinetic energy is

$$E(v_h) = \frac{1}{2} \|v_h\|_2^2 . \tag{28}$$

LC centerline profile error. For LC, the flow-specific diagnostic compares the two standard centerline velocity profiles. Let

$$\Gamma_x = \left\{ (0.5, y_j) \right\}_j$$

denote the vertical centerline samples used for the horizontal velocity component u_x, and let

$$\Gamma_y = \left\{ (x_j, 0.5) \right\}_j$$

denote the horizontal centerline samples used for the vertical velocity component u_y. The profile error at rollout horizon r is

$$\mathcal{E}_{\mathrm{prof},r} = \frac{1}{2N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \left(\frac{\left\| \widehat{u}_{h,r,x}^{(i)}\big|_{\Gamma_x} - u_{h,r,x}^{(i)}\big|_{\Gamma_x} \right\|_2}{\left\| u_{h,r,x}^{(i)}\big|_{\Gamma_x} \right\|_2 + \epsilon} + \frac{\left\| \widehat{u}_{h,r,y}^{(i)}\big|_{\Gamma_y} - u_{h,r,y}^{(i)}\big|_{\Gamma_y} \right\|_2}{\left\| u_{h,r,y}^{(i)}\big|_{\Gamma_y} \right\|_2 + \epsilon} \right). \tag{29}$$

Figure 2 plots $\mathcal{E}_{\mathrm{eng},r}$ for P-NS and $\mathcal{E}_{\mathrm{prof},r}$ for LC across rollout horizons. Table 2 reports $\mathcal{E}_{\mathrm{eng},50}$ for P-NS settings and $\mathcal{E}_{\mathrm{prof},50}$ for LC settings. Table 4 additionally reports $\mathcal{E}_{\mathrm{prof},50}$ for the LC component variants.

D Additional Experiment Results

D.1 External Structure-Preserving Baseline

ClawNO is included in the P-NS block of Table 1 as an external structure-preserving baseline [25]. We use its FNO-based clawFNO realization, which predicts a scalar potential ψ_θ and constructs

$$\widehat{u}_{h,k+1} = \begin{bmatrix} \partial_{y,h} \psi_\theta \\ -\partial_{x,h} \psi_\theta \end{bmatrix}, \tag{30}$$

through fixed spectral differentiation. Hence $D_h \widehat{u}_{h,k+1} \approx 0$ up to numerical differentiation error. ClawNO therefore enforces incompressibility through the output parameterization, whereas FinalProj projects a directly predicted state and PCM-NO projects a candidate residual update before advancement and retraction.

Table 6: Hyperparameter settings for the paper-scale experiments. The split sizes are numbers of trajectories. A dash indicates that the corresponding term is absent for that constraint configuration.

Setting	P-NS	LC
Spatial resolution	64×64	64×64
Train/validation/test trajectories	500/100/200	500/100/200
FNO layers / width / Fourier modes	4/48/16	4/48/16
Optimizer	AdamW	AdamW
Learning rate / weight decay	$7 \times 10^{-4}/10^{-4}$	$7 \times 10^{-4}/10^{-4}$
Maximum epochs	100	100
Training / evaluation batch size	16/8	8/8
Training rollout horizon	$H = 4$	$H = 4$
Validation / test rollout horizon	20/50	50/50
Random seeds	$\{0, 1, 2\}$	$\{0, 1, 2\}$
DivReg/PINO divergence weight λ_{div}	0.10	0.05
Boundary penalty weight λ_{bc}	–	0.10
PINO residual weight λ_{pde}	10^{-5}	10^{-2}
Constraint solve	spectral FFT	CG: 30/120 train/eval iterations
Evaluation solve tolerance / damping	–	$10^{-6}/10^{-8}$

Dataset	Batch	Method	Step ms/traj. ↓	50-step ms/traj. ↓	Speedup vs. solver ↑	Cost vs. FNO ↓
P-NS	1	Reference solver	26.70 ± 0.90	1297.33 ± 7.78	1.0×	–
	1	FNO	2.62 ± 0.29	124.37 ± 0.86	10.4×	1.00×
	1	FinalProj	3.13 ± 0.29	147.71 ± 0.73	8.8×	1.19×
	1	PCM-NO	3.63 ± 0.59	170.21 ± 1.07	7.6×	1.37×
	8	Reference solver	3.55 ± 0.12	174.26 ± 2.21	1.0×	–
	8	FNO	0.35 ± 0.04	15.75 ± 0.17	11.1×	1.00×
	8	FinalProj	0.39 ± 0.03	18.73 ± 0.24	9.3×	1.19×
	8	PCM-NO	0.45 ± 0.03	21.53 ± 0.14	8.1×	1.37×
LC	1	Reference solver	639.02 ± 13.57	31806.49 ± 68.21	1.0×	–
	1	FNO	2.69 ± 0.49	120.11 ± 0.56	264.8×	1.00×
	1	FinalProj	53.05 ± 6.06	2512.74 ± 12.68	12.7×	20.92×
	1	PCM-NO	97.80 ± 0.92	4988.19 ± 81.61	6.4×	41.53×
	8	Reference solver	80.27 ± 0.64	4008.23 ± 3.08	1.0×	–
	8	FNO	0.33 ± 0.06	16.16 ± 0.67	248.0×	1.00×
	8	FinalProj	6.40 ± 0.09	316.86 ± 0.26	12.6×	19.61×
	8	PCM-NO	12.54 ± 0.22	618.00 ± 4.39	6.5×	38.24×

Table 7: End-to-end computational cost on an NVIDIA A100-SXM4-40GB GPU. One learned transition and one reference-solver call advance the state over the same saved physical interval. P-NS uses ten pseudo-spectral RK4 micro-steps per saved step, while LC uses ten MAC micro-steps with the data-generation constraint-solve budget. Timings include all inference-time projection and retraction operations and exclude checkpoint loading and file I/O. Speedup is computed against the corresponding reference solver at the same batch size. Values are mean ± standard deviation over repeated synchronized timing runs.

The matched comparison uses the same P-NS splits, 64^2 resolution, Reynolds-number range, training horizon $H = 4$, and seeds $\{0, 1, 2\}$. ClawNO is trained from scratch with matched backbone capacity, while FinalProj and PCM-NO reuse their validation-selected checkpoints. All methods use the same 50-step evaluator; runtime is measured at batch size one. We restrict ClawNO to P-NS because its periodic potential representation does not directly encode the stacked LC constraint $C_h = [D_h; B_h]$.

Table 8 supplements the main ID table with the P-NS kinetic-energy diagnostic and inference cost.

Since all three methods impose P-NS incompressibility structurally, Div_{50} mainly verifies feasibility. The discriminating quantities are RelL2_{50}, $\mathcal{E}_{\text{eng},50}$, and inference cost.

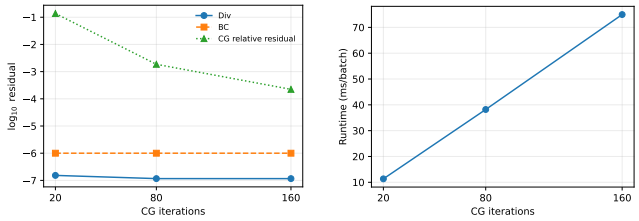
Method	RelL2_{50} ↓	Div_{50} ↓	$\mathcal{E}_{\text{eng},50}$ ↓	Time/step (ms) ↓
ClawNO	0.9270 ± 0.3536	−5.3557 ± 0.0179	1.1820 ± 0.8115	4.15 ± 0.07
FinalProj	0.0503 ± 0.0017	**−5.4981 ± 0.0001**	**0.0080 ± 0.0004**	**3.13 ± 0.29**
PCM-NO	**0.0429 ± 0.0005**	−5.4978 ± 0.0002	0.0086 ± 0.0058	3.63 ± 0.59

Table 8: P-NS comparison with ClawNO. ClawNO parameterizes the velocity field through a learned potential and fixed differentiation, while FinalProj and PCM-NO retain the velocity-space backbone used in the main experiments.

D.2 Constraint-solver diagnostics

P-NS uses the closed-form spectral Helmholtz projection implemented with FFTs, so its feasibility floor is determined by floating-point accuracy rather than an iterative stopping rule. LC uses a MAC-grid normal-equation projection with boundary lifting. We evaluate its numerical behavior separately because the linear-solve budget controls a cost–feasibility tradeoff that is distinct from neural-backbone hyperparameters.

Ground-truth consistency. Projecting held-out LC ground-truth states at rollout step 50 changes them by approximately 1.08×10^{-7} in relative norm. The prescribed-wall residual remains at -6.0000 on the reported $\log_{10}$ scale. Thus the stored solver trajectories already lie close to the same discrete feasible set used by the constraint



(a) Constraint and linear-solve residuals (b) Projection runtime at tolerance 10^{-5}

Figure 5: Numerical sensitivity of the LC projection/retraction solve on step-50 ground-truth states. At a fixed CG iteration budget, tolerances in $\{10^{-3}, 10^{-5}, 10^{-7}\}$ produce the same projected states, showing that the iteration cap is the effective control in this diagnostic.

layer, and the evaluation does not compare predictions against a materially different post-processed target.

Increasing the CG budget from 20 to 80 iterations reduces the interior Div residual from −6.8162 to −6.9341, while 160 iterations yields only a marginal further change to −6.9345. BC remains at −6.0000 throughout. At tolerance 10^{-5}, runtime increases from 11.34 to 38.19 and 74.97 ms per batch for 20, 80, and 160 iterations, respectively. The paper-scale LC evaluation uses 120 iterations, which lies in the saturation regime indicated by the 80- and 160-iteration diagnostics. These results justify using an intermediate-to-high solve budget rather than the highest-cost setting. The observed saturation is consistent with Appendix A.4, where feasibility is controlled by the achieved linear-system residual rather than by a nominal stopping tolerance alone.

D.3 End-to-End Computational Cost

Table 7 compares the complete learned transitions with the corresponding reference solvers on the same GPU and over the same saved physical interval. The measurements include all projection and retraction operations used during inference and exclude checkpoint loading and file I/O. DivReg and PINO share the FNO inference graph because their physical terms are used only during training.

On P-NS, the FFT-based constraint layers introduce moderate overhead: PCM-NO costs approximately 1.37× the FNO rollout while retaining a 7.6× speedup over the reference solver at batch size one. The result is similar at batch size eight. On LC, the iterative constraint solves dominate the learned-model cost. FinalProj, which applies one state retraction per step, costs approximately 21× FNO, while PCM-NO, which applies both tangent projection and state retraction, costs approximately 42× FNO. Nevertheless, the complete LC transition remains approximately 6.4× faster than the reference MAC solver.

These measurements expose a discretization-dependent cost–feasibility trade-off. Spectral constraints can be enforced with relatively small overhead, whereas boundary-aware iterative solves are substantially more expensive. The reported speedups characterize the implementations and hardware used in this study and should not be interpreted as universal comparisons with all optimized CFD solvers. Reducing the LC cost through warm starts, preconditioning, or adaptive solve budgets is an important direction for future work.