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

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.

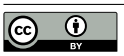
CCS Concepts

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

Keywords

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

*Corresponding author



This work is licensed under a Creative Commons Attribution 4.0 International License.
KDD '27, San Jose, CA, USA
© 2027 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3134-1/2027/08
https://doi.org/10.1145/3862868.3871605

ACM Reference Format:
Nannan Zhou, Guanmeng Xian, and Ning Yang. 2027. Learning Autoregressive Neural Operators on Solver-Defined Physical Constraint Manifolds. In *Proceedings of the 33rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '27), August 01–05, 2027, San Jose, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3862868.3871605

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 show that PCM-NO preserves solver-level feasibility over long rollouts and under parameter, boundary-value, and grid shifts, retains substantial accuracy gains over post-hoc projection at matched divergence residuals, and extends cleanly to 3D staggered MAC constraints.

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.

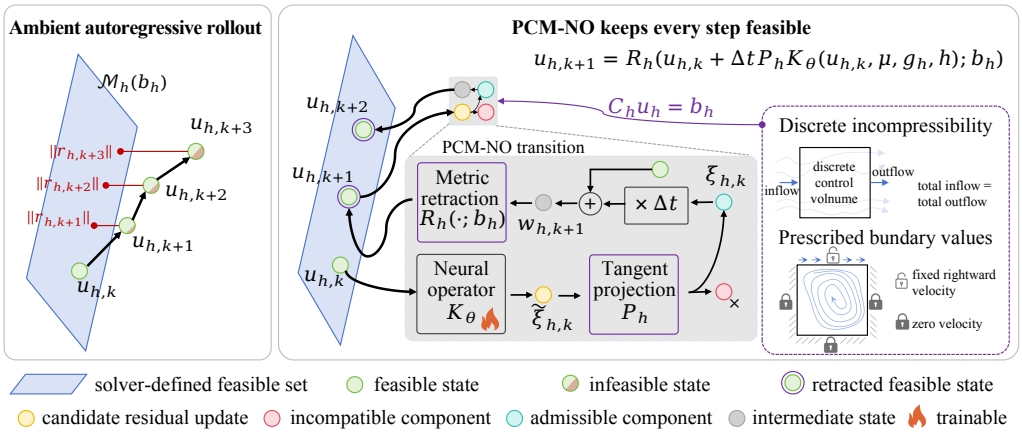


Figure 1: Overview of PCM-NO. An ambient autoregressive rollout can propagate constraint-incompatible components across steps. In PCM-NO, the backbone predicts a candidate update, P_h retains its admissible component, and R_h returns the advanced state to the prescribed affine feasible set. The right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

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

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

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 $\mathcal{U}_h$:

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

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:

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

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 $K_\theta^G(u_h) = \frac{G_\theta(u_h) - u_h}{\Delta t}$. For $u_h \in \mathcal{M}_h(b_h)$, the exact affine projection identity gives

$$\begin{aligned} R_h\Big(G_\theta(u_h); b_h'\Big) &= R_h\Big(u_h + \Delta t\, K_\theta^G(u_h); b_h'\Big) \\ &= R_h\Big(u_h + \Delta t\, P_h K_\theta^G(u_h); b_h'\Big). \end{aligned} \quad (10)$$

Thus, post-hoc state projection is the direct-state realization of the affine feasible-transition principle, while PCM-NO is its projected-residual realization. For a shared residual predictor K_θ, exact affine solves likewise satisfy $R_h(u_h + \Delta t\, P_h K_\theta(u_h); b_h') = R_h(u_h + \Delta t\, K_\theta(u_h); b_h')$, whereas independent multi-step training or finite-budget iterative solves induce distinct transitions.

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 Solver-Defined Constraint Realizations

We instantiate the fixed constraint layers under two representative discretizations, applying the exact same discrete operators across data generation, training-time rollouts, and evaluation.

Periodic spectral projection (P-NS). For periodic flow, $C_h = D_h$ is the spectral divergence operator with $b_h = 0$, and $G_h = -D_h^{*_{W_h}}$ denotes the adjoint spectral gradient under the discrete periodic inner product. Both P_h and $R_h(\cdot; 0)$ reduce to the closed-form Helmholtz projection

$$P_h = I - G_h(D_h G_h)^\dagger D_h, \quad (12)$$

which is evaluated via FFTs while preserving the zero Fourier mode.

Boundary-lifted MAC projection (LC). On a staggered MAC finite-volume grid, $D_h u_h = 0$ enforces interior control-volume incompressibility and $B_h u_h = g_h$ prescribes the wall and lid velocities, yielding $C_h = [D_h; B_h]$ and $b_h = [0; g_h]$. Let $\ell_h(g_h)$ store the prescribed boundary and ghost values, and let $E_h : \mathcal{Z}_h \to \mathcal{X}_h$ inject free-face velocities into the full state so that every boundary-admissible state admits the unique decomposition

$$u_h = \ell_h(g_h) + E_h z_h, \qquad B_h E_h = 0, \qquad B_h \ell_h(g_h) = g_h. \quad (13)$$

Defining the reduced divergence $\overline{D}_h = D_h E_h$ and boundary flux target $c_h(g_h) = -D_h \ell_h(g_h)$ converts $C_h u_h = b_h$ into $\overline{D}_h z_h = c_h(g_h)$ on the free velocity faces. Under the induced metric $\overline{W}_h = E_h^\top W_h E_h$ (set to $\overline{W}_h = I$ in our MAC implementation) with $\overline{Q}_h = \overline{W}_h^{-1}\overline{D}_h^\top(\overline{D}_h\overline{W}_h^{-1}\overline{D}_h^\top)^\dagger$, assigning homogeneous boundary values to candidate updates and applying boundary lifting prior to retraction yield the reduced operators

$$\overline{P}_h = I - \overline{Q}_h\overline{D}_h, \qquad \overline{R}_h(z_h; c_h) = z_h - \overline{Q}_h(\overline{D}_h z_h - c_h). \quad (14)$$

This factorization satisfies the boundary condition by construction and isolates the interior divergence residual to a damped conjugate-gradient (CG) solve on free faces, whose finite-iteration behavior follows Proposition 3 and Appendix A.3.

3.5 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 $\widehat{u}_{h,0}^{(i,s)} = u_{h,s}^{(i)}$, and recursively apply

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

The reported models are trained with

$$\mathcal{L}_{\text{train}}(\theta) = \sum_{i=1}^{n} \sum_{s=0}^{T_i - H} \sum_{r=1}^{H} \omega_r\, \ell_h\Big(\widehat{u}_{h,r}^{(i,s)}, u_{h,s+r}^{(i)}\Big), \quad (15)$$

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*

$$\Phi_\theta(u_{h,k}; \mu, g_h, h) \in \mathcal{M}_h(b_h). \quad (16)$$

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 $\delta u_h^\star \in \mathcal{U}_h$,*

$$\begin{aligned}\|\widetilde{\delta u}_h - \delta u_h^\star\|_{W_h}^2 &= \|P_h\widetilde{\delta u}_h - \delta u_h^\star\|_{W_h}^2 \\ &\quad + \|(I-P_h)\widetilde{\delta u}_h\|_{W_h}^2.\end{aligned} \tag{17}$$

Consequently,

$$\|P_h\widetilde{\delta u}_h - \delta u_h^\star\|_{W_h} \le \|\widetilde{\delta u}_h - \delta u_h^\star\|_{W_h}. \tag{18}$$

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

$$\left\|C_h\widehat{P}_{h,k}v_{h,k}\right\| \le \epsilon_{h,k}^P, \quad \left\|C_h\widehat{R}_{h,k}(w_h; b_{h,k+1}) - b_{h,k+1}\right\| \le \epsilon_{h,k}^R.$$

Here, $\epsilon_{h,k}^P \ge 0$ and $\epsilon_{h,k}^R \ge 0$ 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 $\widehat{u}_{h,k+1}^{\tan} = \widehat{u}_{h,k}^{\tan} + \Delta t\,\widehat{P}_{h,k}v_{h,k}$ satisfies*

$$\|r_{h,k}^{\tan}\| \le \|r_{h,0}^{\tan}\| + \Delta t \sum_{j=0}^{k-1} \epsilon_{h,j}^P. \tag{19}$$

The full transition with a final approximate metric retraction satisfies

$$\|C_h\widehat{u}_{h,k+1}^{\text{full}} - b_{h,k+1}\| \le \epsilon_{h,k}^R. \tag{20}$$

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, or projection solve budget without changing the underlying dataset definitions. Section 3.4 and Appendix B.1 specify the two constraint configurations and stress protocols, with dataset and training statistics summarized in Table 7.

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 C.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 B.2. Main test results are reported as mean ± standard deviation over three seeds. Constraint-solver diagnostics and full batch-scaling timings appear in Appendices C.2 and C.3, while matched-feasibility and scalability evaluations are presented in Section 5.7.

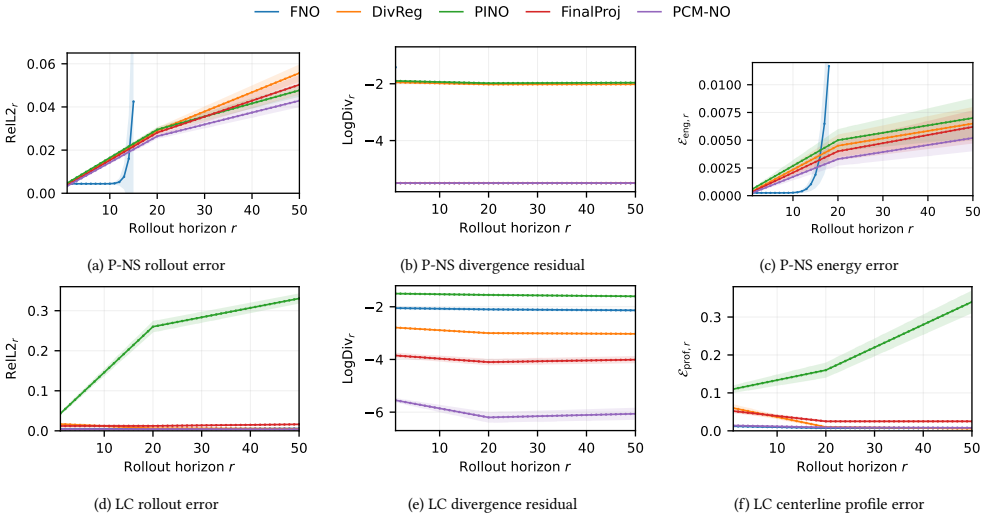
Metrics. Rollout **accuracy** is measured by RelL2_r, where r denotes the rollout horizon. We quantify solver-level **feasibility** using Div_r, the base-10 logarithm of the discrete incompressibility residual. On LC, we additionally report BC_r, defined analogously from the **boundary-condition residual**. 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 B.3.

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

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	$\mathbf{-5.4981 \pm 0.0001}$	–
	PCM-NO	$\mathbf{0.0033 \pm 0.0001}$	$\mathbf{0.0264 \pm 0.0002}$	$\mathbf{0.0429 \pm 0.0005}$	-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	$\mathbf{-6.0000 \pm 0.0000}$
	PCM-NO	$\mathbf{0.0037 \pm 0.0001}$	$\mathbf{0.0043 \pm 0.0002}$	$\mathbf{0.0039 \pm 0.0001}$	$\mathbf{-6.0625 \pm 0.2304}$	$\mathbf{-6.0000 \pm 0.0000}$

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.

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. On the 50-step P-NS kinetic-energy diagnostic, PCM-NO further achieves $\mathcal{E}_{\mathrm{eng},50} = 0.0086 \pm 0.0058$ (comparable to FinalProj at 0.0080 ± 0.0004), whereas ClawNO degrades to 1.1820 ± 0.8115 while requiring 4.15 ± 0.07 ms per step versus 3.63 ± 0.59 ms for PCM-NO (Appendix C.1).

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 achieves the lowest long-horizon state error and competitive energy error 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 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

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.

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 B.1 and B.3.

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

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.

updates satisfy the homogeneous boundary constraint, and the 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 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

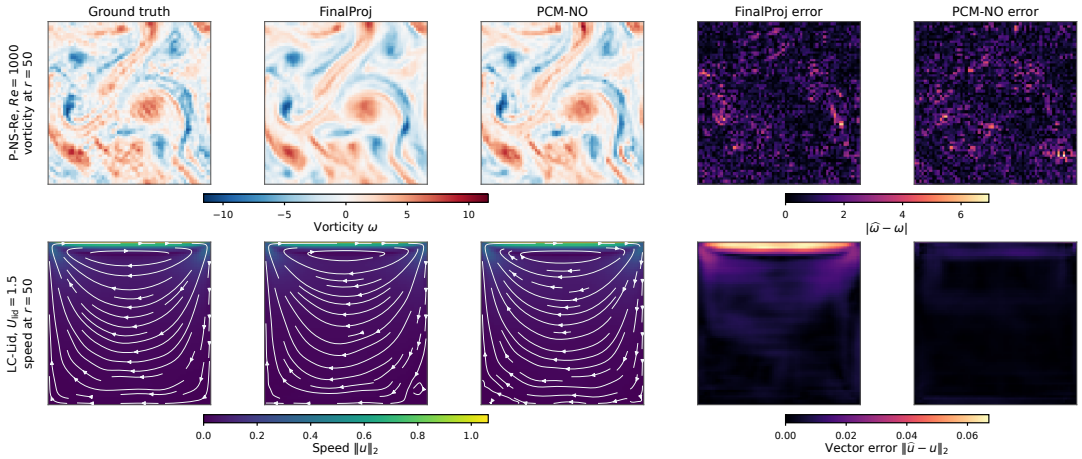
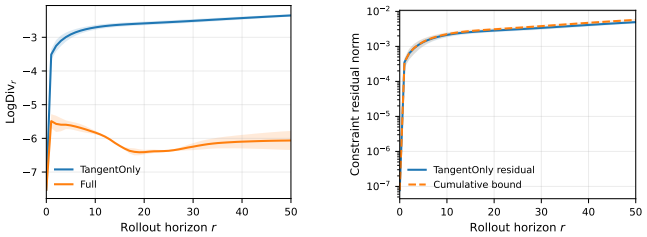


Figure 3: Representative 50-step stress predictions: P-NS Reynolds extrapolation at $Re = 1000$ and the LC lid shift at $U_{\text{lid}} = 1.5$. Rows share field and error color scales, respectively. Trajectories are selected by median ground-truth kinetic energy, independently of model error.



(a) Accumulation versus per-step reset.

(b) Realized residual and cumulative upper bound.

Figure 4: Controlled LC replay with identical projected updates.

unconstrained residual update and therefore recovers much of the feasibility and accuracy. Relative to RetractOnly, PCM-NO reduces RelL2_{50} by 36.1% and $\mathcal{E}_{\text{prof},50}$ by 57.7% while keeping interior divergence at the iterative solve floor. Because exact solves make both residual transitions coincide for a fixed network, this improvement reflects how explicit tangent filtering interacts with finite-budget iterative solves, motivating the matched-feasibility control in Section 5.7. DivOnly removes the prescribed-boundary component from C_h, causing severe boundary violation and degraded profile accuracy. Together, these results confirm the complementary roles of the complete affine constraint, per-step feasibility reset, and projected-update parameterization.

Computational implications. Benchmarking complete learned transitions against the reference solvers on an NVIDIA A100-SXM4-40GB GPU over identical physical intervals (10 solver micro-steps per transition, detailed in Appendix C.3 and Table 8) shows that at batch size $B = 1$, PCM-NO achieves a 7.6× speedup on P-NS (170.21 vs. 1297.33 ms) and a 6.4× speedup on LC (4988.19 vs. 31806.49 ms) over 50-step rollouts, with consistent speedups at $B = 8$. Enforcing spectral constraints on P-NS requires 1.37× the

Variant	RelL2_{50} ↓	Div_{50} ↓	BC_{50} ↓	$\mathcal{E}_{\text{prof},50}$ ↓
TangentOnly	0.0156 ± 0.0021	-1.2424 ± 0.0587	$\mathbf{-6.0000 \pm 0.0000}$	0.0723 ± 0.0129
RetractOnly	$\underline{0.0061 \pm 0.0001}$	-2.7142 ± 0.0288	$\mathbf{-6.0000 \pm 0.0000}$	$\underline{0.0208 \pm 0.0007}$
DivOnly	0.0739 ± 0.0121	$\underline{-2.7606 \pm 0.1050}$	-1.8536 ± 0.0653	0.2075 ± 0.0269
PCM-NO	$\mathbf{0.0039 \pm 0.0001}$	$\mathbf{-6.0625 \pm 0.2304}$	$\mathbf{-6.0000 \pm 0.0000}$	$\mathbf{0.0088 \pm 0.0027}$

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

FNO runtime, whereas the two iterative MAC-grid CG solves on LC require 41.53× FNO (1.99× FinalProj), motivating warm starts, preconditioning, and adaptive solve budgets.

5.7 Matched-Feasibility, Grid-Scaling, and 3D Diagnostics

Matched-feasibility control and 128^2 LC transfer. Because nominal CG stopping rules equalize neither achieved divergence residuals nor solver work between single-solve FinalProj and two-solve PCM-NO, we evaluate frozen 64^2 checkpoints on both 64^2 and reconstructed 128^2 LC MAC operators under relative linear-residual stopping at 10^{-6} (damping 10^{-8}, per-solve CG caps of 480 at 64^2 and 960 at 128^2). Table 5 reports 50-step state error, mean trajectory-wise unstabilized interior divergence RMS q, and batch-one 50-step GPU runtime. Under identical 10^{-6} stopping, PCM-NO attains lower divergence q alongside substantially lower rollout error on both grids. Tightening FinalProj so that mean q matches PCM-NO within 5% barely alters FinalProj accuracy ($0.0145 \rightarrow 0.0142$ on 64^2 and $0.0252 \rightarrow 0.0248$ on 128^2), leaving PCM-NO with 71.8% and 71.0% lower RelL2_{50} at only 19.9% and 9.2% extra runtime. On the 128^2 grid, PCM-NO also delivers a 3.9× end-to-end speedup over the 128^2 reference MAC solver (67429.76 ms).

Standalone 3D MAC operator verification. To verify that our boundary lifted projection–retraction construction scales beyond two dimensions, we instantiate joint interior divergence and balanced normal-boundary flux constraints on 32^3 and 64^3 staggered 3D

LC Grid	Method (Solve Regime)	RelL2$_{50}$ ↓	Div RMS q ↓	50-step (ms) ↓
64^2	FinalProj (tol 10^{-6})	0.0145 ± 0.0018	1.6×10^{-6}	3082.98
	FinalProj (matched-q)	0.0142	8.2×10^{-7}	3435.90
	PCM-NO (tol 10^{-6})	$\mathbf{0.0040 \pm 0.0002}$	8.4×10^{-7}	4120.05
128^2	FinalProj (tol 10^{-6})	0.0252 ± 0.0053	4.4×10^{-6}	14574.44
	FinalProj (matched-q)	0.0248	9.1×10^{-7}	15774.16
	PCM-NO (tol 10^{-6})	$\mathbf{0.0072 \pm 0.0007}$	9.0×10^{-7}	17225.79

Table 5: Controlled LC evaluation across 64^2 and zero-shot 128^2 MAC grids ($B = 1$). Here q denotes unstabilized step-50 interior divergence RMS. The 128^2 reference MAC solver requires 67429.76 ms (3.9× slower than PCM-NO).

3D Grid	Max Div RMS ↓	Normal BC Err. ↓	Proj. (ms) ↓	Retract. (ms) ↓
32^3	3.02×10^{-13}	0	1.68	1.66
64^3	1.31×10^{-12}	0	17.21	18.70

Table 6: Standalone 3D staggered MAC joint-constraint verification over six fields per grid (CPU `float64`, median of 21 warmed single-worker DCT calls).

MAC grids using boundary lifting paired with an undamped discrete cosine transform (DCT) solve in CPU `float64`. Across six synthetic test fields per grid (Table 6), both P_h and R_h satisfy normal boundary conditions to exact machine zero and reduce maximum divergence RMS to $O(10^{-13}$–$10^{-12})$ within 2 ms at 32^3 and 19 ms at 64^3, confirming 3D operator composability.

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. For smooth nonlinear constraints $f_h(u_h) = 0$, the same principle generalizes at regular states by projecting candidate increments onto the linearized Jacobian kernel $\ker J_{f_h}(u_{h,k})$ to cancel first-order constraint drift and applying a local nonlinear metric retraction to correct finite-step manifold curvature, offering a natural path toward state-dependent and geometry-adaptive PDEs.

8 Limitations and Ethical Considerations

The current theoretical guarantees assume fixed linear discrete constraints and nonempty affine feasible sets. While standalone operator diagnostics verify 3D staggered MAC constraints, learned rollout training in this study covers two-dimensional incompressible flows with an FNO backbone, leaving end-to-end 3D rollouts, irregular geometries, and nonlinear constraints for future investigation. 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.

Acknowledgments

This work is supported by National Natural Science Foundation of China under grant 61972270.

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

A.1 Proof of Proposition 1 and Eq. (10)

Let $A_h = C_h W_h^{-1} C_h^\top$. For every $z \in \mathbb{R}^{m_h}$, $z^\top A_h z = \|W_h^{-1/2} C_h^\top z\|_2^2$, which implies $\ker A_h = \ker C_h^\top$. By symmetry of A_h, $\mathrm{range}(A_h) = (\ker A_h)^\perp = (\ker C_h^\top)^\perp = \mathrm{range}(C_h)$. Defining $Q_h = W_h^{-1} C_h^\top A_h^\dagger$, the KKT conditions for Eqs. (7) and (8) yield $P_h v_h = (I - Q_h C_h) v_h$ and $R_h(w_h; b_h) = w_h - Q_h(C_h w_h - b_h)$. Since $C_h v_h \in \mathrm{range}(C_h) = \mathrm{range}(A_h)$, we have $C_h P_h v_h = (I - A_h A_h^\dagger) C_h v_h = 0$. Similarly, nonemptiness of $\mathcal{M}_h(b_h)$ ensures $b_h \in \mathrm{range}(C_h)$ and hence $C_h w_h - b_h \in \mathrm{range}(A_h)$, giving $C_h R_h(w_h; b_h) = C_h w_h - A_h A_h^\dagger (C_h w_h - b_h) = b_h$. When $u_{h,k} \in \mathcal{M}_h(b_h)$, the intermediate state satisfies $C_h(u_{h,k} + \Delta t\, P_h K_\theta(u_{h,k}, \mu, g_h, h)) = b_h$, and $R_h(\cdot; b_{h,k+1})$ maps any intermediate state into $\mathcal{M}_h(b_{h,k+1})$. Induction over k proves Proposition 1.

Furthermore, for any $u_h \in \mathcal{M}_h(b_h)$ and any feasible target b_h', expanding R_h gives

$$R_h(u_h + \Delta t\, v_h; b_h') = u_h + \Delta t\, P_h v_h - Q_h(b_h - b_h') = R_h(u_h + \Delta t\, P_h v_h; b_h'). \tag{21}$$

Setting $v_h = K_\theta^G(u_h)$ proves Eq. (10), and taking $b_h' = b_h$ yields $R_h(u_h + \Delta t\, v_h; b_h) = u_h + \Delta t\, P_h v_h$.

A.2 Proof of Proposition 2

Because P_h is the W_h-orthogonal projection onto $\mathcal{U}_h = \ker C_h$ and $\delta u_h^\star \in \mathcal{U}_h$, the residual $(I - P_h)\widetilde{\delta u}_h$ is W_h-orthogonal to $P_h \widetilde{\delta u}_h - \delta u_h^\star \in \mathcal{U}_h$, so $\langle P_h \widetilde{\delta u}_h - \delta u_h^\star, (I - P_h)\widetilde{\delta u}_h \rangle_{W_h} = 0$. Taking squared W_h-norms of $\widetilde{\delta u}_h - \delta u_h^\star = (P_h \widetilde{\delta u}_h - \delta u_h^\star) + (I - P_h)\widetilde{\delta u}_h$ establishes Eq. (17), and dropping the nonnegative term $\|(I - P_h)\widetilde{\delta u}_h\|_{W_h}^2$ yields the contraction bound.

A.3 Approximate Solves and Proof of Proposition 3

Consider solving a damped normal equation $(A_h + \gamma_h I)\widehat{\lambda} = d_h$ with $\gamma_h \geq 0$ (including $\gamma_h = 0$) up to linear-system residual $\rho_h = (A_h + \gamma_h I)\widehat{\lambda} - d_h$, which satisfies the algebraic identity $d_h - A_h \widehat{\lambda} = \gamma_h \widehat{\lambda} - \rho_h$. For tangent projection, setting $d_{h,k}^P = C_h v_{h,k}$ and $\widehat{P}_{h,k} v_{h,k} = v_{h,k} - W_h^{-1} C_h^\top \widehat{\lambda}_{h,k}^P$ gives $C_h \widehat{P}_{h,k} v_{h,k} = \gamma_h \widehat{\lambda}_{h,k}^P - \rho_{h,k}^P$, which satisfies Assumption 1 with $\epsilon_{h,k}^P = \|\gamma_h \widehat{\lambda}_{h,k}^P - \rho_{h,k}^P\| \leq \|\rho_{h,k}^P\| + \gamma_h \|\widehat{\lambda}_{h,k}^P\|$. For metric retraction, setting $d_{h,k}^R = C_h w_h - b_{h,k+1}$ and $\widehat{R}_{h,k}(w_h; b_{h,k+1}) = w_h - W_h^{-1} C_h^\top \widehat{\lambda}_{h,k}^R$ similarly gives $C_h \widehat{R}_{h,k}(w_h; b_{h,k+1}) - b_{h,k+1} = \gamma_h \widehat{\lambda}_{h,k}^R - \rho_{h,k}^R$, validating $\epsilon_{h,k}^R = \|\gamma_h \widehat{\lambda}_{h,k}^R - \rho_{h,k}^R\|$. Under the tangent-only update, $r_{h,k+1}^{\tan} = r_{h,k}^{\tan} + \Delta t\, C_h \widehat{P}_{h,k} v_{h,k}$. Applying the triangle inequality recursively over $j = 0, \ldots, k-1$ proves Eq. (19), whereas the terminal $\widehat{R}_{h,k}$ step in the full transition directly implies Eq. (20).

Setting	P-NS	LC
Physical domain / grid	$[0, 2\pi]^2$, 64×64 Dealiased pseudo-spectral	$[0, 1]^2$, 64×64 MAC-grid projection
Reference generator	RK4 solver	solver
State representation	Collocated velocity	Staggered MAC velocity
Stored states / transitions	56/55	56/55
Saved interval Δt / burn-in	0.05 (8 intervals)	0.006 (4 intervals)
Train / val / test trajectories	500/100/200	500/100/200
Training Re / boundary data	[100, 500], periodic	[100, 500], $U_{\mathrm{lid}} \in [0.8, 1.2]$
FNO layers / width / modes	4/48/16 (GELU)	4/48/16 (GELU)
Optimizer / lr / weight decay	AdamW, $7e{-4}/10^{-4}$	AdamW, $7e{-4}/10^{-4}$
Max epochs / batch (train, eval)	100 / 16, 8	100 / 8, 8
Horizons (H / val / test)	4/20/50	4/50/50
Weights $\lambda_{\mathrm{div}}/\lambda_{\mathrm{bc}}/\lambda_{\mathrm{pde}}$	$0.10 / - / 10^{-5}$	$0.05 / 0.10 / 10^{-2}$
Constraint solve (train / eval)	Spectral FFT	CG (30/120 iters)
Eval tolerance / CG damping	–	$10^{-6}/10^{-8}$

Table 7: Dataset generation, discretization, and hyperparameter settings across seeds $\{0, 1, 2\}$. Stored states include the initial condition of each trajectory, and all methods share identical splits within each dataset.

B Experimental Details

B.1 Constraint Configurations and Stress Protocols

Both benchmarks are governed by the nondimensionalized 2D incompressible Navier–Stokes equations with viscosity $\nu = 1/Re$:

$$\partial_t u + u \cdot \nabla u = -\nabla p + \nu \Delta u + f, \qquad \nabla \cdot u = 0. \tag{22}$$

P-NS solves forced Kolmogorov flow ($f = f_{\mathrm{K}}(x)$) on the periodic torus $\Omega = [0, 2\pi)^2$, inducing the homogeneous discrete feasible set $\mathcal{M}_h(0) = \{u_h : D_h u_h = 0\}$ with periodicity built into the Fourier basis. **LC** solves unforced lid-driven cavity flow ($f = 0$) on $\Omega = [0, 1]^2$ with stationary no-slip walls $u = (0, 0)$ on the left, right, and bottom boundaries and horizontal lid velocity $u = (U_{\mathrm{lid}}, 0)$ on the top wall, inducing the nonhomogeneous stacked feasible set $\mathcal{M}_h(b_h) = \{u_h : D_h u_h = 0,\ B_h u_h = g_h\}$. Within the feasible-transition family, FinalProj instantiates the direct-state realization in Eq. (10) and matches the state-level feasibility floor, isolating the accuracy gains of the projected-residual parameterization in PCM-NO.

Zero-adaptation stress protocols. All stress evaluations reuse the in-distribution validation-selected checkpoints, normalization statistics, and hyperparameters without test-time adaptation. **P-NS-Re** ($Re \in \{700, 1000\}$) and **LC-Re** ($Re \in \{750, 1000\}$ with $U_{\mathrm{lid}} \in [0.8, 1.2]$) probe lower viscosity and sharper boundary layers while holding boundary distributions fixed. **LC-Lid** fixes $Re \in [100, 500]$ and shifts the top-wall speed to $U_{\mathrm{lid}} = 1.5$, altering $g_h(U_{\mathrm{lid}})$ and thus the affine target $\mathcal{M}_h(b_h)$. **P-NS-Grid** trains at 64^2 and evaluates against direct 128^2-solver trajectories after reconstructing D_h, P_h, and R_h on the 128^2 grid [36], while Section 5.7 evaluates $64^2 \to 128^2$ MAC operator transfer on LC under controlled CG stopping budgets.

B.2 Model and Hyperparameter Settings

Table 7 details the shared architecture and training configurations. FNO, DivReg, PINO, and FinalProj predict next states directly, whereas PCM-NO predicts candidate increments using only the rollout loss in Eq. (15) without auxiliary penalty weights, and checkpoints are

selected by validation rollout error. All soft-constraint weights are fixed prior to testing. On P-NS, validation screening of PINO over $\lambda_{\text{pde}} \in \{10^{-5}, 3\times10^{-5}, 10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$ followed by three-seed reruns of the top two candidates yielded validation errors of 0.0460 ± 0.0008 ($\lambda_{\text{pde}} = 10^{-5}$) and 0.0465 ± 0.0008 ($\lambda_{\text{pde}} = 3\times10^{-5}$), selecting $\lambda_{\text{pde}} = 10^{-5}$.

B.3 Metric Definitions

Each metric is averaged over all N_{test} test trajectories per seed (pooling trajectories across extrapolated parameters in P-NS-Re and LC-Re) and reported as mean ± standard deviation over seeds $\{0, 1, 2\}$. Here, $\|\cdot\|_2$ denotes the grid-weighted discrete Euclidean norm over the relevant degrees of freedom with stabilizer $\epsilon = 10^{-6}$, while the LC divergence diagnostic evaluates RMS interior divergence with logarithmic stabilizer $\epsilon_{\text{div}} = 10^{-12}$ to resolve iterative CG residuals below 10^{-6} (Section 5.7 additionally reports unstabilized RMS q). At horizon r:

- **Rollout accuracy and constraint feasibility:**

$$\begin{aligned}
\text{RelL2}_r &= \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\frac{\|\widehat{u}_{h,r}^{(i)} - u_{h,r}^{(i)}\|_2}{\|u_{h,r}^{(i)}\|_2 + \epsilon},\\
\text{Div}_r &= \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\log_{10}\left(\|D_h\widehat{u}_{h,r}^{(i)}\|_2 + \epsilon_{\text{div}}\right),\\
\text{BC}_r &= \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\log_{10}\left(\|B_h\widehat{u}_{h,r}^{(i)} - g_h^{(i)}\|_2 + \epsilon\right) \quad \text{(LC)}.
\end{aligned} \tag{23}$$

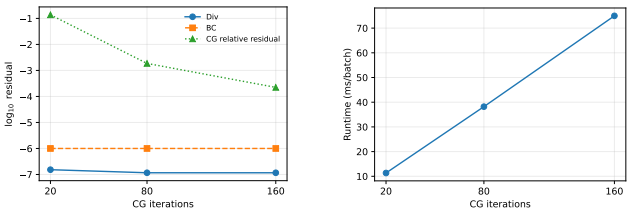
- **Flow-specific diagnostics:** With discrete kinetic energy $E(v_h) = \frac{1}{2}\|v_h\|_2^2$ on P-NS (where $\epsilon_{\text{div}} = \epsilon$) and centerline sample sets $\Gamma_x = \{(0.5, y_j)\}_j$, $\Gamma_y = \{(x_j, 0.5)\}_j$ on LC,

$$\begin{aligned}
\mathcal{E}_{\text{eng},r} &= \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\frac{|E(\widehat{u}_{h,r}^{(i)}) - E(u_{h,r}^{(i)})|}{|E(u_{h,r}^{(i)})| + \epsilon},\\
\mathcal{E}_{\text{prof},r} &= \frac{1}{2N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\left(\frac{\|\widehat{u}_{h,r,x}^{(i)}|_{\Gamma_x} - u_{h,r,x}^{(i)}|_{\Gamma_x}\|_2}{\|u_{h,r,x}^{(i)}|_{\Gamma_x}\|_2 + \epsilon}\right.\\
&\qquad\left. + \frac{\|\widehat{u}_{h,r,y}^{(i)}|_{\Gamma_y} - u_{h,r,y}^{(i)}|_{\Gamma_y}\|_2}{\|u_{h,r,y}^{(i)}|_{\Gamma_y}\|_2 + \epsilon}\right).
\end{aligned} \tag{24}$$

C Additional Diagnostics

C.1 External Structure-Preserving Baseline

ClawNO [25] is evaluated on P-NS via its FNO-based clawFNO realization, which predicts a scalar potential ψ_θ and constructs $\widehat{u}_{h,k+1} = [\partial_{y,h}\psi_\theta, -\partial_{x,h}\psi_\theta]^\top$ through fixed spectral differentiation so that $D_h\widehat{u}_{h,k+1} \approx 0$ up to numerical differentiation precision. We train ClawNO from scratch with matched backbone capacity under identical P-NS splits, 64^2 resolution, $H = 4$, and seeds $\{0, 1, 2\}$, measuring single-step runtime at batch size $B = 1$ (Section 5.2). ClawNO is restricted to P-NS because its periodic potential representation cannot directly encode the nonhomogeneous LC constraint $C_h = [D_h; B_h]$.



(a) Constraint and CG residuals (b) Runtime at tolerance 10^{-5}

Figure 5: Sensitivity of the LC projection/retraction solve on step-50 ground-truth states. Tolerances in $\{10^{-3}, 10^{-5}, 10^{-7}\}$ yield identical states at fixed CG caps.

Set	B	Method	Step (ms) ↓	50-step (ms) ↓	Speedup ↑	vs. FNO ↓
P-NS	1	Ref. 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	Ref. 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	Ref. 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	Ref. 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 8: End-to-end inference cost per trajectory on an NVIDIA A100-40GB GPU (synchronized mean ± std), including all projection and retraction solves. Speedup is relative to the reference solver at matched batch size B.

C.2 Constraint-Solver Diagnostics

While P-NS uses closed-form FFT Helmholtz projection at floating-point precision, LC relies on finite-iteration damped CG solves over free MAC faces. Projecting held-out step-50 LC ground-truth states alters them by 1.08×10^{-7} in relative norm with BC = −6.0000, confirming consistency between stored solver trajectories and the discrete constraint layer.

Figure 5 varies the CG iteration cap on step-50 LC states: raising the budget from 20 to 80 and 160 iterations improves interior Div from −6.8162 to −6.9341 and −6.9345 (with BC = −6.0000 throughout) while increasing runtime from 11.34 to 38.19 and 74.97 ms/batch at tolerance 10^{-5}. Our 120-iteration evaluation budget sits well inside this saturation regime, matching Appendix A.3 where feasibility tracks the achieved linear-system residual.

C.3 End-to-End Runtime Breakdown

Table 8 reports full single-step and 50-step inference timings across batch sizes $B \in \{1, 8\}$ on an NVIDIA A100-40GB GPU (excluding checkpoint and file I/O, with DivReg and PINO sharing the FNO inference graph).