LCKOOP: KOOPMAN-INSPIRED SMOOTH AUTOREGRESSIVE NEURAL OPERATORS IN PDE FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern autoregressive neural operators evolve high-dimensional states through expressive nonlinear transition maps. However, their local first-order behavior varies rapidly along generated trajectories, leading to unstable error propagation. In contrast, Koopman-based models and dynamic mode decomposition (DMD) impose a globally or locally static linear transition, providing more stable representations of state evolution, but are inherently limited in their ability to capture high-dimensional strongly nonlinear dynamics. In this work, we introduce LCKoop (Low-Curvature-is-Koopman), a *plug-in* regularizer that bridges these two frameworks. LCKoop instantiates the model-implied first-order affine approximation and encourages it to remain consistent along generated trajectories without modifying the forecasting architecture or inference path. We show that this regularization controls a trajectory-conditioned higher-order remainder in autoregressive error propagation, complementing conventional first-order sensitivity analysis. Across four PDE forecasting tasks and three neural-operator families, LCKoop reduces MSE by an average of 8.9% at the short forecasting horizon and 14.6% at the extended horizon, with maximum reductions of 53.7% and 42.4%, respectively. The code is available at `https://anonymous.4open.science/r/LCKoop-Low-Curvature-is-Koopman-80E4/`.

1 INTRODUCTION

Auto-regressive neural operators are effective and expressive models for forecasting time-dependent partial differential equations (PDEs) (Kovachki et al., 2023). Despite their expressive power, the high-dimensional and chaotic nature of PDE dynamics can cause models to exhibit unstable predictions (McCabe et al., 2023; Li et al., 2022; Cheng et al., 2025). Understanding and mitigating this instability remains a central challenge, with most existing approaches focusing only on first-order sensitivity through uniform Lipschitz bounds on the transition map, thereby suppressing non-contractive and intermittent transients present in PDEs (McCabe et al., 2023; Oh et al., 2024).

Koopman theory offers a complementary inductive bias by representing nonlinear dynamics through linear evolution in a lifted observable space, often yielding stable and interpretable dynamics (Koopman, 1931; Kutz et al., 2016) at the cost of expressivity. While recent extensions of Koopman models use kernel-based, variational, and probabilistic formulations to enrich the finite-dimensional representation and linear transition operator (Kutz et al., 2016; Bevanda et al., 2023; Wu et al., 2025; Zheng et al., 2025), these approaches focus more on the explicit construction of rigid linear operators, such that accommodating nonlinear dynamics requires a complex combination of observables, kernels, or transition parameterizations.

In this work, we tackle the challenge from the opposite direction: rather than recovering nonlinear expressivity by enriching a finite-dimensional linear model, we start from a highly expressive, physically motivated neural operator and introduce Koopman-style local affinity as a soft regularization bias. Such plug-in design allows the diverse architectural and physical inductive biases of neural operators to be effectively combined with Koopman-inspired local stability.

Building on this viewpoint, we introduce **LCKoop: Low-Curvature-is-Koopman**, a plug-in regularizer that controls how a model's local linearization varies along generated trajectories. LCKoop combines three components: (1) dynamic mode decomposition (DMD) for trajectory-local affine approximation; (2) normalized low-curvature regularization for penalizing deviations from the local

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

first-order approximation (Srinivas et al., 2022); and (3) circular wrapping for exposing the model's latent dynamics as a closed autoregressive transition. Together, these components retain the backbone's capacity to represent strongly nonlinear dynamics while promoting local stability, without introducing an explicit global linear operator or modifying the forecasting architecture.

This plug-in construction places LCKoop between three representative ways of modeling dynamics: global Koopman models, DMD, and neural operators (in Figure 1). LCKoop overcomes the restricted local adaptivity of a globally shared linear transition, the noisy estimation of window-fitted DMD operators, and the potential instability of unconstrained nonlinear transitions in neural operators. We also provide a new theoretical perspective on Koopman-inspired stability for autoregressive neural operators, showing that low curvature bounds the higher-order remainder in autoregressive error propagation, complementing conventional analysis based on first-order sensitivity.

Across four PDE forecasting tasks with high-dimensional sustained transient dynamics, LCKoop generally improves extended-rollout accuracy across spectral, attention-based, and multiscale-convolution neural operators. Across 12 domain-backbone settings, LCKoop reduces the MSE of the target model by an average of 8.9% for short-horizon forecasts and 14.6% for long-horizon forecasts, with gains of up to 53.7% and 42.4%, respectively. Our contributions are as follows:

- LCKoop introduces a Koopman-inspired plug-in regularizer for nonlinear auto-regressive neural operators. Rather than imposing a global linear transition, LCKoop preserves the backbone's expressive dynamics while promoting the consistency of its local affineness along roll-out trajectories.
- We establish a connection between the local linearity underlying DMD models and low-curvature regularization. We theoretically characterize its effect on roll-out error by showing that LCKoop controls higher-order terms, beyond conventional first-order stability control.
- We validate LCKoop across Fourier-, attention-, and UNet-based neural operators on four PDE forecasting tasks, demonstrating improvement in forecasting accuracy while remaining a flexible, architecture-level plug-in.

2 Related Work

Auto-regressive neural operators. Neural operators have become a standard approach to PDE forecasting for their ability to learn mappings between function spaces (Kovachki et al., 2023). At inference, these models are often applied autoregressively: each predicted state becomes the input to the next transition. A small one-step error therefore perturbs all subsequent inputs, causing the learned transition to be evaluated progressively farther from the reference trajectory.

A common stability analysis controls perturbation amplification through a uniform Lipschitz bound (McCabe et al., 2023; Oh et al., 2024). Limiting the maximum Jacobian gain of the transition across the state space provides useful stability guarantees, but may be conservative for PDE systems with legitimate non-contractive regimes and intermittent bursts of rapid evolution (Mikhaeil et al., 2022). Moreover, Jacobian magnitude alone does not fully characterize roll-out behavior. This motivates LCKoop to control the variation of the local linearization rather than only its magnitude.

Koopman and dynamic mode decomposition. Koopman theory represents nonlinear dynamics through a linear operator on a lifted observable space (Koopman, 1931; Rowley et al., 2009). We classify methods by the circled locations in Figure 1. The resulting trade-off between the regions is discussed in detail in Section 3, 4, and Appendix B.

① *Locally adapted affine operators.* Classical DMD, eDMD, and kernel-DMD estimate finite affine transitions from snapshot pairs (Kutz et al., 2016). Recent methods extend the algorithm with kernel or continuous eigen-spectrum approximation approaches (Bevanda et al., 2023; Colbrook et al., 2023; Xu et al., 2025). These DMD methods mostly use non-parametric estimation, offering local adaptability to fit each subtrajectory at the cost of potentially noisy or incomplete operator estimates.

② *Globally parameterized affine operators.* These methods learn an encoder-decoder network with a globally parameterized finite-dimensional affine transition matrix in between (Lusch et al., 2018; Xiong et al., 2024; Bai & Ding, 2025; Cheng et al., 2025; Chen et al., 2026; Cheng et al., 2026), which we call the Koopman matrix. Global parameter sharing improves statistical efficiency and operator consistency across trajectories, but a single affine transition gives limited expressivity.

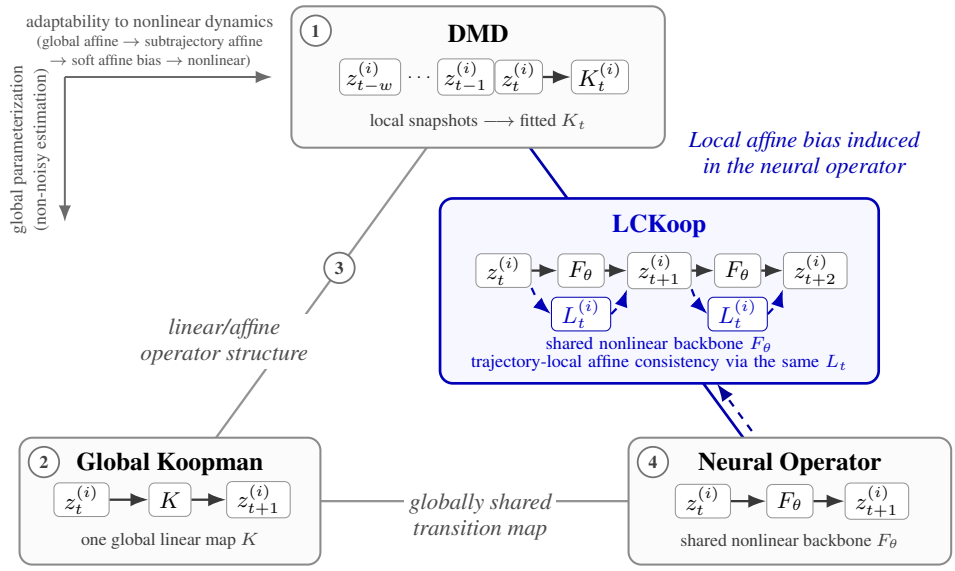


Figure 1: **LCKoop in the Koopman model space.** $L_t^{(i)}$ represents the first-order approximation of the nonlinear operator F_θ at state $z_t^{(i)}$. LCKoop is a plug-in regularizer on the DMD–neural-operator edge: it retains the nonlinear backbone while inducing DMD's local linearity bias by applying $L_t^{(i)}$ throughout the future i-th trajectory. Details are provided in Section 2.

③ *Hybrid methods.* Recent works combine separate global and DMD components (Liu et al., 2023; Wu et al., 2025) or use a global prior to regularize local affine operator adaptation (Zheng et al., 2025; Selim et al., 2025), interpolating between the two paradigms above. LCKoop instead operates at the nonlinear end of this spectrum: it retains a fully nonlinear neural operator and introduces local affinity only as a soft regularization bias, providing more flexibility.

④ *Neural operators.* Unlike the Koopman-based component, neural operators provide the greatest modeling flexibility, enabling expressive data-driven representations. This uncontrolled flexibility, however, can introduce potential instabilities, as discussed above and in Section 4.

3 Preliminaries

3.1 Neural Operators

Neural operators learn mappings between function spaces, commonly approximating PDE solution operators (Kovachki et al., 2023). Let $u_t \in \mathcal{X}$ denote the physical state at time t, where $\mathcal{X}$ is the state space. In auto-regressive forecasting, we learn approximate transition map F_θ that predicts $\hat{u}_{t+1}$ as

$$\hat{u}_{t+1} = F_\theta(u_t), \qquad F_\theta = D_\omega \circ R_\theta \circ E_\psi, \tag{1}$$

where $E_\psi : \mathcal{X} \to \mathcal{Z}$ lifts the physical field to a latent function space, $R_\theta : \mathcal{Z} \to \mathcal{Z}$ updates that representation, and $D_\omega : \mathcal{Z} \to \mathcal{X}$ projects it back to the physical state. Common building blocks are spectral layers, global attention, and multiscale convolutions (Li et al., 2021; Xiong et al., 2024; Tran et al., 2023; Li et al., 2023; Cao, 2021; Wu et al., 2023; Ronneberger et al., 2015; Zhang et al., 2024). Recursively applying F_θ produces multi-step predictions.

3.2 Koopman Operator

For discrete-time dynamics, let $F : \mathcal{X} \to \mathcal{X}$ denote the underlying nonlinear transition map:

$$u_{t+1} = F(u_t). \tag{2}$$

Koopman theory describes the dynamics through the evolution of scalar observables $\psi : \mathcal{X} \to \mathbb{R}$ induced by this transition. Let Ψ be a vector space of observables that is closed under composition

with F. The Koopman operator $\mathcal{K} : \Psi \to \Psi$ is defined to be the composition operator:

$$(\mathcal{K}\psi)(u) = \psi(F(u)), \qquad \psi \in \Psi,\ u \in \mathcal{X}. \tag{3}$$

Linearity. The Koopman operator has the linearity property: even when F is nonlinear,

$$\mathcal{K}(a\psi_1 + b\psi_2) = a\mathcal{K}\psi_1 + b\mathcal{K}\psi_2, \qquad \psi_1, \psi_2 \in \Psi,\ a, b \in \mathbb{R}. \tag{4}$$

Thus, Koopman theory represents nonlinear state dynamics as linear evolution in a generally infinite-dimensional space of observables (Brunton et al., 2022).

3.2.1 Global Koopman operator

In practice, Koopman models seek a finite-dimensional lifting $\psi : \mathcal{X} \to \mathbb{R}^d$ whose components are observables. If the span of ψ is invariant under $\mathcal{K}$, there exists a global transition matrix $A \in \mathbb{R}^{d\times d}$ such that

$$\psi(F(u_t)) = A\psi(u_t). \tag{5}$$

Data-driven models typically learn this lifting using an encoder network $z_t = E(u_t)$ and reconstruct the physical state with a decoder $\hat{u}_t = D(z_t)$. Including an augmented constant coordinate, this yields an affine transition in the lifted space: $z_{t+1} \approx Az_t + b$. The Koopman matrix $K = [A; b]$ is globally parameterized in data-driven models. However, such a fixed finite-dimensional affine transition is limited in approximating strongly nonlinear dynamics by nature.

3.2.2 Dynamic mode decomposition (DMD)

Instead of using a global matrix, dynamic mode decomposition (DMD) estimates a finite-dimensional linear transition operator from a window of consecutive state snapshots using least-squares regression on a dictionary of observables (Kutz et al., 2016), assuming a constant affine matrix within each window. The representative algorithm is affine DMD: given a window $z_{0:w} \in \mathbb{R}^{d\times(w+1)}$ of observables, each adjacent pair $z_i \mapsto z_{i+1}$ is treated as a transition pair. It aims to fit a single affine operator $z \mapsto Az + b$ between collected transition pairs $X = [z_0, \ldots, z_{w-1}]$ and $Y = [z_1, \ldots, z_w]$.

With the augmented input matrix $\overline{X} = [X^\top, 1_w]^\top$, the regression problem gives minimum-Frobenius-norm solution as

$$[\widehat{A}, \widehat{b}] \in \arg\min_{A,b} \left\| Y - AX - b1_w^\top \right\|_{\mathrm{F}}^2 \xrightarrow{\min \|\cdot\|_{\mathrm{F}}} [\widehat{A}\ \widehat{b}] = Y\overline{X}^\dagger, \tag{6}$$

where $(\cdot)^\dagger$ denotes the Moore–Penrose pseudoinverse. The fitted parameters yield the forecast

$$\widehat{z}_{w+1} = \widehat{A}z_w + \widehat{b}. \tag{7}$$

Non-parametric estimation of the operator helps it adapt to arbitrary subtrajectories, at the cost of noisy or ill-conditioned estimates when the context window is short relative to the state dimension. DMD can also be used as a linearization regularizer, but it exhibits two fundamental failure modes as well, analyzed in Appendix B and evaluated in Section 6.2.

Remarks on Koopman Methods. In summary, for high-dimensional dynamics, learning a finite-dimensional global map offers stable but limited adaptability to local trajectories, while DMD offers flexible adaptation yet is typically ill-suited to capture high-dimensional dynamics, with prior works restricted to diagonalizable, low-rank, short-window settings (Wu et al., 2025; Zheng et al., 2025; Chen et al., 2026). LCKoop navigates differently by inducing a soft, trajectory-aware bias reminiscent of DMD within a globally parameterized nonlinear model, promoting local affinity without directly incurring the estimation instability of non-parametric operator fitting.

4 Theory: Curvature as a Surrogate for Koopman Stability

To develop a plug-in regularizer that transfers DMD's stability-promoting inductive bias to nonlinear neural operators, we first identify input-space curvature as a tractable, differentiable surrogate for this inductive bias and show that it controls the coefficient of the higher-order term in autoregressive error propagation. This analysis justifies curvature regularization as a practical surrogate for the resulting stability property. A detailed comparison between curvature-based and DMD regularization is provided in Appendix B.

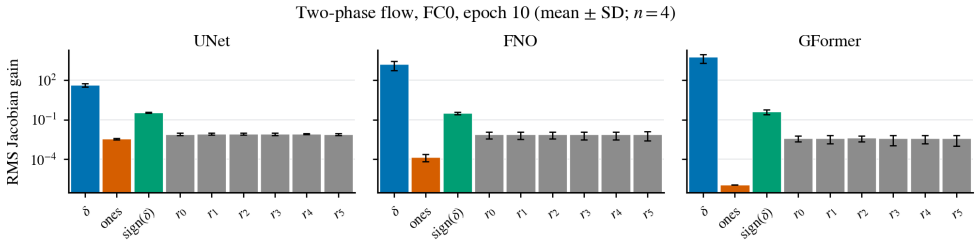


Figure 2: Jacobian gain across multiple δ families at four states z_t. Roll-out displacement directions δ and their sign variants have markedly larger gain than coordinate and random probes r_i, indicating that the roll-out direction captures active local behavior.

4.1 Trajectory-centered directional curvature control

For any state z and displacement δ, input-space curvature controls how rapidly the model-implied local affine transition operator changes from z to $z+\delta$. For a twice-differentiable transition F,

$$\|J_F(z+\delta) - J_F(z)\|_{\mathrm{op}} \le \|\delta\| \sup_{s\in[0,1]} \left\|D_z^2 F(z+s\delta)\right\|_{\mathrm{op}}. \tag{8}$$

As J_F represents the local affine transition, small input-space curvature along neighboring states limits its variation and promotes DMD-like affine-transition consistency. Directly regularizing the full Hessian, however, is impractical in high-dimensional state spaces. Thus, we adopt a probe-based relaxation that measures curvature along selected directions. Motivated by Figure 2, we use roll-out displacements as dynamically active probes and formalize their coverage below.

Assumption 4.1 (Roll-out-direction coverage)**.** *Let $\mathcal{R}$ be a convex neighborhood containing the true and generated roll-outs. For $z \in \mathcal{R}$, let ν_z be the distribution of unit directions induced by nearby roll-out displacements. There exist constants $\kappa \ge 1$ and $G > 0$ such that*

$$\left\|D_z^2 F(z)\right\|_{\mathrm{op}} \le \kappa \left(\mathbb{E}_{v\sim\nu_z} \left\|D_z^2 F(z)[v,v]\right\|^2\right)^{1/2}. \tag{9}$$

Moreover, $0 < \|J_F(z)v\| \le G$ for ν_z-almost every v. Both bounds hold uniformly on $\mathcal{R}$.

Under Assumption 4.1, the readily computable normalized directional curvature sampled along roll-outs controls an upper bound on the input-Hessian norm, making it a suitable regularization target for LCKoop (proof in Appendix A). The following subsection shows that this Hessian bound determines the coefficient of the higher-order term in autoregressive error propagation.

4.2 Reduction of Higher-order roll-out error propagation

Most existing stability analyses focus on first-order sensitivity by bounding the Lipschitz constant or Jacobian norm of the transition map. We instead establish a roll-out error decomposition that isolates the higher-order term in auto-regressive error propagation.

Starting from initial condition $z_0 = z_0^\star$, let the true and learned dynamics be

$$\text{true dynamics: } z_{t+1}^\star = F^\star(z_t^\star), \quad \text{learned dynamics: } \hat{z}_{t+1} = F_\theta(\hat{z}_t).$$

We define the accumulated roll-out state error and one-step error, respectively, as

$$e_t = \hat{z}_t - z_t^\star, \quad \xi_t = F_\theta(z_t^\star) - F^\star(z_t^\star).$$

Theorem 4.2 (Roll-out error decomposition with higher order term)**.** *If F_θ is twice continuously differentiable, then*

$$e_{t+1} = J_{F_\theta}(z_t^\star)e_t + \xi_t + q_t, \tag{10}$$

where

$$q_t = F_\theta(z_t^\star + e_t) - F_\theta(z_t^\star) - J_{F_\theta}(z_t^\star)e_t = C(F_\theta; z_t^\star, e_t). \tag{11}$$

If $\left\|D_z^2 F_\theta(z_t^\star + se_t)\right\|_{\mathrm{op}} \le H$ for every $s \in [0,1]$, then the higher-order error term is bounded as

$$\|q_t\| \le \frac{H}{2}\|e_t\|^2. \tag{12}$$

Under Assumption 4.1, the Hessian norm H is bounded by directional curvature of LCKoop. Hence, reducing this curvature reduces the quadratic higher-order roll-out term q_t.

Corollary 4.3 (Curvature-regulated roll-out bound)**.** *Let*

$$A_t = J_{F_\theta}(z_t^\star), \qquad \Phi(T,s) = A_{T-1}A_{T-2}\cdots A_s, \tag{13}$$

with $\Phi(T,T) = I$*. Then*

$$e_T = \Phi(T,0)e_0 + \sum_{t=0}^{T-1}\Phi(T,t+1)\xi_t + \sum_{t=0}^{T-1}\Phi(T,t+1)q_t. \tag{14}$$

If $\left\|D_z^2 F_\theta\right\|_{\mathrm{op}} \le H$ *along the corresponding roll-out region, then*

$$\|e_T\| \le \|\Phi(T,0)e_0\| + \sum_{t=0}^{T-1}\|\Phi(T,t+1)\xi_t\| + \frac{H}{2}\sum_{t=0}^{T-1}\|\Phi(T,t+1)\|\,\|e_t\|^2. \tag{15}$$

The key idea of plug-in regularization is that an inductive bias from a restricted model class can be transferred to a nonlinear backbone by regularizing a measurable surrogate of its desired consequence. From Corollary 4.3, the roll-out error consists of the initial error e_0, one-step error ξ_t, and higher-order error q_t. We identify the desired consequence of affine DMD as $q_t \approx 0$, which we can softly transfer by regularizing the surrogate trajectory-wise curvature H. By targeting this higher-order remainder without imposing uniform contractivity, such regularization also complements first-order stability controls and avoids the suppression of transient local amplification needed to represent rapidly evolving flow interactions in PDEs. The detailed proof is given in Appendix A.

5 Proposed Method: LCKoop

We now turn the curvature control quantified by H in (12) into a practical plug-in regularizer for nonlinear autoregressive neural operators. We first describe normalized curvature measurement along roll-out directions, then introduce circular wrapping as a novel plug-in technique that exposes latent dynamics as a closed autoregressive transition while preserving the backbone's forecasting architecture and inference path. Implementation pseudocode is provided in Appendix E.5.

5.1 Normalized Curvature Measurement in the roll-out direction

At time t, let z_t denote the lifted state and F_θ the training nonlinear transition map. We instantiate a local affine operator at z_t as

$$L_t(z) = F_\theta(z_t) + J_{F_\theta}(z_t)(z - z_t). \tag{16}$$

For an arbitrary displacement δ_t from z_t, define the error of propagating the affine operator as

$$\begin{aligned} C(F_\theta; z_t, \delta_t) :&= F_\theta(z_t+\delta_t) - L_t(z_t+\delta_t) \\ &= F_\theta(z_t+\delta_t) - F_\theta(z_t) - J_{F_\theta}(z_t)\delta_t. \end{aligned} \tag{17}$$

where the error represents the directional second-order Taylor remainder. A Jacobian–vector product can be used to compute $J_{F_\theta}(z_t)\delta$ without forming the full Jacobian. LCKoop evaluates this remainder along the generated roll-out displacement $\delta_t = F_\theta(z_t) - z_t$. Thus, minimizing it enforces consistency of the local affine operator along the generated trajectory.

The raw Taylor remainder is prone to trivial collapse due to its scale sensitivity; although curvature is the goal, minimizing the remainder also reduces the latent-feature scale and the roll-out displacement. Instead, we use the normalization approach of Srinivas et al. (2022), defining the normalized directional curvature as

$$C_{\mathrm{RN}}(F_\theta; z_t) = \left[\frac{\mathrm{R}(C(F_\theta; z_t', \delta_t'))}{\mathrm{R}(J_{F_\theta}(z_t')\delta_t')\,\mathrm{R}(\delta_t') + \varepsilon}\right]^2, \qquad \mathrm{R}(x) = \sqrt{d^{-1}\,\|x\| + \varepsilon_1}, \tag{18}$$

where $z_t' = \mathrm{detach}(z_t)$, $\delta_t' = F_\theta(z_t') - z_t'$, and d is the vector dimension. The denominator measures the corresponding first-order change and displacement magnitude, matching the quadratic dependence of the Taylor remainder on δ_t'.

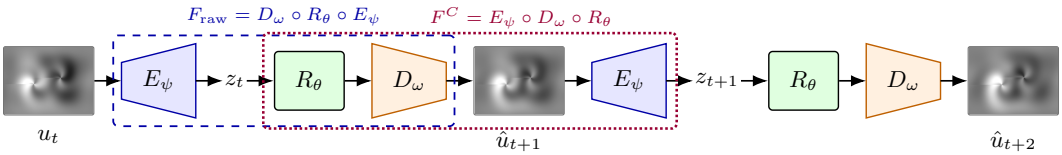


Figure 3: Circular wrapping of an autoregressive neural operator. The dashed box denotes the original forecasting in raw space: $F_{\text{raw}} = D_\omega \circ R_\theta \circ E_\psi$. The dotted box denotes the lifted-space transition induced by circular wrapping $F^C = E_\psi \circ D_\omega \circ R_\theta$, which maps z_t to z_{t+1} through decoding and re-encoding, thereby exposing latent dynamics as a target of regularization.

For an n-step autoregressive rollout $z_{t+1} = F_\theta(z_t)$, we optimize the prediction loss together with the sampled curvature penalty as

$$\mathcal{L}_{\text{total}}(u_{1:n}) = \frac{1}{n}\sum_{t=1}^{n} \ell_{\text{pred}}(\hat{u}_t, u_t) + \lambda_{\text{FC}} \sum_{t=1}^{n} m_t \, C_{\text{RN}}(F_\theta; u_t), \qquad m_t \overset{\text{iid}}{\sim} \text{Bernoulli}(p_{\text{time}}), \tag{19}$$

where $u_{1:n}$ is the physical training trajectory. The coefficient λ_{FC} controls the strength of curvature regularization. Bernoulli sampling speeds up training by stochastically selecting timepoints for curvature regularization.

5.2 Circular wrapping for auto-regressive neural operators

Let u_t denote the raw state. The neural operator F_θ is trained to match the true transition $F^\star$, with a backbone consisting of a lifting encoder, a main operator, and a decoder, schematically written as

$$u_t \xrightarrow{E_\psi} z_t \xrightarrow{R_\theta} h_t \xrightarrow{D_\omega} \hat{u}_{t+1}, \qquad F_{\theta,\text{raw}} = D_\omega \circ R_\theta \circ E_\psi. \tag{20}$$

As shown in Figure 3, we instead linearize in the lifted space while preserving the physical forecasting map. This *circular wrapping* maps $z_t \to z_{t+1}$ through $F^C_{\theta,\text{lifted}} = E_\psi \circ D_\omega \circ R_\theta$. Although any complete latent set can be used, we adopt the representation immediately after the linear lifting to avoid further ambiguity (see Appendix E). By linearizing the latent dynamics, we prevent heterogeneous raw physical channels from dominating and follow the Koopman rationale of seeking approximately linear evolution in a richer observation space; an affine map in raw space cannot represent multiple isolated equilibria, attracting limit cycles, or state-dependent interactions (Brunton et al., 2022).

Thus, after selecting the latent set, we can compute the LCKoop regularization term by sampling the normalized directional curvature on the circularly wrapped transition, as in the right below:

$$\mathcal{L}_{\text{total}}(u_{1:n}) = \frac{1}{n}\sum_{t=1}^{n} \ell_{\text{pred}}(\hat{u}_t, u_t) + \lambda_{\text{FC}} \sum_{t=1}^{n} m_t \, C_{\text{RN}}(F^C_\theta; z_t), \; m_t \overset{\text{iid}}{\sim} \text{Bernoulli}(p_{\text{time}}) \tag{21}$$

where F^C_θ is the circularly wrapped transition and $z_t = E_\psi(u_t)$ is the hidden latent state generated during rollout.

6 Experiments

6.1 Experimental Settings

Benchmark. We evaluate LCKoop on CFDNNEval, a benchmark of fluid-PDE forecasting problems with diverse geometries, physical parameters, and boundary conditions in noiseless simulations.[1] We evaluate all models at 32×32 spatial resolution using a downscaled dataset for each domain. Given one state at time t, models are trained autoregressively to predict the next three states (ms3), where ms$k$ denotes a k-step autoregressive rollout. Validation and testing evaluate both ms3 and ms9. We analyze four domains with nontrivial transient dynamics: Tube, Cavity, Two-Phase Flow / NSCH, and Shallow Waters. Detailed descriptions and hyperparameter settings are in Appendix C.

[1] https://github.com/Lowbcgz/CFDNNEval

Table 1: Percentage change from the unregularized test metric, $\Delta_m = 100(m_0 - m_{\text{LCKoop}})/m_0\%$, for a given metric m. Positive values indicate improvement. Green and red highlights mark improvements and degradations of at least 1%, respectively. Absolute values are given in Appendix G.

		ms3				ms9			
Domain	Backbone	MSE	RMSE	MAE	L2RE	MSE	RMSE	MAE	L2RE
Cavity	GFormer	-6.4%	-10.3%	-5.9%	-4.3%	$+36.5\%$	$+15.0\%$	$+25.1\%$	$+10.1\%$
	UNet	$+0.2\%$	$+1.6\%$	$+1.2\%$	$+1.2\%$	$+10.7\%$	$+6.2\%$	$+4.7\%$	$+6.7\%$
	FNO	$+1.7\%$	$+1.1\%$	$+1.2\%$	$+0.5\%$	-22.4%	-10.9%	-12.3%	-11.6%
Tube	GFormer	$+7.2\%$	$+3.3\%$	$+5.2\%$	$+2.0\%$	$+20.2\%$	$+12.0\%$	$+13.9\%$	$+9.2\%$
	UNet	$+8.3\%$	$+4.1\%$	$+6.1\%$	$+1.1\%$	$+18.7\%$	$+7.0\%$	$+7.3\%$	$+2.8\%$
	FNO	-0.2%	-0.1%	0.0%	-0.1%	$+21.6\%$	$+12.3\%$	$+13.5\%$	$+6.8\%$
Two-phase flow / NSCH	GFormer	$+14.2\%$	$+7.3\%$	$+5.9\%$	$+11.0\%$	$+14.0\%$	$+7.3\%$	$+5.9\%$	$+8.5\%$
	UNet	$+53.7\%$	$+23.7\%$	$+25.7\%$	$+21.2\%$	$+42.4\%$	$+22.0\%$	$+25.0\%$	$+15.1\%$
	FNO	-0.6%	$+0.2\%$	-0.2%	$+3.2\%$	$+1.3\%$	$+0.5\%$	-0.3%	$+3.2\%$
Shallow Waters	GFormer	$+10.3\%$	$+4.1\%$	$+3.7\%$	$+5.4\%$	$+8.7\%$	$+2.8\%$	$+2.3\%$	$+4.0\%$
	UNet	$+4.7\%$	$+2.8\%$	$+2.7\%$	$+3.6\%$	$+5.2\%$	$+3.1\%$	$+3.1\%$	$+3.0\%$
	FNO	$+13.6\%$	$+8.2\%$	$+9.9\%$	$+10.7\%$	$+18.1\%$	$+10.1\%$	$+12.3\%$	$+11.8\%$

Model Backbones. Modern autoregressive neural operators are commonly built from three fundamental architectural paradigms: spectral convolution, attention, and multiscale convolution. We apply LCKoop to representative instances of each paradigm, namely FNO (Li et al., 2021), GFormer (Cao, 2021), and UNet (Ronneberger et al., 2015). The exact circular-wrapping locations are described with full architecture-specific pseudocode in Appendix E. We use Bernoulli sampling with a probability of 15% to skip regularization-loss computation and speed up training, as described in Section 5.

6.2 Main Results

Overall forecasting performance. First of all, we test whether LCKoop improves forecasting across four PDE domains and three neural-operator backbones. Across the 12 domain–backbone pairs, LCKoop reduces MSE by an average of 8.9% at ms3 and 14.6% at ms9, and ms9 MSE improves in almost every pair, with gains of up to 53.7% at ms3 and 42.4% at ms9. Table 1 reports changes in four denormalized field-mean metrics: MSE, RMSE, MAE, and L2RE; complete means, standard deviations, and regularization coefficients are given in Appendix G.

In rare cases, the baseline already exhibits consistent local Jacobians along its trajectories and is close to the Koopman-inspired low-curvature regime, leaving little room for further regularization. In some cases, the model follows the traditional short-term–long-term performance trade-off, where the former represents expressivity and the latter represents stability. This indicates that expressivity may be sacrificed to improve stability by reducing the model's curvature.

Comparison with DMD regularization. We compare LCKoop with the aforementioned DMD forecasting-loss penalty on Cavity-UNet, which is another plug-in regularization, in Table 2. We see that naive DMD regularization increases MSE at both horizons even at small regularization. Full analysis is in Appendices B and G.

Table 2: Regularization comparison on Cavity UNet: MSE ($\times 10^{-4}$, $\downarrow$).

Horizon	Baseline	DMD	LCKoop
ms3	1.819	2.125	**1.816**
ms9	2.402	2.887	**2.146**

Trend of gains across regularization level. We test the trend of gains across various regularization strengths in Figure 4. In Shallow Waters with the FNO backbone, increasing λ_{FC} improves MSE and L2RE at both ms3 and ms9 up to an intermediate strength. The performance increases gradually until $\lambda_{\text{FC}} = 10$, then excessive regularization degrades performance by suppressing curvature needed to represent nonlinear dynamics.

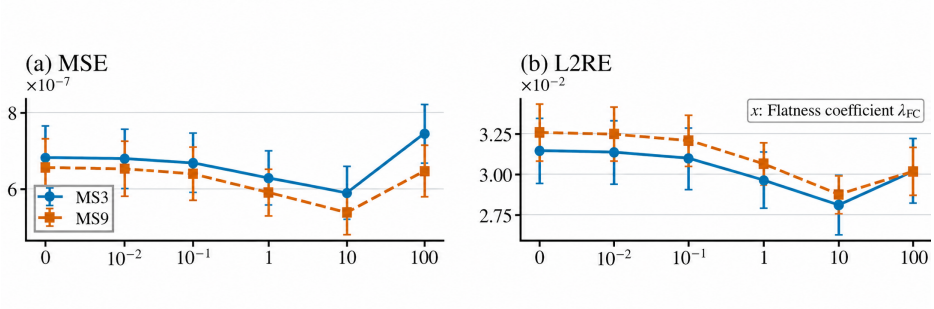


Figure 4: FNO coefficient sweep on Shallow Waters. Increasing the flatness coefficient λ_{FC} improves the metrics up to an intermediate strength; excessive regularization degrades both metrics.

6.3 Additional Study: Robotic Graph-Dynamics Forecasting

To test whether LCKoop transfers beyond PDE forecasting, we evaluate it on the multi-body robotic control benchmarks introduced by Li et al. (2020).[2] These comprise simple typed interacting nodes: Rope, Soft, and Swim. As with CFDNNEval, we train autoregressively on ms2 windows and report test MSE at ms2 and ms6 using checkpoints selected by the corresponding validation horizon. In Table 3, Regularization improves ms6 MSE on all three tasks: by 6.8% on Rope, 5.1% on Soft, and 44.6% on Swim. Thus, the regularizer can substantially improve long-horizon rollouts, sometimes at the cost of the same stability–expressivity trade-off observed in the PDE benchmarks. Detailed setup and full results are provided in Appendix D, E, and G.

Table 3: Percentage change from the FC$=0$ test MSE, $\Delta = 100(m_0 - m_{\mathrm{FC}})/m_0\%$. Positive values indicate improvement. Full results are given in Appendix D.4.

Domain	ms2 MSE	ms6 MSE
Rope	$+6.8\%$	$+6.8\%$
Soft	-18.6%	$+5.1\%$
Swim	$+0.8\%$	$+44.6\%$

7 Conclusion

LCKoop is a plug-in regularizer that penalizes the higher-order deviation of a nonlinear autoregressive transition from its local affine approximation along generated trajectories. Coupled with latent-state exposure through circular wrapping, it provides a practical way to inject Koopman-inspired structure while preserving nonlinear expressiveness. Although the current study is limited to evaluating LCKoop only on chaotic PDE and robotic graph-dynamics autoregressive forecasting tasks, we believe that the proposed method can generalize to other domains. Future work should characterize when this second-order differential bias is most effective beyond chaotic dynamics, how similar biases can be imposed without expensive higher-order derivative computations, how its effectiveness depends on the choice of internal cut and representation, and whether it generalizes to other autoregressive architectures.

[2] https://github.com/YunzhuLi/CompositionalKoopmanOperators

AI USE STATEMENT

In this work, we used generative AI tools to draft and polish portions of the manuscript by expressing author-developed ideas in complete sentences. Most literature discovery was conducted without AI; we used AI selectively to refine search strategies and identify related work that explicitly discusses instability in neural-operator forecasting of chaotic dynamics. The authors read and assessed all identified works before citing them. The main research idea, conceptual framework, experimental design, propositions concerning roll-out error and DMD regularization, and the lemma were developed by the authors. AI assistance in the mathematical analysis was limited to helping formulate the roll-out-direction coverage assumption (Assumption 4.1) and complete parts of the proofs from author-developed arguments. All AI-assisted text, citations, assumptions, mathematical claims, and proofs were subsequently revised and checked by the authors. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The authors have read and adhered to the ICLR Code of Ethics. This work uses simulated PDE and robotic-dynamics benchmarks and does not involve human participants, personal data, or direct deployment in safety-critical decision-making. Although improved physical forecasting may support scientific and engineering applications, models should be independently validated before use in consequential settings.

REPRODUCIBILITY STATEMENT

We report all main results as means and standard deviations over matched sets of four random seeds for the CFDNNEval PDE benchmarks and robotic graph-dynamics study. The anonymized code repository provides all necessary implementation and experimental settings.

REFERENCES

Hanru Bai and Weiyang Ding. KoNODE: Koopman-driven neural ordinary differential equations with evolving parameters for time series analysis. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2484–2519. PMLR, 2025.

Petar Bevanda, Max Beier, Armin Lederer, Stefan Sosnowski, Eyke Hüllermeier, and Sandra Hirche. Koopman kernel regression. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/34678d08b36076de986df95c5bbba92f-Abstract-Conference.html.

Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern Koopman theory for dynamical systems. *SIAM Review*, 64(2):229–340, 2022. doi: 10.1137/21M1401243.

Shuhao Cao. Choose a Transformer: Fourier or Galerkin. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24924–24940, 2021.

Siyuan Chen, Zhecheng Wang, Yixin Chen, Yue Chang, Peter Yichen Chen, Eitan Grinspun, and Jonathan Panuelos. Factorized neural implicit DMD for parametric dynamics. *arXiv preprint arXiv:2603.10995*, 2026. doi: 10.48550/arXiv.2603.10995.

Xiaoyuan Cheng, Yi He, Yiming Yang, Xiao Xue, Sibo Cheng, Daniel Giles, Xiaohang Tang, and Yukun Hu. Learning chaos in a linear way. In *International Conference on Learning Representations*, 2025.

Xiaoyuan Cheng, Wenxuan Yuan, Yiming Yang, Yuanzhao Zhang, Sibo Cheng, Yi He, and Zhuo Sun. Information shapes Koopman representation. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Szh0ELyQxL.

Matthew J. Colbrook, Lorna J. Ayton, and Máté Szőke. Residual dynamic mode decomposition: Robust and verified koopmanism. *Journal of Fluid Mechanics*, 955:A21, 2023. doi: 10.1017/jfm.2022.1052.

Scott T. M. Dawson, Maziar S. Hemati, Matthew O. Williams, and Clarence W. Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57(3):42, 2016. doi: 10.1007/s00348-016-2127-7.

Bernard O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

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

J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016.

Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional Koopman operators for model-based control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1ldzA4tPr.

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

Zongyi Li, Miguel Liu-Schiaffini, Nikola Kovachki, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Learning dissipative dynamics in chaotic systems. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.neurips.cc/paper_files/paper/2022/file/6ad68277e27b42c60ac228c9859fc1a2-Paper-Conference.pdf.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with Koopman predictors. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/28b3dc0970fa4624a63278a4268de997-Abstract-Conference.html.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, 2018. doi: 10.1038/s41467-018-07210-0.

Michael McCabe, Peter Harrington, Shashank Subramanian, and Jed Brown. Towards stability of autoregressive neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Jonas Mikhaeil, Zahra Monfared, and Daniel Durstewitz. On the difficulty of learning chaotic dynamics with RNNs. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-0821.

YongKyung Oh, Dongyoung Lim, and Sungil Kim. Stable neural stochastic differential equations in analyzing irregular time series data. In *International Conference on Learning Representations*, 2024.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241. Springer, 2015.

Clarence W. Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.

Mahmoud Selim, Sriharsha Bhat, and Karl H. Johansson. MetaKoopman: Bayesian meta-learning of Koopman operators for modeling structured dynamics under distribution shifts. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/2a5dd6bad2d10966a8bfa254dfb7dc76-Abstract-Conference.html.

Suraj Srinivas, Kyle Matoba, Himabindu Lakkaraju, and François Fleuret. Efficiently training low-curvature neural networks. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *International Conference on Learning Representations*, 2023.

Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional PDEs with latent spectral models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 37417–37438, 2023.

Xingjian Wu, Xiangfei Qiu, Hongfan Gao, Jilin Hu, Bin Yang, and Chenjuan Guo. K^2VAE: A Koopman-Kalman enhanced variational autoencoder for probabilistic time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 67562–67583. PMLR, 2025.

Wei Xiong, Xiaomeng Huang, Ziyang Zhang, Ruixuan Deng, Pei Sun, and Yang Tian. Koopman neural operator as a mesh-free solver of non-linear partial differential equations. *Journal of Computational Physics*, 513:113194, 2024. doi: 10.1016/j.jcp.2024.113194.

Yuanchao Xu, Kaidi Shao, Nikos K. Logothetis, and Zhongwei Shen. ResKoopNet: Learning Koopman representations for complex dynamics with spectral residuals. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69647–69674. PMLR, 2025.

Xuan Zhang, Jacob Helwig, Yuchao Lin, Yaochen Xie, Cong Fu, Stephan Wojtowytsch, and Shuiwang Ji. SineNet: Learning temporal dynamics in time-dependent partial differential equations. In *International Conference on Learning Representations*, 2024.

Ronghua Zheng, Hanru Bai, and Weiyang Ding. KooNPro: A variance-aware Koopman probabilistic model enhanced by neural process for time series forecasting. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4846257e355f6923fc2a1fbe35099e91-Abstract-Conference.html.

A PROOFS

A.1 PROOF OF DIRECTIONAL CURVATURE IDENTITY

Proposition A.1 (Idealized directional curvature controls the Hessian)**.** *Let F be twice continuously differentiable in a neighborhood of z, let v be a unit vector, and let $\delta = hv$ for $h > 0$. For analysis, consider the idealized, ε-free unsquared normalized remainder*

$$\widetilde{C}_{\mathrm{RN}}(F; z, \delta) = \frac{\|C(F; z, \delta)\|}{\|J_F(z)\delta\| \, \|\delta\|}. \tag{22}$$

If $J_F(z)v \neq 0$, then

$$\lim_{h \to 0} \widetilde{C}_{\mathrm{RN}}(F; z, hv) = \frac{\|D_z^2 F(z)[v, v]\|}{2\|J_F(z)v\|}. \tag{23}$$

Under Assumption 4.1, this identity further gives

$$\|D_z^2 F(z)\|_{\mathrm{op}} \leq 2\kappa G \left(\mathbb{E}_{v \sim \nu_z} \left[\left(\lim_{h \to 0} \widetilde{C}_{\mathrm{RN}}(F; z, hv) \right)^2 \right] \right)^{1/2}. \tag{24}$$

Proof of Proposition A.1. By the second-order Taylor expansion of F at z in the direction v,

$$F(z + hv) = F(z) + hJ_F(z)v + \frac{h^2}{2} D_z^2 F(z)[v, v] + o(h^2). \tag{25}$$

Therefore,

$$C(F; z, hv) = \frac{h^2}{2} D_z^2 F(z)[v, v] + o(h^2). \tag{26}$$

Because $\|v\| = 1$,

$$\|J_F(z)(hv)\| \, \|hv\| = h^2 \|J_F(z)v\|. \tag{27}$$

Dividing the Taylor remainder by this expression and taking $h \to 0$ proves (23). The nonzero-Jacobian condition ensures that the idealized normalization is well defined; the numerical objective uses $\varepsilon > 0$.

Finally, substituting (23) into (9) gives

$$\|D_z^2 F(z)\|_{\mathrm{op}} \leq 2\kappa \left(\mathbb{E}_{v \sim \nu_z} \left[\|J_F(z)v\|^2 \left(\lim_{h \to 0} \widetilde{C}_{\mathrm{RN}}(F; z, hv) \right)^2 \right] \right)^{1/2} \tag{28}$$

$$\leq 2\kappa G \left(\mathbb{E}_{v \sim \nu_z} \left[\left(\lim_{h \to 0} \widetilde{C}_{\mathrm{RN}}(F; z, hv) \right)^2 \right] \right)^{1/2}, \tag{29}$$

which proves Equation (24). □

A.2 PROOF OF ROLL-OUT ERROR RESULTS

Proof of Theorem 4.2. By the definitions of e_t and ξ_t,

$$e_{t+1} = \widehat{z}_{t+1} - z_{t+1}^\star \tag{30}$$

$$= F_\theta(\widehat{z}_t) - F^\star(z_t^\star) \tag{31}$$

$$= F_\theta(z_t^\star + e_t) - F_\theta(z_t^\star) + \xi_t. \tag{32}$$

Adding and subtracting $J_{F_\theta}(z_t^\star)e_t$ in (32) gives

$$e_{t+1} = J_{F_\theta}(z_t^\star)e_t + \xi_t + q_t, \tag{33}$$

where

$$q_t = F_\theta(z_t^\star + e_t) - F_\theta(z_t^\star) - J_{F_\theta}(z_t^\star)e_t = C\big(F_\theta; z_t^\star, e_t\big). \tag{34}$$

This proves the decomposition.

For the bound, define $g : [0,1] \to \mathbb{R}^d$ by $g(s) = F_\theta(z_t^\star + se_t)$. Since F_θ is twice continuously differentiable on the segment, Taylor's theorem with integral remainder yields

$$q_t = \int_0^1 (1-s) D_z^2 F_\theta(z_t^\star + se_t)[e_t, e_t]\, \mathrm{d}s. \tag{35}$$

Hence, if $\left\|D_z^2 F_\theta(z)\right\|_{\mathrm{op}} \le H$ on that segment,

$$\|q_t\| \le \int_0^1 (1-s) \left\|D_z^2 F_\theta(z_t^\star + se_t)\right\|_{\mathrm{op}} \|e_t\|^2 \, \mathrm{d}s \tag{36}$$

$$\le H \|e_t\|^2 \int_0^1 (1-s)\, \mathrm{d}s = \frac{H}{2} \|e_t\|^2 . \tag{37}$$

□

Proof of Corollary 4.3. Set $A_t = J_{F_\theta}(z_t^\star)$. By Theorem 4.2,

$$e_{t+1} = A_t e_t + \xi_t + q_t. \tag{38}$$

Repeated substitution in (38) gives

$$e_T = \Phi(T,0)e_0 + \sum_{t=0}^{T-1} \Phi(T,t+1)\xi_t + \sum_{t=0}^{T-1} \Phi(T,t+1) q_t, \tag{39}$$

where the convention $\Phi(T,T) = I$ handles the final summand. This identity also follows directly by induction on T.

Taking norms and applying the triangle inequality gives

$$\begin{aligned} \|e_T\| \le\ & \|\Phi(T,0)e_0\| + \sum_{t=0}^{T-1} \|\Phi(T,t+1)\xi_t\| \\ & + \sum_{t=0}^{T-1} \|\Phi(T,t+1)\| \, \|q_t\| , \end{aligned} \tag{40}$$

where the norm of $\Phi(T,t+1)$ is the induced operator norm. Applying (12) at every time step yields

$$\|e_T\| \le \|\Phi(T,0)e_0\| + \sum_{t=0}^{T-1} \|\Phi(T,t+1)\xi_t\| + \frac{H}{2} \sum_{t=0}^{T-1} \|\Phi(T,t+1)\| \, \|e_t\|^2 , \tag{41}$$

which is the claimed bound. □

B Limitations of finite-window DMD

As mentioned in Section 3.2.2, DMD can be used as a linearization regularizer through two losses: the reconstruction loss, which is the residual of the least-squares fit, and the forecasting loss, which is the error in predicting future states using the fitted operator. In high-dimensional finite window settings, the reconstruction loss typically degenerates to zero. Thus, we suggest two limitations of forecasting loss, particularly from its nonparametric, trajectory-specific pseudoinverse estimation. In contrast, LCKoop derives its local operator from the Jacobian of a globally trained, parameter-shared model, avoiding the limitations of such pseudoinverse estimation.

Proposition B.1 (Subspace limitation in DMD)**.** *Suppose the true dynamics are given by the affine transition $F(z) = Az + b$, and represent the augmented state and affine operator as*

$$\bar{z}_k = \begin{bmatrix} z_k \\ 1 \end{bmatrix}, \qquad K = [A \quad b], \tag{42}$$

so that $z_{k+1} = K\bar{z}_k$. For a past window $z_0, \ldots, z_w$, with the pair of snapshots

$$X = [\bar{z}_0 \quad \bar{z}_1 \quad \cdots \quad \bar{z}_{w-1}], \qquad Y = [z_1 \quad z_2 \quad \cdots \quad z_w], \tag{43}$$

the DMD algorithm gives $\widehat{K} = YX^\dagger$ and $P = XX^\dagger$, where P is the orthogonal projector onto the augmented past-snapshot subspace. Since $Y = KX$, it follows that $\widehat{K} = KP$. Consequently, for any future augmented state $\bar{z}$, the DMD forecasting regularization loss is $K\bar{z} - \widehat{K}\bar{z} = K(I-P)\bar{z}$.

In short-window DMD, the past snapshots may not span all locally relevant directions. Consequently, the pseudoinverse estimate restricts the learned dynamics to the snapshot subspace, whose dimension is at most w and may be smaller than the intrinsic dimension of the local dynamics. This restriction can yield nonzero forecasting error even when the true dynamics are exactly affine, such that the model cannot capture the full complexity of the underlying system. In contrast, LCKoop incurs zero curvature penalty for affine dynamics because it derives its local operator from the model Jacobian rather than the current snapshot span.

Proposition B.2 (Finite-window noise sensitivity of DMD)**.** *Following the analysis of DMD in (Dawson et al., 2016), consider a clean trajectory segment $\{\bar{z}_t\}_{t=0}^{w} \subset \mathbb{R}^{d+1}$. Define the snapshot pair $X = [\bar{z}_0, \dots, \bar{z}_{w-1}] \in \mathbb{R}^{(d+1)\times w}$ and $Y = [z_1, \dots, z_w] \in \mathbb{R}^{d\times w}$.*

Let $X = U\Sigma V^\top$ be the compact SVD of X, where Σ contains all positive singular values. The clean DMD estimate is given below, with its Moore–Penrose pseudoinverse:

$$\widehat{K} = YX^\dagger, \qquad X^\dagger = V\Sigma^{-1}U^\top. \tag{44}$$

Consider randomly perturbed output snapshots $\widetilde{Y} = Y + E$, where $E_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0, \tau^2)$ and E is independent of X, and let $\widetilde{K} = \widetilde{Y}X^\dagger$. Conditional on X,

$$\mathbb{E}\Big[\|\widetilde{K} - \widehat{K}\|_{\mathrm{F}}^2 \,\Big|\, X\Big] = d\tau^2\|X^\dagger\|_{\mathrm{F}}^2 = d\tau^2 \sum_{\sigma_i(X)>0} \frac{1}{\sigma_i(X)^2}. \tag{45}$$

Short windows typically contain highly correlated snapshots, producing small nonzero singular values and thus stronger noise amplification.

Proof. The difference satisfies $\widetilde{K} - \widehat{K} = EX^\dagger$. Conditional on X, each row $e_j^\top$ of E satisfies $e_j \sim \mathcal{N}(0, \tau^2 I_w)$, and therefore

$$\mathbb{E}\big[\|e_j^\top X^\dagger\|_2^2 \,\big|\, X\big] = \tau^2 \operatorname{tr}\big(X^\dagger (X^\dagger)^\top\big) = \tau^2\|X^\dagger\|_{\mathrm{F}}^2.$$

Summing over the d rows and using $\|X^\dagger\|_{\mathrm{F}}^2 = \|\Sigma^{-1}\|_{\mathrm{F}}^2 = \sum_{\sigma_i(X)>0} \sigma_i(X)^{-2}$ proves (45). □

The preceding proposition illustrates a weakness of DMD: nonparametrically estimating a separate operator from each finite trajectory window can make it vulnerable to noise when few snapshot pairs are available. In contrast, LCKoop derives its local operators from a globally parameterized transition model shared across training trajectories, reducing its reliance on any single noisy window.

Existing Remedies.

Classical methods use truncated SVD to reduce the effective state dimension, which can mitigate noise sensitivity but may also discard relevant dynamics. Modern strategies typically use hybrid methods, in which global and local operators are combined to reduce DMD's noise sensitivity (Liu et al., 2023; Wu et al., 2025), or a global prior is placed on local-operator adaptation (Zheng et al., 2025; Selim et al., 2025). LCKoop instead combines DMD's subtrajectory affineness with a global nonlinear transition model, balancing the stability of global parameter sharing with the adaptability of a softer affine constraint.

Experiments.

Our setting is already an extreme case for DMD, in which the DMD reconstruction loss is zero whereas its forecasting loss remains nonzero because of the computational cost and autonomous structure of the neural operator. The resulting trajectory-specific regularization is therefore noisy and excessive. Table 4 compares DMD regularization with the matched canonical $\mathrm{FC} = 0$ baseline for the Cavity UNet `k6f12` model. Even a modest DMD coefficient of 10^{-2} degrades every reported metric at both horizons, while increasing the coefficient to 10^{-1} yields severe instability and large variation. In contrast to LCKoop, these results show that finite-window DMD regularization can impose an unreliable local constraint when the fitted operator is estimated from short, noisy trajectory windows.

Table 4: Cavity UNet `k6f12`: denormalized test field-mean metrics for DMD regularization, averaged over four seeds. Results show high regularization instability, compared to LCKoop.

FC	Test	MSE	RMSE	MAE	L2RE
0	ms3	$1.8187 \cdot 10^{-4}$ $\pm 2.0000 \cdot 10^{-5}$	$1.1420 \cdot 10^{-2}$ $\pm 5.3000 \cdot 10^{-4}$	$5.7900 \cdot 10^{-3}$ $\pm 3.5000 \cdot 10^{-4}$	$3.5286 \cdot 10^{-1}$ $\pm 1.9000 \cdot 10^{-2}$
DMD 10^{-2}	ms3	$2.1254 \cdot 10^{-4}$ $\pm 4.1300 \cdot 10^{-5}$	$1.2270 \cdot 10^{-2}$ $\pm 1.0900 \cdot 10^{-3}$	$6.4100 \cdot 10^{-3}$ $\pm 6.4000 \cdot 10^{-4}$	$3.7696 \cdot 10^{-1}$ $\pm 3.8020 \cdot 10^{-2}$
DMD 10^{-1}	ms3	$3.1294 \cdot 10^{-3}$ $\pm 4.9600 \cdot 10^{-3}$	$3.4390 \cdot 10^{-2}$ $\pm 2.6040 \cdot 10^{-2}$	$2.4130 \cdot 10^{-2}$ $\pm 2.4910 \cdot 10^{-2}$	$1.3911 \cdot 10^{0}$ $\pm 1.5834 \cdot 10^{0}$
0	ms9	$2.4023 \cdot 10^{-4}$ $\pm 4.0700 \cdot 10^{-5}$	$1.3320 \cdot 10^{-2}$ $\pm 9.5000 \cdot 10^{-4}$	$6.6200 \cdot 10^{-3}$ $\pm 5.8000 \cdot 10^{-4}$	$3.8065 \cdot 10^{-1}$ $\pm 2.2670 \cdot 10^{-2}$
DMD 10^{-2}	ms9	$2.8871 \cdot 10^{-4}$ $\pm 5.8800 \cdot 10^{-5}$	$1.4390 \cdot 10^{-2}$ $\pm 1.4500 \cdot 10^{-3}$	$7.2400 \cdot 10^{-3}$ $\pm 8.2000 \cdot 10^{-4}$	$4.0152 \cdot 10^{-1}$ $\pm 4.7410 \cdot 10^{-2}$
DMD 10^{-1}	ms9	$1.1861 \cdot 10^{-2}$ $\pm 2.2130 \cdot 10^{-2}$	$5.5790 \cdot 10^{-2}$ $\pm 6.3140 \cdot 10^{-2}$	$3.6330 \cdot 10^{-2}$ $\pm 4.5680 \cdot 10^{-2}$	$2.6531 \cdot 10^{0}$ $\pm 4.0627 \cdot 10^{0}$

C Detailed Main Experimental Setup

C.1 Detailed benchmark setting

The original CFDNNEval forecasting setting represents each physical state at 64×64 resolution and predicts the next three states from a single input state. For all reported experiments, we use 32×32 states to place greater emphasis on learning temporal transition dynamics than on high-resolution spatial reconstruction while reducing computational cost. We nevertheless retain the backbone capacity used at the original resolution, as experiments showed that it outperformed proportionally reduced configurations.

Koopman-inspired model benchmark results indicate that structurally constrained Koopman-inspired models can remain competitive despite their lower transition expressivity, suggesting that their inductive bias may be particularly useful when data coverage is limited (Xiong et al., 2024; Cheng et al., 2025; Chen et al., 2026). Accordingly, we downscale each domain relative to conventional benchmarks, following a similar procedure.

C.2 Domains and preprocessing

The four two-dimensional CFDNNEval domains retain nontrivial transient dynamics over the forecasting interval:

- **Tube** models velocity-field evolution in tube flow, with varied inlet, fluid, and geometric conditions.
- **Cavity** is lid-driven flow in a closed cavity. The upper-wall motion drives a dominant recirculation region and smaller corner structures. We predict horizontal velocity, vertical velocity, and pressure.
- **Two-Phase Flow / NSCH** models coupled phase-field and velocity evolution in two-phase flow under varying physical parameters.
- **Shallow Waters** is a two-dimensional shallow-water system with reflective boundary conditions. The predicted state contains fluid height and two depth-averaged velocity components.

Each physical variable is normalized independently using min–max statistics computed from the complete training data. The normalization and preprocessing steps follow the CFDNNEval benchmark, except for the Cavity pressure channel, whose default min–max normalization is sensitive to outliers. For this channel, we use robust min–max normalization with the difference between the 90th and 10th training-data percentiles as the denominator. Following the benchmark preprocessing, trajectory segments after numerical convergence are removed according to a predefined convergence threshold so that training focuses on transient dynamics.

C.3 Backbones Description

We apply LCKoop to one representative from each common neural-operator family:

- **FNO:** a spectral operator combining truncated Fourier convolution with pointwise channel mixing. We place the circular-wrapping cut after the lifting layer and before the Fourier evolution blocks.
- **GFormer:** an attention-based operator that evolves embedded grid features through global interactions. We place the cut after the initial feature embedding and before the main attention blocks.
- **UNet:** a multiscale convolutional operator with encoder, decoder, and skip connections. We place the cut after the first full-resolution encoder block, immediately before the first pooling operation; this activation also supplies the level-zero skip connection.

The model takes the image and static conditioning variables as input and outputs the next state. The conditioning variables are treated as global variables that remain fixed throughout the rollout. The exact cut locations are described with full architecture-specific pseudocode in Appendix E.

C.4 Evaluation metrics

All reported values are field-mean test metrics rather than global normalized-space metrics. For sample s and physical channel c, spatial positions and roll-out steps are flattened into vectors $\hat{x}_{s,c}, x_{s,c} \in \mathbb{R}^N$, with $e_{s,c} = \hat{x}_{s,c} - x_{s,c}$. Each metric is computed for a sample–channel pair and then averaged with equal weight using $\langle g_{s,c} \rangle_{s,c} = (SC)^{-1} \sum_{s,c} g_{s,c}$.

Table 5: Field-mean evaluation metrics.

Metric	Equation	Interpretation
MSE	$\left\langle \frac{1}{N} \lVert e_{s,c} \rVert_2^2 \right\rangle_{s,c}$	Quadratically penalizes large errors.
RMSE	$\left\langle \sqrt{\frac{1}{N} \lVert e_{s,c} \rVert_2^2} \right\rangle_{s,c}$	Has physical units; not generally $\sqrt{\mathrm{MSE}}$.
MAE	$\left\langle \frac{1}{N} \lVert e_{s,c} \rVert_1 \right\rangle_{s,c}$	Less sensitive to isolated large errors.
L2RE	$\left\langle \frac{\lVert e_{s,c} \rVert_2}{\max(\lVert x_{s,c} \rVert_2, \sqrt{\varepsilon})} \right\rangle_{s,c}$	Dimensionless; gives each channel equal weight.

For all reported experiments, predictions and targets are converted back to physical scale before these metrics are evaluated. Each run therefore produces one test value per metric, and the means and standard deviations in the four-seed tables are computed across the four resulting per-run values.

C.5 HYPERPARAMETER TABLE

We replace the nonsmooth ReLU activation function with the smooth SiLU activation to avoid sparse regularization losses. We otherwise retain the predefined CFDNNEval hyperparameter configurations across domains. Batch normalization is removed to avoid coupling the independent regularization of sub-trajectories; it also provided no measurable performance benefit in preliminary experiments. The shared optimization and LCKoop settings are summarized in Table 6.

Table 6: Experimental hyperparameters used in all reported experiments. Panels (a)–(c) summarize the backbone and data configurations, while panel (d) lists the shared optimization and LCKoop settings.

(a) Domain-specific backbone configurations. GFormer tuples list hidden width h, layers L, attention heads H, and FFN width; frequency and modes are fixed to 32 and 12. FNO tuples list width, Fourier modes, and padding. UNet uses `k6f12` (kernel size 6 with 12 base feature channels) in every domain. Both boundary settings use raw (x, y) coordinates: `dirichlet` sets the outer prediction to zero, while `null` applies no boundary enforcement.

Domain	GFormer (h, L, H, FFN)	FNO $(w, \mathrm{modes}, \mathrm{pad})$	UNet	Boundary
Cavity	$(128, 6, 2, 128)$	$(32, 12{\times}12, 2)$	`k6f12`	`dirichlet`
Tube	$(48, 4, 2, 96)$	$(32, 12{\times}12, 2)$	`k6f12`	`dirichlet`
Two-Phase Flow	$(96, 6, 4, 256)$	$(32, 12{\times}12, 2)$	`k6f12`	`dirichlet`
Shallow Waters	$(128, 6, 2, 128)$	$(32, 12{\times}12, 2)$	`k6f12`	`null`

(b) Shared backbone structure. All backbones use SiLU.

Backbone	Other details
GFormer	Galerkin attention; IFFT2 decoder; 2 regressors
FNO	Four spectral blocks; linear state lift
UNet	No BatchNorm; average pooling; widths $12/24/48/96/192$

(c) Shared data configuration used across all domains.

Setting	Value
Image resolution	32×32
Dataset setting	4k
Case selection	Deterministic uniform subsampling
Training horizon	ms3
Validation/test horizons	ms3 and ms9

(d) Shared optimization and LCKoop settings. The flatness penalty is evaluated at sampled rollout times over a horizon of three.

Parameter	Value
Optimizer	Adam
Learning rate / weight decay	10^{-3} / 10^{-4}
Scheduler	OneCycleLR ($\max \mathrm{LR} = 10^{-3}$)
Gradient clipping	1.0
Sampling probability	15%
Flatness horizon	3
ε	10^{-2}

D Detailed Setup for Additional Study: Robotic Graph-Dynamics Forecasting

D.1 Detailed benchmark setting

We use the Rope, Soft, and Swim multi-body control benchmarks of Li et al. (2020), following its public repository structure with a smaller-data protocol.[3] Each dataset uses 40/10/10 training, validation, and test trajectories, yielding 3,960 MS2 training windows (40×99). Models are trained autoregressively on MS2; tested on MS2 and MS6, with a checkpoint selected separately for each validation horizon.

D.2 Domains and preprocessing

The Rope, Soft, and Swim domains are simple simulated multi-body systems. Node types encode physical roles or materials (e.g., root/child or actuator/soft/rigid/fixed), whereas relation types encode directed interaction structure (e.g., spring connectivity or relative box position). We normalize each state channel with dataset-level z-score statistics. Results are test field-mean MSE, reported as mean $\pm$ sample standard deviation across four seeds.

- **Rope** represents a deformable rope with one root (anchor) mass node and child mass nodes. Every node state is $(x, y, \dot{x}, \dot{y})$; the one-dimensional horizontal control force is applied only to the root. Its eight directed relation types distinguish root–root and child–child self edges, root–child and child–root one-hop spring edges, and the corresponding two-hop (ghost-spring) edges.
- **Soft** represents a swing-task soft robot with soft-actuator, soft, rigid, and fixed box nodes. Each box state contains positions and velocities of its four corners (16 dimensions), and a scalar control expands or contracts each actuator box. Its $9 \times 4 = 36$ directed relation types combine the receiving box type with one of nine relative-position classes: self, four cardinal offsets, or four diagonal offsets.
- **Swim** represents a swimming soft robot with soft-actuator, soft, and rigid box nodes. It uses the same 16-dimensional box state and scalar actuator control as Soft, with $9 \times 3 = 27$ relation types defined by receiving box type and the same nine relative-position classes. During actuator contraction, the simulator also applies a quadratic, velocity-dependent hydrodynamic force to exposed actuator-box edges, producing propulsion.

D.3 Controlled cyclic transition and loss

The graph benchmarks differ from the PDE backbones in two relevant ways: the decoder predicts a residual that is added to the physical state, and the supplied forcing is time-varying. We therefore regularize the complete rollout on the augmented state while holding the observed forcing fixed.

Let u_t denote the physical node state and f_t the supplied per-node forcing/control. Node attributes and typed relations form static graph context and are suppressed below. After the GNN encoder, $z_t = E_\psi(u_t)$, and the circular state must include the residual path's physical input: $q_t = [u_t, z_t]$. With f_t fixed, the controlled circular map is

$$F^C_{\theta,\mathrm{graph}}(q_t; f_t) = [\widehat{u}_{t+1},\ E_\psi(\widehat{u}_{t+1})]\,, \qquad \widehat{u}_{t+1} = u_t + D_\omega(T_\theta(z_t, f_t))\,. \tag{46}$$

The concatenation closes the entire rollout cycle. For curvature regularization, the same observed f_t is used in the base evaluation, perturbed evaluation, and JVP, so derivatives are taken only with respect to q_t:

$$\begin{gathered}\mathcal{L}_{\mathrm{flat},t} = \left\| F^C_{\theta,\mathrm{graph}}(q'_t + \delta_t; f_t) - F^C_{\theta,\mathrm{graph}}(q'_t; f_t) - J_q F^C_{\theta,\mathrm{graph}}(q'_t; f_t)\delta_t \right\|_2^2 \\ \delta_t = F^C_{\theta,\mathrm{graph}}(q'_t; f_t) - q'_t, \qquad q'_t = \mathrm{sg}(q_t).\end{gathered} \tag{47}$$

This targets recursively accumulated state error while leaving legitimate nonlinear forcing dependence unconstrained. The corresponding forward pass and plug-in training procedure are given in Algorithms 5 and 6.

[3] https://github.com/YunzhuLi/CompositionalKoopmanOperators

D.4 FULL FC SWEEP

Table 7 reports the full FC sweep for the robotic graph-dynamics study. Each entry is test field-mean MSE, averaged over four seeds; MS2 and MS6 use independently selected checkpoints from their matching validation horizons.

Table 7: Robotic graph-dynamics study: test field-mean MSE across all FC values.

Rope			Soft			Swim		
FC	MS2 MSE	MS6 MSE	FC	MS2 MSE	MS6 MSE	FC	MS2 MSE	MS6 MSE
0	$3.2254 \cdot 10^{-3} \pm 1.2632 \cdot 10^{-4}$	$1.1132 \cdot 10^{-2} \pm 8.2728 \cdot 10^{-4}$	0	$1.8027 \cdot 10^{-2} \pm 3.8368 \cdot 10^{-4}$	$5.4075 \cdot 10^{-2} \pm 2.5387 \cdot 10^{-4}$	0	$3.1324 \cdot 10^{-2} \pm 3.5274 \cdot 10^{-4}$	$1.8238 \cdot 10^{-1} \pm 6.0946 \cdot 10^{-2}$
10^{-2}	$3.1946 \cdot 10^{-3} \pm 1.2286 \cdot 10^{-4}$	$1.1018 \cdot 10^{-2} \pm 7.5275 \cdot 10^{-4}$	10^{-1}	$1.7987 \cdot 10^{-2} \pm 7.3440 \cdot 10^{-4}$	$5.3031 \cdot 10^{-2} \pm 1.0702 \cdot 10^{-3}$	10^{-1}	$3.1544 \cdot 10^{-2} \pm 6.9000 \cdot 10^{-4}$	$1.4710 \cdot 10^{-1} \pm 6.3498 \cdot 10^{-2}$
10^{-1}	$3.1413 \cdot 10^{-3} \pm 1.1255 \cdot 10^{-4}$	$1.1030 \cdot 10^{-2} \pm 6.8967 \cdot 10^{-4}$	1	$2.1379 \cdot 10^{-2} \pm 7.8304 \cdot 10^{-4}$	$5.1311 \cdot 10^{-2} \pm 7.9622 \cdot 10^{-4}$	1	$3.1064 \cdot 10^{-2} \pm 2.4043 \cdot 10^{-4}$	$1.0097 \cdot 10^{-1} \pm 2.8932 \cdot 10^{-4}$
1	$3.1576 \cdot 10^{-3} \pm 1.0114 \cdot 10^{-4}$	$1.1216 \cdot 10^{-2} \pm 7.8453 \cdot 10^{-4}$	10	$2.2678 \cdot 10^{-2} \pm 1.0276 \cdot 10^{-4}$	$5.2883 \cdot 10^{-2} \pm 4.6840 \cdot 10^{-4}$			
10	$3.0055 \cdot 10^{-3} \pm 2.4020 \cdot 10^{-5}$	$1.0380 \cdot 10^{-2} \pm 3.9774 \cdot 10^{-4}$	100	$2.2763 \cdot 10^{-2} \pm 8.4000 \cdot 10^{-5}$	$5.3364 \cdot 10^{-2} \pm 2.9700 \cdot 10^{-4}$			

D.5 GRAPH-MODEL HYPERPARAMETERS

The graph-model, optimization, and LCKoop configurations are summarized below.

Table 8: Graph-dynamics hyperparameters used in all Rope, Soft, and Swim experiments. Panel (a) summarizes the shared GNN encoder–decoder and relation-wise transition architecture, panel (b) lists the data, optimization, and evaluation configuration, and panel (c) specifies controlled circular-state regularization with fixed forcing.

(a) Graph-model architecture settings.

Parameter	Value
Encoder / decoder	Separate 2-step GNNs; SiLU
Widths (node/effect, relation)	$(100, 120)$
State path in T_θ	Relation aggregate $\to \mathrm{MLP}_r$: $32 \to 100 \to 100 \to 32$
Forcing path in T_θ	Relation aggregate $\to B_r$: $1 \to 32$
Zero initialization	Final layers of D_ω and MLP_r
Physical update	$\widehat{u}_{t+1} = u_t + \Delta u_t$

(b) Training and evaluation settings.

Parameter	Value
Split (train/val/test)	$40/10/10$ trajectories; 3,960 MS2 training windows
Horizons	Train: MS2; val/test: MS2, MS6 (separate checkpoints)
Optimizer	Adam
Learning-rate schedule	$10^{-4} \xrightarrow{\cos} 10^{-6}$
Epochs / batch / clipping	$100/16/1.0$
Loss	MS2 rollout MSE; $\mathcal{L}_{\mathrm{rec}} = 0$

(c) LCKoop settings. Derivatives are taken with respect to q_t while f_t is held fixed.

Parameter	Value
Flatness coefficient	FC sweep in Appendix D.4
Circular state	$q_t = [u_t, z_t]$
Cut	Full GNN-encoder output
Differentiation	∇_{q_t} with fixed f_t (base, perturbation, JVP)
Penalty	Squared Taylor remainder
Horizon / time sampling	$3/0.15$

E CIRCULAR-WRAPPING TARGET LOCATIONS AND PLUG-IN TRAINING

E.1 CIRCULAR-WRAPPING MAP

The algorithms below show the ordinary one-step forward pass of each backbone with the LCKoop weight set to zero. The highlighted comment marks the separator at which the circular-wrapping state z_t is recorded; it is not an additional network operation and does not change the forecasting path. If $\mathcal{E}_{\text{cut}}$ denotes the operations through the marked line and $\mathcal{D}_{\text{cut}}$ denotes the operations after it, then

$$z_t = \mathcal{E}_{\text{cut}}(u_t; c), \qquad \widehat{u}_{t+1} = F_\theta(u_t; c) = \mathcal{D}_{\text{cut}}(z_t; c), \qquad F_\theta^C(z_t; c) = \mathcal{E}_{\text{cut}}(\widehat{u}_{t+1}; c)\,, \tag{48}$$

where F_θ is the original physical predictor, F_θ^C is its circular-wrapping map in the cut space, and c contains conditioning that is fixed across rollout time, such as the spatial grid, geometry mask, or case parameters. Thus, LCKoop decodes through the remainder of the unmodified backbone and re-executes the prefix through the same marked cut.

E.2 FOURIER NEURAL OPERATOR

Algorithm 1 Default FNO forward pass and circular-wrapping target

Description: Lift state, case-parameter with E, evolve the latent field with Fourier layers $\{\mathcal{K}_\ell, W_\ell\}_{\ell=0}^{L-1}$, and project it with D.

$v^{(0)} \leftarrow E(\text{Concat}(u_t, c))$ ▷ **Circular-wrapping target:** $z_t = v^{(0)}$
for $\ell = 0, \ldots, L-1$ **do**
  $v^{(\ell+1)} \leftarrow \sigma\big(\mathcal{K}_\ell v^{(\ell)} + W_\ell v^{(\ell)}\big)$
end for
$\widehat{u}_{t+1} \leftarrow D\big(v^{(L)}\big)$
return $\widehat{u}_{t+1}$

FNO lifts the physical state and static conditioning to a channel-wise latent field. Each Fourier layer combines a learned pointwise channel map with a spectral convolution: it transforms the field to Fourier space, applies learned multipliers to a fixed set of low-frequency modes, returns to physical space, and applies the activation. A pointwise projection maps the final latent field to the next physical state.

E.3 GFORMER

Algorithm 2 GFormer forward pass and circular-wrapping target

Description: Lift state, case-parameter, and coordinate tokens with E; apply L Galerkin blocks; and decode with D_{spec}.

$r \leftarrow \text{Flatten}(\text{Concat}(u_t, a, p))$
$h^{(0)} \leftarrow E(r)$ ▷ **Circular-wrapping target:** $z_t = h^{(0)}$
for $\ell = 0, \ldots, L-1$ **do**
  $q_\ell, k_\ell, v_\ell \leftarrow Q_\ell h^{(\ell)}, K_\ell h^{(\ell)}, V_\ell h^{(\ell)}$
  $\tilde{h}^{(\ell)} \leftarrow h^{(\ell)} + \text{GalerkinAttn}(q_\ell, k_\ell, v_\ell; p)$ ▷ $Q(K^\top V)$; per each head
  $h^{(\ell+1)} \leftarrow \tilde{h}^{(\ell)} + \text{FFN}_\ell\Big(\tilde{h}^{(\ell)}\Big)$
end for
$H \leftarrow \text{Unflatten}\big(h^{(L)}\big)$
$\widehat{u}_{t+1} \leftarrow D_{\text{spec}}(H)$ ▷ two spectral layers with truncated Fourier modes, followed by a pointwise head
return $\widehat{u}_{t+1}$

GFormer first concatenates the state, time-invariant case parameters, and spatial coordinates, then flattens and linearly lifts the resulting grid tokens. Each block performs global Galerkin attention using the non-softmax product $Q(K^\top V)$ (with attention normalization of keys and values), followed

by a residual feed-forward update. The decoder restores the spatial layout and applies two truncated Fourier layers, including inverse Fourier transforms, before the pointwise output head. The cut is immediately after the token lift and before the first attention block.

E.4 UNet

Algorithm 3 Default UNet forward pass and circular-wrapping target

Description: Encode $\mathrm{Concat}(u_t, c)$ through an L-level UNet, then decode with the corresponding skip connections.

$s^{(0)} \leftarrow \textsc{EncoderBlock}(0, \mathrm{Concat}(u_t, c))$ ▷ **Circular-wrapping target:** $z_t = s^{(0)}$
$q^{(1)} \leftarrow \textsc{Pool}(0, s^{(0)})$
for $\ell = 1, \ldots, L-1$ **do**
  $s^{(\ell)} \leftarrow \textsc{EncoderBlock}(\ell, q^{(\ell)})$ ▷ encoder skip at level ℓ
  $q^{(\ell+1)} \leftarrow \textsc{Pool}(\ell, s^{(\ell)})$
end for
$d^{(L)} \leftarrow \textsc{Bottleneck}(q^{(L)})$
for $\ell = L-1, \ldots, 0$ **do**
  $a^{(\ell)} \leftarrow \textsc{Upsample}(\ell, d^{(\ell+1)})$
  $d^{(\ell)} \leftarrow \textsc{DecoderBlock}(\ell, \mathrm{Concat}(a^{(\ell)}, s^{(\ell)}))$
end for
$\widehat{u}_{t+1} \leftarrow \textsc{OutputHead}(d^{(0)})$
return $\widehat{u}_{t+1}$

UNet first maps the state and static conditioning to full-resolution features, then repeatedly applies encoder blocks and pooling to construct a multiscale feature hierarchy. A bottleneck processes the coarsest representation. The decoder upsamples this representation stage by stage, concatenating the corresponding encoder feature at each resolution, before an output head returns the next physical state. The cut is the full-resolution activation produced by the first encoder block, immediately before the first pooling operation.

E.5 Plug-in training algorithm

Algorithm 4 LCKoop as a plug-in for an autoregressive neural operator

Description: Train the original autoregressive predictor while adding the normalized circular-wrapping curvature penalty only at Bernoulli-selected rollout states.

Sample $m_{0:n-1} \overset{\text{iid}}{\sim} \mathrm{Bernoulli}(p_{\text{time}})$
$\widehat{u}_0 \leftarrow u_0$, $\mathcal{L}_{\text{pred}} \leftarrow 0$, $\mathcal{L}_{\text{flat}} \leftarrow 0$
for $t = 0, \ldots, n-1$ **do**
  $z_t \leftarrow \mathcal{E}_{\text{cut}}(\widehat{u}_t; c)$
  $\widehat{u}_{t+1} \leftarrow F_\theta(\widehat{u}_t; c)$
  $\mathcal{L}_{\text{pred}} \leftarrow \mathcal{L}_{\text{pred}} + \ell_{\text{pred}}(\widehat{u}_{t+1}, u_{t+1})$
  if $m_t = 1$ **then**
    $z'_t \leftarrow \mathrm{sg}(z_t)$, $\delta'_t \leftarrow F^C_\theta(z'_t; c) - z'_t$
    $g_t \leftarrow \mathrm{JVP}_{z'_t}\left[F^C_\theta(\,\cdot\,; c)\right](\delta'_t)$
    $r_t \leftarrow F^C_\theta(z'_t + \delta'_t; c) - F^C_\theta(z'_t; c) - g_t$
    $\mathcal{L}_{\text{flat}} \leftarrow \mathcal{L}_{\text{flat}} + \left[\dfrac{\mathrm{R}(r_t)}{\mathrm{R}(g_t)\mathrm{R}(\delta'_t) + \varepsilon}\right]$
  end if
end for
$\mathcal{L} \leftarrow \mathcal{L}_{\text{pred}} + \lambda_{\text{FC}}\mathcal{L}_{\text{flat}}$
Update all backbone parameters using $\nabla_\theta \mathcal{L}$

Here, F_θ denotes the original physical predictor and F^C_θ denotes the circular-wrapping transition from one cut state to the next; $\mathcal{E}_{\text{cut}}$ and $\mathcal{P}_{\text{cut}}$ are respectively the backbone prefix and suffix at the chosen

cut, while c contains fixed conditioning. Further, sg denotes stop-gradient and the JVP evaluates the Jacobian–vector product without materializing a Jacobian. The prediction path is exactly the original autoregressive backbone; LCKoop only adds the sampled penalty during training. Setting $\lambda_{\mathrm{FC}} = 0$ therefore recovers the original training objective.

E.6 Additional Study: Robotic Graph-Dynamics Forecasting

Let u_t denote the physical node-state tensor and f_t the supplied per-node forcing/control. The model predicts a residual increment, so the physical update $u_t + \Delta u_t$ is part of every autoregressive step. The circular map therefore carries both u_t and the output of the full GNN encoder, z_t, while keeping f_t fixed. Note that static attributes, such as node and relation types, are suppressed in the following circular-map notation.

E.6.1 Graph-Dynamics

Algorithm 5 Robot forward pass and circular-wrapping target

Description: Encode the node embedding, apply the relation-type-wise transition, decode a residual increment using GNN and MLP blocks. The architecture strongly resembles the original operator, but with an MLP-based transition replacing the original linear transition.
Inputs: $u_t \in \mathbb{R}^{N \times d_u}$ and $f_t \in \mathbb{R}^{N \times 1}$.

$z_t \leftarrow E_\psi(u_t)$ ▷ full two-step GNN encoder
$q_t \leftarrow [u_t, z_t]$ ▷ **Circular-wrapping target:** q_t
$z'_t \leftarrow T_\theta(z_t, f_t)$ ▷ latent transition

$$[T_\theta(z_t, f_t, R)]_i = z_{t,i} + \sum_r \mathrm{MLP}_r\Big(\sum_j R_{ijr} z_{t,j}\Big) + \sum_r B_r\Big(\sum_j R_{ijr} f_{t,j}\Big).$$

$\Delta u_t \leftarrow D_\omega(z'_t)$; $\widehat{u}_{t+1} \leftarrow u_t + \Delta u_t$ ▷ physical residual
$\widehat{z}_{t+1} \leftarrow E_\psi(\widehat{u}_{t+1})$ ▷ full re-encoding
return $\widehat{u}_{t+1}, \widehat{z}_{t+1}$

We retain the original encoder and decoder and replace the linear transition with an MLP-based transition, preserving the same architectural inductive bias. The encoder and decoder are separate two-step GNNs: each forms node and edge representations from node attributes and state, relation type, states, and sender/receiver attributes, then aggregates relation effects at each receiving node. The transition preserves the structure of the original Koopman operator, where each relation first aggregates neighboring states and forcing, and then applies a shared relation-specific linear operator block:

$$z'_{t,i} = z_{t,i} + \sum_{r=1}^{R}(\sum_j R_{ijr} z_{t,j}) A_r + \sum_{r=1}^{R}(\sum_j R_{ijr} f_{t,j}) B_r. \tag{49}$$

Our model makes a minimal change by replacing A_r with an MLP-based nonlinear transition.

E.6.2 PLUG-IN TRAINING ALGORITHM ON ROBOTIC GRAPH-DYNAMICS

Algorithm 6 LCKoop plug-in training for robotic graph dynamics

Description: Train the residual predictor autoregressively while adding the circular-wrapping curvature penalty only at Bernoulli-selected rollout states.

Sample $b_{0:n} \overset{\text{iid}}{\sim} \text{Bernoulli}(0.15)$; set $\widehat{u}_0 \leftarrow u_0$, $\mathcal{L}_{\text{pred}} \leftarrow 0$, $\mathcal{L}_{\text{flat}} \leftarrow 0$
for $t = 0, \dots, n$ **do**
  $z_t \leftarrow E_\psi(\widehat{u}_t)$, $q_t \leftarrow [\widehat{u}_t, z_t]$
  $[\widehat{u}_{t+1}, \widehat{z}_{t+1}] \leftarrow F^C_{\theta,\text{graph}}(q_t; f_t)$
  if $t < n$ **then**
    $\mathcal{L}_{\text{pred}} \leftarrow \mathcal{L}_{\text{pred}} + \text{MSE}(\widehat{u}_{t+1}, u_{t+1})$
  end if
  if $b_t = 1$ **then**
    $q'_t \leftarrow \text{sg}(q_t)$, $\delta'_t \leftarrow F^C_{\theta,\text{graph}}(q'_t; f_t) - q'_t$
    $g_t \leftarrow \text{JVP}_{q'_t}[F^C_{\theta,\text{graph}}(\cdot; f_t)](\delta_t)$ ▷ the same fixed f_t
    $r_t \leftarrow F^C_{\theta,\text{graph}}(q'_t + \delta'_t; f_t) - F^C_{\theta,\text{graph}}(q'_t; f_t) - g_t$
    $\mathcal{L}_{\text{flat}} \leftarrow \mathcal{L}_{\text{flat}} + \left[\dfrac{\text{R}(r_t)}{\text{R}(g_t)\text{R}(\delta'_t) + \varepsilon}\right]$
  end if
end for
$\mathcal{L} \leftarrow \mathcal{L}_{\text{pred}} + \lambda_{\text{FC}}\mathcal{L}_{\text{flat}}$; update all E–F–D parameters

Similarly to Algorithm 4, $F^C_{\theta,\text{graph}}$ denotes the circular-wrapping transition of the original autoregressive prediction step, but with the additional forcing term f_t and augmented state $q_t = [u_t, z_t]$. Importantly, f_t is fixed after the base state is selected so that the penalty measures curvature only with respect to the propagated state q_t.

F ADDITIONAL CFDNNEVAL EXPERIMENTS

F.1 COMPUTATIONAL OVERHEAD

We quantify the additional training cost of LCKoop per epoch. At the fixed 15% Bernoulli sampling rate, per-epoch overhead ranges from 14.0% for FNO to 31.7% for GFormer, with 26.9% for UNet (Table 9). At each selected timepoint, the penalty adds two forward evaluations from a detached state and a JVP computed with one backward pass; Bernoulli sampling limits this computation to a subset of the rollout.

Table 9: Per-epoch training time with and without LCKoop ($\lambda_{\text{FC}} = 10^{-2}, 0$). Overhead is measured relative to the unregularized configuration.

Model	$\lambda_{\text{FC}} = 0$	$\lambda_{\text{FC}} = 10^{-2}$	Overhead
GFormer	27.05 s	35.61 s	**+31.7%**
FNO	17.25 s	19.66 s	**+14.0%**
UNet	15.33 s	19.45 s	**+26.9%**

F.2 UNET CAPACITY ABLATION

We test whether LCKoop requires sufficient backbone capacity. On Two-Phase Flow/NSCH we test two kernel size configuration of UNet, with size of 3 and 6. In the case where LCKoop is not applied, k6f12 UNet outperforms k3f12, indicating k3f12 as insufficient low-capacity model. LCKoop degrades most metrics with the small-kernel UNet but improves all metrics after enlarging the kernel (Table 10). This suggests that a low-capacity model may be unable to accommodate both forecasting and local-affine consistency, which is why we use the `k6f12` UNet consistently in the main experiments.

Table 10: UNet capacity results. Top: kernel-size ablation on Two-Phase Flow/NSCH. Bottom: `k3f12` results on the remaining domains as a smaller-capacity comparison with the main `k6f12` results in Table 1. The same 1% coloring rule is used; `k` and `f` denote kernel size and feature channels.

	ms3				ms9			
Configuration	MSE	RMSE	MAE	L2RE	MSE	RMSE	MAE	L2RE
`k3f12`	−17.6%	−11.4%	−16.1%	+5.7%	−8.6%	−6.5%	−10.4%	−4.0%
`k6f12`	+53.7%	+23.7%	+25.7%	+21.2%	+42.4%	+22.0%	+25.0%	+15.1%

(`k3f12`)	ms3				ms9			
Domain	MSE	RMSE	MAE	L2RE	MSE	RMSE	MAE	L2RE
Cavity	+10.0%	+5.0%	+6.9%	+5.0%	+5.1%	+3.5%	+8.4%	+6.0%
Tube	+5.5%	+4.5%	+6.5%	+2.1%	+9.9%	+7.0%	+11.5%	+4.5%
Shallow Waters	−0.7%	−0.3%	−0.4%	+0.1%	+0.6%	+0.6%	+0.9%	+0.5%

F.3 Sampling-rate ablation

In the Shallow Waters FNO ablation, low Bernoulli sampling rates capture most of the ms9 L2RE improvement, after which accuracy plateaus as training time increases (Figure 5).

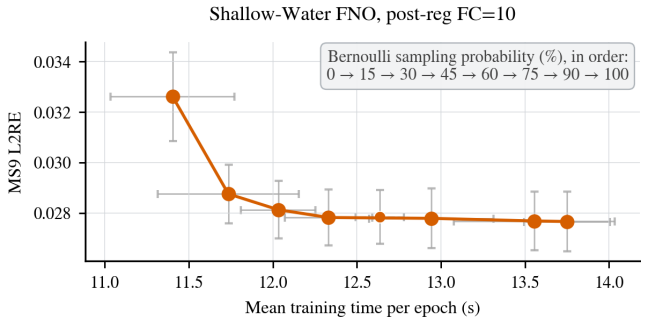


Figure 5: Sampling–performance trade-off for Shallow Waters FNO. Bernoulli sampling probability increases computation time, while ms9 L2RE improves primarily at low sampling rates.

G FULL RESULTS

This appendix reports the full test values used in Table 1. Every entry is a field-mean metric, reported as mean ± standard deviation; these are not global space–time metrics. Metrics use scientific notation with four significant figures. The UNet entries use the consistent main `k6f12` configuration.

G.1 MAIN EXPERIMENT RESULTS

Table 11: Cavity: denormalized test field-mean metrics.

Backbone	Test	λ_{FC}	MSE	RMSE	MAE	L2RE
GFormer	ms3	0	$4.7115 \cdot 10^{-5}$ $\pm 2.6393 \cdot 10^{-5}$	$4.8040 \cdot 10^{-3}$ $\pm 1.3640 \cdot 10^{-3}$	$2.6430 \cdot 10^{-3}$ $\pm 6.7000 \cdot 10^{-4}$	$2.1775 \cdot 10^{-1}$ $\pm 4.4494 \cdot 10^{-2}$
		10^{-2}	$5.0139 \cdot 10^{-5}$ $\pm 3.0623 \cdot 10^{-5}$	$5.2990 \cdot 10^{-3}$ $\pm 2.1020 \cdot 10^{-3}$	$2.7990 \cdot 10^{-3}$ $\pm 1.4500 \cdot 10^{-3}$	$2.2712 \cdot 10^{-1}$ $\pm 5.0373 \cdot 10^{-2}$
UNet	ms3	0	$1.8187 \cdot 10^{-4}$ $\pm 1.7300 \cdot 10^{-5}$	$1.1421 \cdot 10^{-2}$ $\pm 4.6200 \cdot 10^{-4}$	$5.7950 \cdot 10^{-3}$ $\pm 2.9900 \cdot 10^{-4}$	$3.5286 \cdot 10^{-1}$ $\pm 1.6450 \cdot 10^{-2}$
		10^{-1}	$1.8159 \cdot 10^{-4}$ $\pm 3.8500 \cdot 10^{-5}$	$1.1235 \cdot 10^{-2}$ $\pm 1.2180 \cdot 10^{-3}$	$5.7230 \cdot 10^{-3}$ $\pm 7.7000 \cdot 10^{-4}$	$3.4864 \cdot 10^{-1}$ $\pm 3.5450 \cdot 10^{-2}$
FNO	ms3	0	$5.1047 \cdot 10^{-5}$ $\pm 1.0426 \cdot 10^{-5}$	$5.6430 \cdot 10^{-3}$ $\pm 5.6300 \cdot 10^{-4}$	$3.2220 \cdot 10^{-3}$ $\pm 3.4000 \cdot 10^{-4}$	$2.0142 \cdot 10^{-1}$ $\pm 2.0464 \cdot 10^{-2}$
		10^{-2}	$5.7835 \cdot 10^{-5}$ $\pm 1.6430 \cdot 10^{-5}$	$6.0019 \cdot 10^{-3}$ $\pm 8.2680 \cdot 10^{-4}$	$3.4289 \cdot 10^{-3}$ $\pm 4.6759 \cdot 10^{-4}$	$2.1073 \cdot 10^{-1}$ $\pm 2.8134 \cdot 10^{-2}$
GFormer	ms9	0	$1.6804 \cdot 10^{-4}$ $\pm 1.9873 \cdot 10^{-4}$	$9.0050 \cdot 10^{-3}$ $\pm 6.7030 \cdot 10^{-3}$	$4.8980 \cdot 10^{-3}$ $\pm 4.1420 \cdot 10^{-3}$	$3.2100 \cdot 10^{-1}$ $\pm 1.6966 \cdot 10^{-1}$
		10^{-2}	$1.0670 \cdot 10^{-4}$ $\pm 8.8489 \cdot 10^{-5}$	$7.6560 \cdot 10^{-3}$ $\pm 4.2110 \cdot 10^{-3}$	$3.6680 \cdot 10^{-3}$ $\pm 2.7920 \cdot 10^{-3}$	$2.8854 \cdot 10^{-1}$ $\pm 9.4788 \cdot 10^{-2}$
UNet	ms9	0	$2.4023 \cdot 10^{-4}$ $\pm 3.5300 \cdot 10^{-5}$	$1.3317 \cdot 10^{-2}$ $\pm 8.2200 \cdot 10^{-4}$	$6.6150 \cdot 10^{-3}$ $\pm 5.0000 \cdot 10^{-4}$	$3.8065 \cdot 10^{-1}$ $\pm 1.9630 \cdot 10^{-2}$
		10^{-1}	$2.1459 \cdot 10^{-4}$ $\pm 4.1300 \cdot 10^{-5}$	$1.2492 \cdot 10^{-2}$ $\pm 1.2750 \cdot 10^{-3}$	$6.3020 \cdot 10^{-3}$ $\pm 8.3200 \cdot 10^{-4}$	$3.5502 \cdot 10^{-1}$ $\pm 3.1490 \cdot 10^{-2}$
FNO	ms9	0	$3.1441 \cdot 10^{-4}$ $\pm 4.2519 \cdot 10^{-4}$	$1.3367 \cdot 10^{-2}$ $\pm 7.9870 \cdot 10^{-3}$	$7.5070 \cdot 10^{-3}$ $\pm 5.0960 \cdot 10^{-3}$	$3.7939 \cdot 10^{-1}$ $\pm 1.7301 \cdot 10^{-1}$
		10^{-2}	$1.5321 \cdot 10^{-4}$ $\pm 5.6235 \cdot 10^{-5}$	$1.0541 \cdot 10^{-2}$ $\pm 2.0111 \cdot 10^{-3}$	$5.7850 \cdot 10^{-3}$ $\pm 1.1372 \cdot 10^{-3}$	$3.2030 \cdot 10^{-1}$ $\pm 6.2276 \cdot 10^{-2}$

Table 12: Tube: denormalized test field-mean metrics.

Backbone	Test	λ_{FC}	MSE	RMSE	MAE	L2RE
GFormer	ms3	0	$2.8440 \cdot 10^{-3}$ $\pm 6.1900 \cdot 10^{-4}$	$3.7629 \cdot 10^{-2}$ $\pm 4.1550 \cdot 10^{-3}$	$2.3829 \cdot 10^{-2}$ $\pm 3.1840 \cdot 10^{-3}$	$1.0619 \cdot 10^{-1}$ $\pm 8.3210 \cdot 10^{-3}$
		10^{-2}	$2.6390 \cdot 10^{-3}$ $\pm 4.5500 \cdot 10^{-4}$	$3.6401 \cdot 10^{-2}$ $\pm 3.2410 \cdot 10^{-3}$	$2.2600 \cdot 10^{-2}$ $\pm 2.2680 \cdot 10^{-3}$	$1.0402 \cdot 10^{-1}$ $\pm 6.0220 \cdot 10^{-3}$
UNet	ms3	0	$4.6500 \cdot 10^{-3}$ $\pm 1.6700 \cdot 10^{-4}$	$5.0351 \cdot 10^{-2}$ $\pm 6.9600 \cdot 10^{-4}$	$3.2646 \cdot 10^{-2}$ $\pm 1.3380 \cdot 10^{-3}$	$1.4325 \cdot 10^{-1}$ $\pm 5.5100 \cdot 10^{-3}$
		10^{-2}	$4.2640 \cdot 10^{-3}$ $\pm 8.7800 \cdot 10^{-4}$	$4.8291 \cdot 10^{-2}$ $\pm 4.7620 \cdot 10^{-3}$	$3.0663 \cdot 10^{-2}$ $\pm 3.6820 \cdot 10^{-3}$	$1.4162 \cdot 10^{-1}$ $\pm 1.2940 \cdot 10^{-2}$
FNO	ms3	0	$8.2763 \cdot 10^{-3}$ $\pm 6.1000 \cdot 10^{-4}$	$5.9214 \cdot 10^{-2}$ $\pm 1.5530 \cdot 10^{-3}$	$3.8805 \cdot 10^{-2}$ $\pm 1.3120 \cdot 10^{-3}$	$1.5931 \cdot 10^{-1}$ $\pm 2.2760 \cdot 10^{-3}$
		10^{-2}	$8.7070 \cdot 10^{-3}$ $\pm 5.2236 \cdot 10^{-4}$	$5.9764 \cdot 10^{-2}$ $\pm 1.2846 \cdot 10^{-3}$	$3.9926 \cdot 10^{-2}$ $\pm 1.3695 \cdot 10^{-3}$	$1.5920 \cdot 10^{-1}$ $\pm 2.4672 \cdot 10^{-3}$
GFormer	ms9	0	$8.1470 \cdot 10^{-3}$ $\pm 3.9430 \cdot 10^{-3}$	$6.5580 \cdot 10^{-2}$ $\pm 1.4229 \cdot 10^{-2}$	$4.2436 \cdot 10^{-2}$ $\pm 1.0828 \cdot 10^{-2}$	$1.5579 \cdot 10^{-1}$ $\pm 3.0123 \cdot 10^{-2}$
		10^{-2}	$6.4980 \cdot 10^{-3}$ $\pm 2.5260 \cdot 10^{-3}$	$5.7704 \cdot 10^{-2}$ $\pm 1.0565 \cdot 10^{-2}$	$3.6539 \cdot 10^{-2}$ $\pm 9.2830 \cdot 10^{-3}$	$1.4147 \cdot 10^{-1}$ $\pm 1.8898 \cdot 10^{-2}$
UNet	ms9	0	$8.1060 \cdot 10^{-3}$ $\pm 1.0520 \cdot 10^{-3}$	$6.5564 \cdot 10^{-2}$ $\pm 2.9050 \cdot 10^{-3}$	$4.1481 \cdot 10^{-2}$ $\pm 2.2710 \cdot 10^{-3}$	$1.5943 \cdot 10^{-1}$ $\pm 5.0100 \cdot 10^{-3}$
		10^{-2}	$6.5900 \cdot 10^{-3}$ $\pm 1.9450 \cdot 10^{-3}$	$6.0957 \cdot 10^{-2}$ $\pm 7.6910 \cdot 10^{-3}$	$3.8433 \cdot 10^{-2}$ $\pm 5.9910 \cdot 10^{-3}$	$1.5501 \cdot 10^{-1}$ $\pm 1.6750 \cdot 10^{-2}$
FNO	ms9	0	$1.8783 \cdot 10^{-2}$ $\pm 4.1120 \cdot 10^{-3}$	$9.4864 \cdot 10^{-2}$ $\pm 9.9540 \cdot 10^{-3}$	$6.9189 \cdot 10^{-2}$ $\pm 8.5910 \cdot 10^{-3}$	$2.0195 \cdot 10^{-1}$ $\pm 1.4013 \cdot 10^{-2}$
		10^{-2}	$1.7974 \cdot 10^{-2}$ $\pm 2.3810 \cdot 10^{-3}$	$9.2058 \cdot 10^{-2}$ $\pm 7.2262 \cdot 10^{-3}$	$6.7462 \cdot 10^{-2}$ $\pm 7.1846 \cdot 10^{-3}$	$1.9870 \cdot 10^{-1}$ $\pm 8.8666 \cdot 10^{-3}$

Table 13: Two-phase flow / NSCH: denormalized test field-mean metrics.

Backbone	Test	λ_{FC}	MSE	RMSE	MAE	L2RE
GFormer	ms3	0	$1.0390 \cdot 10^{-3} \pm 4.1900 \cdot 10^{-4}$	$2.1947 \cdot 10^{-2} \pm 4.8340 \cdot 10^{-3}$	$1.3643 \cdot 10^{-2} \pm 2.4810 \cdot 10^{-3}$	$1.1676 \cdot 10^{-1} \pm 5.1861 \cdot 10^{-2}$
		10^{-1}	$8.9100 \cdot 10^{-4} \pm 3.8500 \cdot 10^{-4}$	$2.0345 \cdot 10^{-2} \pm 4.6370 \cdot 10^{-3}$	$1.2838 \cdot 10^{-2} \pm 2.5890 \cdot 10^{-3}$	$1.0395 \cdot 10^{-1} \pm 5.1673 \cdot 10^{-2}$
UNet	ms3	0	$1.9490 \cdot 10^{-3} \pm 2.1700 \cdot 10^{-3}$	$2.9692 \cdot 10^{-2} \pm 1.3493 \cdot 10^{-2}$	$1.9781 \cdot 10^{-2} \pm 9.4800 \cdot 10^{-3}$	$2.3924 \cdot 10^{-1} \pm 1.2391 \cdot 10^{-1}$
		10^{-2}	$9.0200 \cdot 10^{-4} \pm 7.8000 \cdot 10^{-5}$	$2.2658 \cdot 10^{-2} \pm 8.2500 \cdot 10^{-4}$	$1.4693 \cdot 10^{-2} \pm 4.7900 \cdot 10^{-4}$	$1.8864 \cdot 10^{-1} \pm 9.8520 \cdot 10^{-3}$
FNO	ms3	0	$9.8700 \cdot 10^{-4} \pm 5.7730 \cdot 10^{-5}$	$2.2494 \cdot 10^{-2} \pm 1.2070 \cdot 10^{-3}$	$1.4990 \cdot 10^{-2} \pm 8.3000 \cdot 10^{-4}$	$1.5940 \cdot 10^{-1} \pm 5.9608 \cdot 10^{-2}$
		10^{-2}	$9.9300 \cdot 10^{-4} \pm 3.1940 \cdot 10^{-5}$	$2.2460 \cdot 10^{-2} \pm 8.0400 \cdot 10^{-4}$	$1.5019 \cdot 10^{-2} \pm 4.9800 \cdot 10^{-4}$	$1.5429 \cdot 10^{-1} \pm 5.4555 \cdot 10^{-2}$
GFormer	ms9	0	$3.6430 \cdot 10^{-3} \pm 1.4840 \cdot 10^{-3}$	$3.5422 \cdot 10^{-2} \pm 8.1980 \cdot 10^{-3}$	$1.8091 \cdot 10^{-2} \pm 3.6060 \cdot 10^{-3}$	$1.2244 \cdot 10^{-1} \pm 4.1723 \cdot 10^{-2}$
		10^{-1}	$3.1320 \cdot 10^{-3} \pm 1.3330 \cdot 10^{-3}$	$3.2841 \cdot 10^{-2} \pm 7.5850 \cdot 10^{-3}$	$1.7019 \cdot 10^{-2} \pm 3.5910 \cdot 10^{-3}$	$1.1208 \cdot 10^{-1} \pm 4.6563 \cdot 10^{-2}$
UNet	ms9	0	$5.3950 \cdot 10^{-3} \pm 4.4140 \cdot 10^{-3}$	$4.4871 \cdot 10^{-2} \pm 1.7336 \cdot 10^{-2}$	$2.4742 \cdot 10^{-2} \pm 1.0512 \cdot 10^{-2}$	$1.9978 \cdot 10^{-1} \pm 7.5555 \cdot 10^{-2}$
		10^{-2}	$3.1090 \cdot 10^{-3} \pm 3.2400 \cdot 10^{-4}$	$3.5015 \cdot 10^{-2} \pm 1.7320 \cdot 10^{-3}$	$1.8555 \cdot 10^{-2} \pm 6.8500 \cdot 10^{-4}$	$1.6970 \cdot 10^{-1} \pm 7.4620 \cdot 10^{-3}$
FNO	ms9	0	$3.0280 \cdot 10^{-3} \pm 5.5500 \cdot 10^{-5}$	$3.3944 \cdot 10^{-2} \pm 9.6400 \cdot 10^{-4}$	$1.8811 \cdot 10^{-2} \pm 7.0400 \cdot 10^{-4}$	$1.4009 \cdot 10^{-1} \pm 3.8012 \cdot 10^{-2}$
		10^{-2}	$2.9880 \cdot 10^{-3} \pm 5.7530 \cdot 10^{-5}$	$3.3778 \cdot 10^{-2} \pm 7.8000 \cdot 10^{-4}$	$1.8870 \cdot 10^{-2} \pm 5.4300 \cdot 10^{-4}$	$1.3567 \cdot 10^{-1} \pm 3.4152 \cdot 10^{-2}$

Table 14: Shallow Waters: denormalized test field-mean metrics.

Backbone	Test	λ_{FC}	MSE	RMSE	MAE	L2RE
GFormer	ms3	0	$7.5615 \cdot 10^{-7} \pm 1.3992 \cdot 10^{-7}$	$5.2066 \cdot 10^{-4} \pm 4.0745 \cdot 10^{-5}$	$3.7805 \cdot 10^{-4} \pm 3.0874 \cdot 10^{-5}$	$2.4450 \cdot 10^{-2} \pm 2.2800 \cdot 10^{-3}$
		10^{-1}	$6.7798 \cdot 10^{-7} \pm 1.0602 \cdot 10^{-7}$	$4.9955 \cdot 10^{-4} \pm 2.5585 \cdot 10^{-5}$	$3.6394 \cdot 10^{-4} \pm 1.8677 \cdot 10^{-5}$	$2.3140 \cdot 10^{-2} \pm 9.6000 \cdot 10^{-4}$
UNet	ms3	0	$3.3441 \cdot 10^{-7} \pm 5.0600 \cdot 10^{-8}$	$4.3265 \cdot 10^{-4} \pm 3.6600 \cdot 10^{-5}$	$3.3476 \cdot 10^{-4} \pm 2.8300 \cdot 10^{-5}$	$2.0627 \cdot 10^{-2} \pm 2.2070 \cdot 10^{-3}$
		10^{-2}	$3.1856 \cdot 10^{-7} \pm 4.1600 \cdot 10^{-8}$	$4.2054 \cdot 10^{-4} \pm 3.1900 \cdot 10^{-5}$	$3.2572 \cdot 10^{-4} \pm 2.4900 \cdot 10^{-5}$	$1.9888 \cdot 10^{-2} \pm 1.9000 \cdot 10^{-3}$
FNO	ms3	0	$6.8319 \cdot 10^{-7} \pm 8.2673 \cdot 10^{-8}$	$6.1791 \cdot 10^{-4} \pm 3.6977 \cdot 10^{-5}$	$4.5477 \cdot 10^{-4} \pm 2.5625 \cdot 10^{-5}$	$3.1490 \cdot 10^{-2} \pm 2.0100 \cdot 10^{-3}$
		10	$5.9042 \cdot 10^{-7} \pm 6.8960 \cdot 10^{-8}$	$5.6715 \cdot 10^{-4} \pm 3.2501 \cdot 10^{-5}$	$4.0956 \cdot 10^{-4} \pm 2.2738 \cdot 10^{-5}$	$2.8110 \cdot 10^{-2} \pm 1.8100 \cdot 10^{-3}$
GFormer	ms9	0	$5.8607 \cdot 10^{-7} \pm 1.0052 \cdot 10^{-7}$	$5.4432 \cdot 10^{-4} \pm 4.2270 \cdot 10^{-5}$	$3.8779 \cdot 10^{-4} \pm 3.2609 \cdot 10^{-5}$	$2.4990 \cdot 10^{-2} \pm 2.2400 \cdot 10^{-3}$
		10^{-1}	$5.3496 \cdot 10^{-7} \pm 7.2719 \cdot 10^{-8}$	$5.2901 \cdot 10^{-4} \pm 2.4690 \cdot 10^{-5}$	$3.7869 \cdot 10^{-4} \pm 1.8196 \cdot 10^{-5}$	$2.3980 \cdot 10^{-2} \pm 9.6000 \cdot 10^{-4}$
UNet	ms9	0	$2.8943 \cdot 10^{-7} \pm 4.2800 \cdot 10^{-8}$	$4.5697 \cdot 10^{-4} \pm 3.5300 \cdot 10^{-5}$	$3.4525 \cdot 10^{-4} \pm 2.6900 \cdot 10^{-5}$	$2.1486 \cdot 10^{-2} \pm 1.6740 \cdot 10^{-3}$
		10^{-2}	$2.7426 \cdot 10^{-7} \pm 3.5900 \cdot 10^{-8}$	$4.4295 \cdot 10^{-4} \pm 3.0300 \cdot 10^{-5}$	$3.3451 \cdot 10^{-4} \pm 2.3500 \cdot 10^{-5}$	$2.0831 \cdot 10^{-2} \pm 1.3990 \cdot 10^{-3}$
FNO	ms9	0	$6.5770 \cdot 10^{-7} \pm 7.5140 \cdot 10^{-8}$	$6.7405 \cdot 10^{-4} \pm 3.9652 \cdot 10^{-5}$	$4.9943 \cdot 10^{-4} \pm 2.9435 \cdot 10^{-5}$	$3.2610 \cdot 10^{-2} \pm 1.7600 \cdot 10^{-3}$
		10	$5.3855 \cdot 10^{-7} \pm 5.6347 \cdot 10^{-8}$	$6.0596 \cdot 10^{-4} \pm 3.0917 \cdot 10^{-5}$	$4.3799 \cdot 10^{-4} \pm 2.2149 \cdot 10^{-5}$	$2.8750 \cdot 10^{-2} \pm 1.1600 \cdot 10^{-3}$