Stable Long-Horizon PDE Forecasting via Latent Structured Spectral Propagators

Xiaoxiao Lu **Jiahao Shi** **Ye Yuan***
School of Artificial Intelligence and Automation,
Huazhong University of Science and Technology, China
{xiaolu, jiahaoshi, yye}@hust.edu.cn

Abstract

Long-horizon forecasting of time-dependent partial differential equations (PDEs) is critical for characterizing the sustained evolution of physical systems. While neural operators have emerged as efficient surrogates, they typically learn implicit finite-time transitions from discrete observations. When deployed autoregressively, such propagators often suffer from rapid error accumulation and dynamic drift. To address this, we propose a neural forecasting framework that reformulates PDE rollout as learning a Structured Spectral Propagator (SSP) in a propagation-oriented latent space. Following an analysis-propagation-synthesis design, our framework: (i) maps physical states into a shared, time-consistent spatial representation; (ii) projects this space into a compact propagation state to isolate recurrent dynamics from fine-grained spatial details, thereby decoupling reconstruction fidelity from rollout regularity; and (iii) evolves retained spectral modes using a frequency-conditioned linear backbone complemented by a nonlinear spectral closure to account for truncated interactions. This explicit structuring endows the propagator with a strong inductive bias for coherent modal evolution. Extensive experiments demonstrate that SSP significantly outperforms state-of-the-art baselines, reducing relative L_2 errors by up to 48.9% and exhibiting improved stability in temporal extrapolation beyond the supervised horizon.

1 Introduction

Modeling physical dynamics governed by time-dependent partial differential equations (PDEs) is a central problem in computational science. Data-driven methods have been used to identify hybrid dynamical systems [1] and to construct neural surrogates for PDE evolution, particularly through neural operators [2–8]. In practice, these models are commonly trained from discrete observations to learn finite-time propagation maps and then deployed autoregressively [9–12]. However, many scientific applications require long-horizon forecasts to characterize the long-time behavior of physical systems. Such forecasts are obtained by iteratively composing the learned propagator, with each predicted state fed back as input. In this recursive regime, even tiny one-step approximation errors can be nonlinearly amplified, causing the model to drift away from the manifold of physical solutions [10–13]. Therefore, the goal is not merely to fit finite-time transitions but to learn a propagator whose internal update remains structured and reliable under repeated composition.

Existing neural PDE surrogates approach long-horizon prediction from several directions. Rollout-oriented methods [9, 10, 12, 14, 15] mitigate error accumulation through recurrent supervision, temporal stencils, or refinement, while the recurrent transition itself often remains an implicit map with limited structure. Latent-space methods [16–21] reduce evolution complexity by learning compact representations, and Koopman-inspired models [22–24] introduce linear dynamics priors in learned

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

observable spaces. However, weak geometric constraints on the latent space create an intrinsic conflict between representation richness, dynamical simplification, and stability. Accurate reconstruction often demands high-dimensional latent spaces to preserve fine-grained spatial details, yet complexity frequently compromises the regularity and identifiable structure of temporal advancement.

A natural question is how to endow structural constraints on the propagator for stable recursive application. We take the spectral properties of PDE evolution as an inductive bias for propagation law learning [25]. For translation-invariant linear PDEs, temporal advancement decomposes into independent low-dimensional linear systems in the spatial Fourier basis, suggesting that propagation may admit a more explicit internal organization in a suitable representation. For general nonlinear PDEs, such regularity is no longer available in the original domain, motivating us to learn a time-consistent spatial representation in which the dynamics admit a more tractable description.

We propose Structured Spectral Propagator (SSP) for long-horizon PDE forecasting in propagation-oriented latent spaces. SSP encodes each time slice into a shared, time-consistent spatial-basis representation and projects it into a compact propagation state. This compact state forms the propagation-oriented space, where redundant degrees of freedom from the rich spatial representation are suppressed. The propagator advances this state by evolving a retained set of latent Fourier modes with a frequency-conditioned linear backbone, complemented by a nonlinear spectral closure that accounts for residual cross-mode interactions and truncation-induced effects. The inverse projection lifts the propagated state back to the spatial-basis space, and the decoder synthesizes the physical field. This design separates spatial representation construction from temporal advancement, yielding a compact and structured update law for recursive rollout.

Our contributions are summarized as follows: (i) we formulate long-horizon PDE forecasting as structured propagator learning under repeated autoregressive composition; (ii) we separate spatial-basis representation from temporal propagation through a frame-wise encoder and a compact propagation state, reducing the conflict between reconstruction fidelity and rollout regularity; (iii) we instantiate structured spectral propagation with a frequency-conditioned linear modal backbone and nonlinear spectral closure, and validate the design on long-horizon and beyond-horizon PDE forecasting benchmarks across distinct dynamical regimes.

2 Background

Partial differential equations. Let $\Omega \subset \mathbb{R}^d$ be a spatial domain with boundary $\partial\Omega$, and let $u : \Omega \times [0, T_{\max}] \to \mathbb{R}^{d_u}$ denote the solution field. We consider the evolution of physical fields governed by time-dependent PDEs:

$$F\left(x, t, \{\partial_t^r D_x^\alpha u(x,t)\}_{0\le r\le s_t,\ |\alpha|\le s_x}\right) = 0, \qquad (x,t) \in \Omega \times (0, T_{\max}], \tag{1}$$

where D_x^α denotes a spatial derivative with multi-index α, and s_t and s_x denote the maximal temporal and spatial derivative orders. The operator F may contain both linear and nonlinear differential terms. The equation is equipped with initial conditions $\partial_t^j u(x,0) = u_j(x)$ for $j = 0, \dots, s_t - 1$, and boundary conditions $\mathcal{B}[u](x,t) = g(x,t)$ on $\partial\Omega \times (0, T_{\max}]$.

Neural operators as finite-time propagators. Let $\mathcal{U}$ and $\mathcal{V}$ be Banach spaces of input and solution functions. Operator learning [6, 26–31] aims to approximate a ground-truth operator $\mathcal{G} : \mathcal{U} \to \mathcal{V}$ by a neural operator $\widehat{\mathcal{G}} : \mathcal{U} \to \mathcal{V}$. For time-dependent PDEs, this operator is typically learned through finite-time prediction between solution snapshots or short temporal windows [22]. Long-horizon forecasting then requires recursively composing the learned propagator to approximate the global evolution induced by $\mathcal{G}$. Therefore, the learning problem is not fully characterized by fitting individual finite-time transitions: the learned map must also remain meaningful and reliable when repeatedly applied during rollout. This motivates learning a time-consistent representation in which the evolving state admits a shared, structurally constrained propagation law across rollout steps.

Latent-space operator factorization. Latent-space methods [16–21, 32–35] approximate the operator through a factorized map, $\mathcal{G} \approx \widehat{\mathcal{G}} := \mathcal{D} \circ \Phi \circ \mathcal{E}$, where the encoder $\mathcal{E} : \mathcal{U} \to \mathcal{Z}$ maps input functions to a latent space, the processor $\Phi : \mathcal{Z} \to \mathcal{Z}$ evolves latent states, and the decoder $\mathcal{D} : \mathcal{Z} \to \mathcal{V}$ maps them back to the solution space. Such encoder–processor–decoder designs are commonly used to reduce computational cost, enable scalable latent processing, and improve

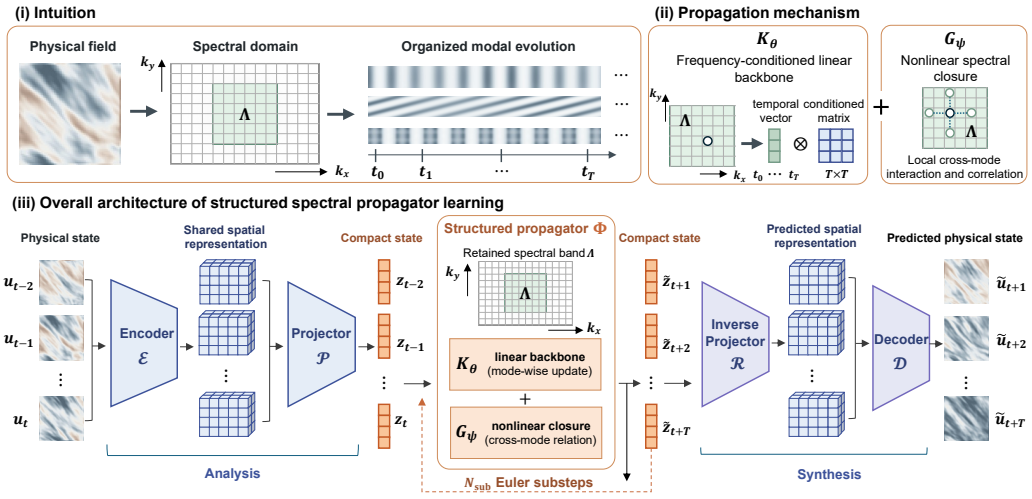


Figure 1: Overview of the proposed SSP framework. (i) Spectral intuition: a suitable spatial basis exposes organized modal evolution in the spectral domain. (ii) Propagation mechanism: the structured propagator combines a frequency-conditioned linear backbone K_θ for mode-wise temporal updates with a nonlinear closure G_ψ for local cross-mode interactions within the retained spectral band. (iii) Overall architecture: SSP follows an analysis–propagation–synthesis pipeline, encoding physical states into a shared spatial representation, projecting them to compact propagation states, performing structured spectral propagation, and decoding the lifted representations into future physical fields.

flexibility across discretizations or geometries. Existing methods such as LNO [18], CALM-PDE [20], AROMA [19], and hybrids based on proper orthogonal decomposition (POD) or reduced-order modeling (ROM) [32, 36, 37] exploit compact latent coordinates for efficiency and adaptation. Despite their effectiveness, compact latent spaces are often designed primarily as computational representations, rather than explicitly structured to support time-consistent recursive propagation.

Koopman-inspired propagation constraints. Koopman-inspired methods [22–24] further constrain latent evolution by parameterizing Φ to advance learned observables in an approximately linear manner. This follows the Koopman view that a nonlinear flow induces a linear operator on a suitable space of observables. Related Koopman-theoretic studies have also established sampling conditions for exact identification of continuous-time nonlinear dynamical systems [38] and generalized the Nyquist–Shannon sampling theorem to signals in generator-bounded spaces [39]. Nevertheless, the learned latent space is often used simultaneously for reconstruction and propagation, creating a tension between spatial richness and dynamical regularity. Moreover, when temporal context within an input window is used to construct the latent representation [22], predictive supervision may allow the encoder to encode short-term advancement into the representation on which Φ operates, making the latent update less identifiable as a standalone propagation rule across rollout steps. This motivates a cleaner separation between time-consistent spatial representation and compact, structured propagation.

3 Architecture Desiderata

The preceding discussion shows that stable long-horizon forecasting requires more than fitting local transitions. We use the spectral structure of PDE evolution as a guiding principle for designing an explicitly organized recurrent propagator.

Spectral decomposition of the linear propagation operator. Consider the linear evolution equation $\partial_t u(x,t) = \mathcal{L}_x u(x,t)$ on the periodic domain $\Omega = \mathbb{T}^2$, where $\mathcal{L}_x$ is linear and translation-invariant in space. Under the spatial Fourier basis, the state decomposes into modes indexed by $k \in \mathbb{Z}^2$, and each modal coefficient $\hat{u}(k,t)$ evolves independently according to a finite-dimensional linear system $\partial_t \hat{u}(k,t) = A(k)\hat{u}(k,t)$, where $A(k) \in \mathbb{C}^{d_u \times d_u}$ is the matrix induced by $\mathcal{L}_x$ at frequency k. The finite-time propagation is $\hat{u}(k, t+\Delta t) = \exp(\Delta t A(k))\hat{u}(k,t)$. This decomposition reveals a

crucial structural bias: for linear systems, the natural propagation space is the spatial Fourier basis, where temporal evolution is decoupled into independent, low-dimensional subsystems [25, 40, 41]. While this exact decoupling holds only in the linear translation-invariant setting, it provides a useful template for organizing latent propagation in more general PDEs. It suggests that an appropriate spatial basis can expose an explicit and structured propagation law for PDE evolution.

Nonlinear coupling and finite-band closure. For general PDEs governed by Eq. (1), nonlinear terms induce interactions across Fourier modes and break the strict mode-wise independence of the linear case. For example, the Fourier coefficient of a quadratic term u^2 at mode k is given by $\widehat{u^2}(k) = \sum_{p+q=k} \hat{u}(p)\hat{u}(q)$, where p and q denote interacting modes. Thus, the evolution of a mode is generally affected by combinations of other modes, allowing energy transfer across spectral bands. Since practical propagators retain only a finite spectral band for tractability and stability, the retained dynamics are non-closed: unresolved modes can still affect the retained band through nonlinear coupling. This calls for a closure term that compensates for energy transfer and unresolved interactions between retained and truncated modes [14, 42–44]. Rather than reconstructing discarded modes, the closure refines retained spectral evolution and reduces spectral distortion during long rollout, thereby improving the consistency of recurrent propagation.

Synthesizing these insights, we model $\mathcal{G}$ through three coupled objectives: (i) constructing a time-consistent spatial representation space shared across all time slices; (ii) projecting this rich spatial space to a compact propagation state to reduce the conflict between reconstruction fidelity and rollout regularity; (iii) structuring the latent propagator Φ in this compact state explicitly as a modal evolution rule, with a frequency-conditioned linear backbone for retained modes and a nonlinear closure for coupled and truncated interactions.

Consequently, we factorize the learned operator as $\widehat{\mathcal{G}} := \mathcal{D} \circ \mathcal{R} \circ \Phi \circ \mathcal{P} \circ \mathcal{E}$, where $\mathcal{E}$, $\mathcal{P}$, Φ, $\mathcal{R}$, and $\mathcal{D}$ instantiate spatial-basis analysis, projection to a compact propagation state, structured spectral propagation, inverse projection, and physical-field synthesis, respectively.

4 Structured Spectral Propagator Learning

Problem formulation. Let $u_n := u(\cdot, t_n) \in \mathcal{U}$ denote the solution state at time t_n. For a fixed sequence length T, we denote the state sequence starting at t_n by $\mathbf{u}_n := (u_n, \ldots, u_{n+T-1})$. The PDE evolution induces a finite-time propagator $\mathcal{G}_T$, with $\mathbf{u}_{n+T} = \mathcal{G}_T(\mathbf{u}_n)$. We approximate this propagator via five maps: $\widehat{\mathcal{G}}_T := \mathcal{D} \circ \mathcal{R} \circ \Phi \circ \mathcal{P} \circ \mathcal{E}$.

Encoder. The encoder $\mathcal{E}$ constructs a shared spatial representation from physical solution states while preserving fine-grained spatial details needed for later synthesis. Given a temporal state sequence $\mathbf{u}_n$, the encoder is applied to each time slice, yielding $\mathbf{h}_n = (h_n, \ldots, h_{n+T-1})$, where $h_j = \mathcal{E}(\hat{u}_j) \in \mathcal{H} \subset \mathbb{R}^{C_s \times n_x \times n_y}$. This frame-wise design prevents the representation space $\mathcal{H}$ from entangling spatial features with short-range temporal advancement. As a result, $\mathcal{H}$ serves as a spatial observation space, leaving temporal evolution modeled explicitly by the structured propagator Φ.

In implementation, each grid-based state $u_j \in \mathbb{R}^{d_u \times N_x \times N_y}$ is concatenated with a normalized coordinate grid $\gamma \in \mathbb{R}^{2 \times N_x \times N_y}$, yielding $\hat{u}_j \in \mathbb{R}^{(d_u+2) \times N_x \times N_y}$. A local convolutional lift maps $\hat{u}_j$ to C_s feature channels and optionally reduces the resolution to $n_x \times n_y$, extracting local geometry and cross-variable correlations from the physical field. Cascaded factorized spectral blocks then apply one-dimensional Fourier transforms along each spatial axis, and mix retained modes across channels, providing lightweight spectral calibration. A final channel attention module reweights feature channels according to their global relevance, yielding $h_j = \mathcal{E}(\hat{u}_j) \in \mathcal{H}$. Thus, the encoder provides a rich and frequency-aware spatial representation for subsequent projection and structured propagation, while preserving fine-grained spatial information for physical-field synthesis.

Projector. The projector $\mathcal{P}$ maps the rich spatial representation $h_j \in \mathcal{H}$ to a compact propagation state $z_j \in \mathcal{Z}$. Its role is to separate the spatial details for synthesis from the dynamically focused state used for recurrent dynamics, mitigating the conflict between reconstruction fidelity and propagation regularity. In implementation, $\mathcal{P}$ is a pointwise channel projection implemented by a 1×1 convolution. Given $h_j \in \mathbb{R}^{C_s \times n_x \times n_y}$, $\mathcal{P}$ maps it to $z_j = \mathcal{P}(h_j) \in \mathbb{R}^{C_z \times n_x \times n_y}$, where $C_z < C_s$. Thus, $\mathcal{P}$ preserves the latent spatial resolution while reducing the channel dimension

from C_s to C_z. This channel bottleneck suppresses redundant representation degrees of freedom and encourages the propagated state to retain dynamically relevant features. Applied frame-wise to $\mathbf{h}_n = (h_n, \dots, h_{n+T-1})$, it yields the compact state sequence $\mathbf{z}_n = (z_n, \dots, z_{n+T-1})$ on which the structured propagator operates. Preserving the latent spatial grid is important for the subsequent Fourier-domain modal update and avoids discarding spatial phase information before rollout.

Propagator. The propagator Φ advances the compact sequence $\mathbf{z}_n = (z_n, \dots, z_{n+T-1})$ to its future sequence $\tilde{\mathbf{z}}_{n+T} = \Phi(\mathbf{z}_n)$ in the propagation space $\mathcal{Z}$, where $z_j \in \mathbb{R}^{C_z \times n_x \times n_y}$. To impose spectral organization on the update, we apply the spatial Fourier transform $\mathcal{F}_s$ and retain two frequency blocks Λ^+ and Λ^- in the rFFT2 layout. Each block contains $m_x \times m_y$ coefficients. For $b \in \{+, -\}$, we define $\widehat{\mathbf{z}}^b_{n,\Lambda} = \Pi_{\Lambda^b} \mathcal{F}_s(\mathbf{z}_n) \in \mathbb{C}^{T \times C_z \times m_x \times m_y}$. For each retained mode k and channel ℓ, the temporal modal vector $q^{(k,\ell)} \in \mathbb{C}^T$ collects the coefficients across the T input time slices. Both blocks are evolved using the shared linear backbone K_θ and nonlinear spectral closure G_ψ.

Frequency-conditioned linear backbone. The linear backbone K_θ models the dominant temporal evolution of retained spectral modes. Motivated by the modal organization of spectral PDE evolution mentioned above, we parameterize the latent update as mode-wise linear temporal dynamics whose action is adaptively modulated by frequency. For each latent channel ℓ, a single temporal evolution matrix $\bar{K}_\ell \in \mathbb{C}^{T \times T}$ is shared across all retained modes, while mode-specific behavior is introduced through a bounded frequency-dependent gate, $K_\theta^{(k,\ell)} q = M_\theta(k, \ell) \bar{K}_\ell q$, where $M_\theta(k, \ell) = 1 + \beta \tanh(a_\theta(\xi(k))_\ell)$. Here, $\xi(k) = (k_x, k_y, \|k\|, \cos\theta_k, \sin\theta_k)$, $\theta_k = \operatorname{atan2}(k_y, k_x)$, and a_θ is a shared MLP. The feature $\xi(k)$ encodes both the Cartesian position and polar geometry of the spatial frequency, allowing the same temporal backbone to adapt across wavelengths and orientations. This yields a parameter-efficient backbone design: instead of learning a separate $T \times T$ matrix for every frequency-channel pair, it uses only C_z temporal matrices together with a shared gating network. The bounded form $1 + \beta \tanh(\cdot)$ limits unconstrained frequency-dependent deviations during recurrent rollout, promoting more consistent modal updates across the retained spectral band.

To regularize the temporal backbone during rollout, we impose the normality penalty $\mathcal{L}_{\text{norm}} = \frac{1}{C_z} \sum_{\ell=1}^{C_z} \|\bar{K}_\ell \bar{K}_\ell^H - \bar{K}_\ell^H \bar{K}_\ell\|_F^2$. This penalty encourages each temporal evolution matrix $\bar{K}_\ell$ to approach a normal matrix, for which repeated action is controlled by eigenvalues through unitary diagonalization. By discouraging transient amplification induced by non-orthogonal eigenvectors, it improves the conditioning of autoregressive temporal updates.

Nonlinear spectral closure. The closure G_ψ compensates for residual effects not represented by the linear modal backbone, including unresolved nonlinear coupling and energy transfer among retained modes, and finite-band truncation effects. It operates on the retained spectral block $\widehat{\mathbf{z}}_{n,\Lambda}$. In implementation, we split the complex coefficients into real and imaginary parts, concatenate time and channel axes, and apply a shallow convolutional network $\mathcal{C}_\psi$ over the retained frequency plane (m_x, m_y). The convolutional structure aggregates nearby retained modes, while the merged temporal-channel dimension provides joint time and channel context. We initialize G_ψ near zero so that early training is dominated by the linear backbone. To encourage complementary updates, we define the local backbone–closure alignment penalty as $\ell_{\text{orth}}(\Delta_K, \Delta_G) = \frac{|\langle \Delta_K, \Delta_G \rangle|^2}{(\|\Delta_K\|_2^2 + \epsilon)(\|\Delta_G\|_2^2 + \epsilon)}$, where $\Delta_K = K_\theta(\widehat{\mathbf{z}}_{n,\Lambda}) - \widehat{\mathbf{z}}_{n,\Lambda}$ and $\Delta_G = G_\psi(\widehat{\mathbf{z}}_{n,\Lambda})$. This discourages G_ψ from duplicating the linear update and encourages it to focus on residual nonlinear and truncation-induced effects.

Residual spectral update. Rather than using a single unconstrained transition on the retained spectral block, we formulate propagation as the discretization of a learned latent flow in the retained spectral space [45]. Starting from $\widehat{\mathbf{z}}^{(0)}_{n,\Lambda} = \widehat{\mathbf{z}}_{n,\Lambda}$, we perform N_{sub} residual substeps:

$$\widehat{\mathbf{z}}^{(s+1)}_{n,\Lambda} = \widehat{\mathbf{z}}^{(s)}_{n,\Lambda} + \Delta\tau \left[\alpha\big(K_\theta(\widehat{\mathbf{z}}^{(s)}_{n,\Lambda}) - \widehat{\mathbf{z}}^{(s)}_{n,\Lambda}\big) + \lambda_g G_\psi(\widehat{\mathbf{z}}^{(s)}_{n,\Lambda}) \right], \qquad s = 0, \dots, N_{\text{sub}} - 1.$$

This update can be viewed as an explicit Euler discretization of the latent-time dynamics $\partial_\tau \widehat{\mathbf{z}}_\Lambda = \alpha(K_\theta(\widehat{\mathbf{z}}_\Lambda) - \widehat{\mathbf{z}}_\Lambda) + \lambda_g G_\psi(\widehat{\mathbf{z}}_\Lambda)$, where τ is an internal evolution variable in the latent spectral space. Accordingly, $\Delta\tau$ is the latent integration step size, while $N_{\text{sub}}\Delta\tau$ controls the total internal propagation depth. The backbone term provides the dominant mode-wise linear increment, and the closure supplies a state-dependent nonlinear correction. In our experiments, α and λ_g are fixed scalar weights, set to $\alpha = 1$ and $\lambda_g = 1$. After the final substep, the updated retained block is inserted back into the spectral representation and transformed to $\mathcal{Z}$ by inverse Fourier transform, yielding $\tilde{\mathbf{z}}_{n+T} = \Phi(\mathbf{z}_n)$.

Inverse Projector. The inverse projector $\mathcal{R}$ lifts the propagated compact state back to the spatial representation space before physical-field synthesis. Given the propagated compact sequence $\tilde{\mathbf{z}}_{n+T} = (\tilde{z}_{n+T}, \ldots, \tilde{z}_{n+2T-1})$, the reconstructor is applied to each state as $\tilde{h}_j = \mathcal{R}(\tilde{z}_j)$, yielding $\tilde{\mathbf{h}}_{n+T} = (\tilde{h}_{n+T}, \ldots, \tilde{h}_{n+2T-1})$. In implementation, $\mathcal{R}$ is a 1×1 convolution that maps $\tilde{z}_j \in \mathbb{R}^{C_z \times n_x \times n_y}$ to $\tilde{h}_j \in \mathbb{R}^{C_s \times n_x \times n_y}$. Thus, $\mathcal{R}$ restores the channel capacity required by the decoder while preserving the latent spatial resolution.

Decoder. The decoder $\mathcal{D}$ maps the lifted representation $\tilde{h}_j \in \mathcal{H}$ back to the physical solution variables. Applied frame-wise to $\tilde{\mathbf{h}}_{n+T}$, it yields the predicted sequence $\tilde{\mathbf{u}}_{n+T} = (\tilde{u}_{n+T}, \ldots, \tilde{u}_{n+2T-1})$, where $\tilde{u}_j = \mathcal{D}(\tilde{h}_j)$. In implementation, $\mathcal{D}$ combines channel attention, a factorized spectral branch, a local convolutional branch, and a readout head. Channel attention first recalibrates the lifted feature channels. The spectral branch synthesizes global structure, while the local branch refines spatial details and residual high-frequency components not explicitly evolved in the compact spectral state. The readout head maps the refined features to d_u physical channels, yielding $\tilde{u}_j \in \mathbb{R}^{d_u \times N_x \times N_y}$.

Training objective. The model is trained by autoregressive rollout on temporal state sequences. Let $\mathbf{u}^{(r)} := \mathbf{u}_{n+rT}$ denote the ground-truth sequence at rollout step r, and let N_{roll} be the number of training rollout steps. Starting from $\tilde{\mathbf{u}}^{(0)} = \mathbf{u}^{(0)}$, we define the coordinate-augmented input $\hat{\mathbf{u}}^{(r)} = \mathrm{concat}(\tilde{\mathbf{u}}^{(r)}, \gamma)$. The model recursively computes $\mathbf{z}_{\mathrm{in}}^{(r)} = \mathcal{P}(\mathcal{E}(\hat{\mathbf{u}}^{(r)}))$, $\tilde{\mathbf{z}}^{(r+1)} = \Phi(\mathbf{z}_{\mathrm{in}}^{(r)})$, and $\tilde{\mathbf{u}}^{(r+1)} = \mathcal{D}(\mathcal{R}(\tilde{\mathbf{z}}^{(r+1)}))$ for $r = 0, \ldots, N_{\mathrm{roll}} - 1$. For latent supervision, the target compact state is $\mathbf{z}_\star^{(r+1)} = \mathcal{P}(\mathcal{E}(\mathrm{concat}(\mathbf{u}^{(r+1)}, \gamma)))$. The objective combines reconstruction, latent rollout, and physical rollout losses to constrain prediction errors in both the compact latent space and the physical solution space. The training objective is

$$\mathcal{L} = \lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} + \frac{1}{N_{\mathrm{roll}}} \sum_{r=0}^{N_{\mathrm{roll}}-1} \left(\lambda_{\mathrm{lat}}\mathcal{L}_{\mathrm{lat}}^{(r)} + \lambda_{\mathrm{phy}}\mathcal{L}_{\mathrm{phy}}^{(r)} \right) + \lambda_{\mathrm{norm}}\mathcal{L}_{\mathrm{norm}} + \lambda_{\mathrm{orth}}\mathcal{L}_{\mathrm{orth}}. \tag{2}$$

The orthogonality loss $\mathcal{L}_{\mathrm{orth}} = \sum_{r=0}^{N_{\mathrm{roll}}-1} \ell_{\mathrm{orth}}^{(r)}$ accumulates local penalties across rollout windows, where each $\ell_{\mathrm{orth}}^{(r)}$ is averaged over the retained frequency blocks and Euler substeps. The reconstruction loss $\mathcal{L}_{\mathrm{rec}} = \mathrm{MSE}\left(\mathcal{D}(\mathcal{E}(\hat{\mathbf{u}}^{(0)})), \mathbf{u}^{(0)}\right)$ preserves spatial information through the encoder–decoder path without temporal propagation. The latent rollout loss $\mathcal{L}_{\mathrm{lat}}^{(r)} = \mathrm{MSE}\left(\tilde{\mathbf{z}}^{(r+1)}, \mathbf{z}_\star^{(r+1)}\right)$ aligns the propagated compact sequence with the encoded target sequence. The physical rollout loss $\mathcal{L}_{\mathrm{phy}}^{(r)} = \mathrm{MSE}\left(\tilde{\mathbf{u}}^{(r+1)}, \mathbf{u}^{(r+1)}\right)$ constrains prediction error in the solution space. The regularizers $\mathcal{L}_{\mathrm{norm}}$ and $\mathcal{L}_{\mathrm{orth}}$ are defined in the propagator, controlling the normality of the temporal backbone and the alignment between the backbone and closure increments, respectively. We set fixed weights $\lambda_{\mathrm{rec}} = 1$, $\lambda_{\mathrm{lat}} = 1$, $\lambda_{\mathrm{phy}} = 1$, and $\lambda_{\mathrm{norm}} = \lambda_{\mathrm{orth}} = 0.01$ in all experiments.

5 Experiments

We run experiments across different dynamical regimes to assess whether the proposed framework improves stability and spectral consistency under recursive rollout. (i) **Long-horizon forecasting accuracy.** We evaluate autoregressive prediction on Shallow–Water (SW), Reaction–Diffusion (RD), and Navier–Stokes (NS) benchmarks [46, 6], comparing against representative spectral and latent neural operator baselines in terms of trajectory accuracy, worst-case local error, and spectral fidelity. (ii) **Temporal extrapolation.** We test whether the learned propagator remains reliable beyond the supervised training horizon, indicating a reusable evolution rule rather than fit to the observed rollout range. (iii) **Disentangling local accuracy and rollout robustness.** We compare teacher-forced one-step predictions with extended 140-step autoregressive extrapolation to assess whether long-horizon gains can be explained by local prediction accuracy. (iv) **Ablation studies.** We assess the contributions of K_θ, G_ψ, $\mathcal{P}, \mathcal{R}$, frame-wise encoding, spectral encoder–decoder design, and individual training losses. For the standard long-horizon forecasting and temporal extrapolation benchmarks, SSP results are averaged over three independent random seeds, with standard deviations provided in Appendix D.4.

Table 1: Quantitative comparison on long-horizon PDE benchmarks. Best results are in **bold**, and second-best results are underlined. - indicates that the corresponding model failed to train under this setting.

Method	Navier–Stokes			Shallow–Water			Reaction–Diffusion		
	L2	$E_{\max}$	f_{low}	**L2**	$E_{\max}$	f_{low}	**L2**	$E_{\max}$	f_{low}
FNO	1.42E-1	6.68E-1	5.94E-2	5.10E-3	8.00E-2	3.37E-4	2.57E-1	6.72E-2	1.28E-2
U-FNO	1.50E-1	1.44E+0	2.38E-2	4.99E-3	6.04E-2	1.36E-3	1.46E-2	3.53E-2	1.88E-4
F-FNO	7.57E-2	6.87E-1	1.09E-2	9.28E-4	1.69E-2	6.99E-5	1.52E-2	2.02E-2	2.69E-4
UNO	1.36E-1	1.19E+0	2.16E-2	1.76E-3	1.71E-2	7.40E-5	8.38E-2	3.29E-1	4.18E-3
KNO	7.77E-2	**3.82E-1**	5.60E-2	9.76E-3	3.12E-2	3.89E-3	2.51E-1	6.85E-2	1.09E-2
LNO	1.32E-1	1.16E+0	2.18E-2	2.31E-3	2.11E-2	6.12E-5	-	-	-
CALM	8.82E-2	9.96E-1	1.40E-2	2.25E-3	2.41E-2	3.61E-5	1.72E-1	2.30E-1	5.40E-4
Ours	**5.50E-2**	5.38E-1	**5.49E-3**	**6.56E-4**	**9.72E-3**	**2.53E-5**	**7.46E-3**	**6.71E-3**	**1.48E-4**

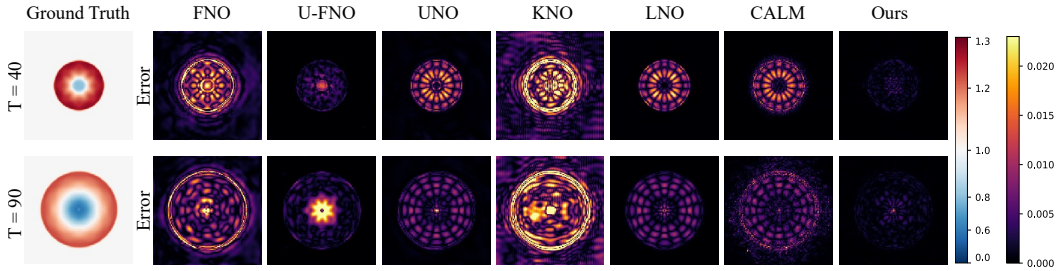


Figure 2: Ground-truth states and absolute-error maps on the Shallow–Water benchmark at rollout steps $t = 40$ and $t = 90$. Leftmost column: ground truth. Remaining columns: pointwise absolute-error maps for different methods.

5.1 Long-horizon forecasting accuracy

We first evaluate long-horizon forecasting accuracy to assess whether a learned propagator remains stable under repeated composition, rather than only fitting short local transitions. We consider the three PDE benchmarks introduced above, covering different dynamical characteristics. We refer to Appendix B for dataset details. All models except F-FNO use 10 input frames on SW and RD and 4 on NS, whereas F-FNO uses only the final frame of the corresponding history window [28]. All methods share the same forecast origin and autoregressively predict 90 future frames on SW and RD and 12 on NS. We report metrics evaluated over the specified rollout horizon, including relative L_2 error for global trajectory accuracy, $E_{\max}$ for worst-case pointwise deviation, and f_{low} for preservation of dominant low-frequency spectral modes.

Table 1 compares our method with representative spectral and latent neural operator baselines. On SW and RD, our method achieves the best results across all reported metrics, showing consistent gains in global accuracy, worst-case local error, and low-frequency spectral fidelity. On NS, it obtains the lowest L_2 and f_{low}, while KNO gives a lower $E_{\max}$; however, KNO has substantially larger global and spectral errors, indicating weaker rollout consistency. The full spectral evaluation in Appendix D.1 further shows that our method also achieves the best mid- and high-frequency errors across all three benchmarks, suggesting that the structured propagator preserves dominant dynamics without sacrificing finer spectral components. To examine error accumulation under a substantially longer NS rollout, we additionally evaluate a 70-step autoregressive setting at $\nu = 10^{-5}$, with SSP and three competitive baselines retrained using 10 input frames (one for F-FNO). SSP achieves the lowest errors across all metrics, with complete experimental details provided in Appendix D.1.

Figure 2 presents ground-truth states and absolute-error maps on SW at rollout steps $t = 40$ and $t = 90$, corresponding to intermediate and late autoregressive prediction stages. This benchmark requires the model to preserve the radial envelope, moving wavefront, and global symmetry as errors accumulate over time. At $t = 40$, several baselines already exhibit structured residual patterns, including ringing artifacts, lattice-like errors, and deviations concentrated near the wavefront, whereas our method maintains weaker and more localized errors. By $t = 90$, these artifacts become more

Table 2: Temporal extrapolation results on the Shallow–Water benchmark. Errors are computed only over rollout steps 51–90, which lie beyond the supervised training horizon. Best results are in **bold**, and second-best results are underlined.

Method	L2	$E_{\max}$	BRMS	f_{low}	f_{mid}	f_{high}	f_{mse}
FNO	1.83E+02	3.29E+02	4.86E+01	1.16E+01	7.30E+00	6.63E-01	2.17E+00
U-FNO	1.28E+00	1.02E+01	1.37E-02	2.40E-01	8.51E-02	8.07E-03	3.22E-02
F-FNO	**4.36E-02**	4.69E-01	6.51E-03	**7.71E-03**	5.04E-03	1.68E-03	2.48E-03
UNO	3.56E+11	4.21E+11	1.47E+11	6.01E+10	9.50E+09	1.26E+09	5.97E+09
KNO	8.19E-02	9.96E-01	1.90E-02	4.71E-02	1.94E-02	8.99E-03	1.27E-02
LNO	1.71E+00	2.96E+02	1.13E+01	2.43E+00	1.05E+00	1.18E+00	1.24E+00
CALM	6.41E-02	3.68E-01	2.17E-02	1.56E-02	5.36E-03	7.98E-04	2.29E-03
Ours	4.49E-02	**2.55E-01**	**5.46E-04**	1.24E-02	**2.02E-03**	**2.59E-04**	**1.24E-03**

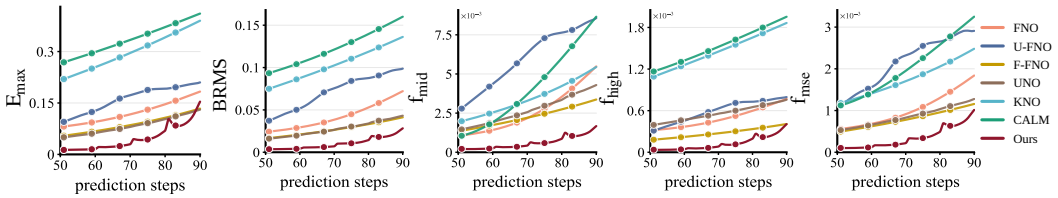


Figure 3: Temporal extrapolation error curves for channel u on the Reaction–Diffusion benchmark over rollout steps 51–90. Each curve reports the mean error at each extrapolation step; lower values indicate smaller prediction errors.

pronounced for the baselines, indicating progressive degradation of wavefront location and radial morphology under recursive rollout. In contrast, our method produces the smallest-magnitude and least structured error field, suggesting better control of error accumulation and spatial error organization. These visual observations are consistent with Table 1. Additional error visualizations on the remaining benchmarks are provided in the Appendix D.1.

5.2 Temporal extrapolation beyond the training horizon

We next evaluate temporal extrapolation on unseen test trajectories beyond the supervised rollout range. In this standard benchmark setting, all models use the input configurations specified above and are trained with 50 supervised prediction steps. At test time, all methods autoregressively predict the same 90 future frames, with errors evaluated on prediction steps 51–90. We report relative L_2, $E_{\max}$, and BRMS for global, local, and boundary errors, together with f_{low}, f_{mid}, f_{high}, and f_{mse} for spectral fidelity.

Table 2 presents the SW extrapolation results. FNO, UNO, and LNO exhibit substantially larger errors beyond the supervised horizon. F-FNO achieves the lowest relative L_2 and f_{low} errors, indicating strong global accuracy and dominant-mode fidelity. In comparison, SSP achieves the lowest $E_{\max}$, BRMS, f_{mid}, f_{high}, and f_{mse} errors, demonstrating advantages in local accuracy, boundary consistency, and finer-scale spectral fidelity. CALM achieves the second-lowest f_{mse}, followed by F-FNO. We additionally evaluate the fully coupled three-variable SW system, jointly predicting water depth h and horizontal momenta hu and hv. With all models retrained under the same extrapolation protocol, SSP reduces relative L_2 errors by 25.4%, 34.0%, and 26.1% for the three variables, respectively, compared with the strongest baseline for each variable over steps 51–90. Full results are provided in Appendix D.2.

Figure 3 further shows step-wise extrapolation errors on RD over the same beyond-horizon interval. Across $E_{\max}$, BRMS, f_{mid}, f_{high}, and f_{mse}, our method maintains the lowest error curves and the consistently lower error growth throughout the rollout. The step-wise curves confirm that the improvement persists throughout the extrapolation interval, rather than being an artifact of averaging. Full numerical metrics in both physical and spectral domains are reported in Table 30.

Table 3: Ablation study on Navier–Stokes, Shallow–Water, and Reaction–Diffusion. w/o denotes removing the corresponding component from the full model. Best results are in **bold**, and second-best results are underlined.

Method	Navier–Stokes			Shallow–Water			Reaction–Diffusion		
	L2	$E_{\max}$	f_{low}	**L2**	$E_{\max}$	f_{low}	**L2**	$E_{\max}$	f_{low}
Full	**5.50E-2**	**5.38E-1**	**5.49E-3**	**6.56E-4**	**9.72E-3**	**2.53E-5**	**7.46E-3**	**6.71E-3**	**1.48E-4**
w/o K_θ	6.20E-2	6.48E-1	6.41E-3	9.09E-4	1.30E-2	8.50E-5	1.43E-2	1.15E-2	4.80E-4
w/o G_ψ	6.04E-2	5.93E-1	5.92E-3	1.78E-3	2.16E-2	7.70E-5	1.12E-2	8.12E-3	2.90E-4
w/o $\mathcal{P}, \mathcal{R}$	6.07E-2	6.21E-1	6.11E-3	1.05E-3	1.38E-2	4.90E-5	2.41E-2	2.20E-2	1.40E-3
$T \to C$ Enc	8.35E-2	9.43E-1	8.18E-3	2.06E-3	3.36E-2	1.01E-4	6.41E-2	3.99E-2	2.24E-3
Conv Enc-Dec	1.53E-1	1.48E+0	2.57E-2	1.08E-3	1.76E-2	5.50E-5	1.95E-2	1.68E-2	8.50E-4

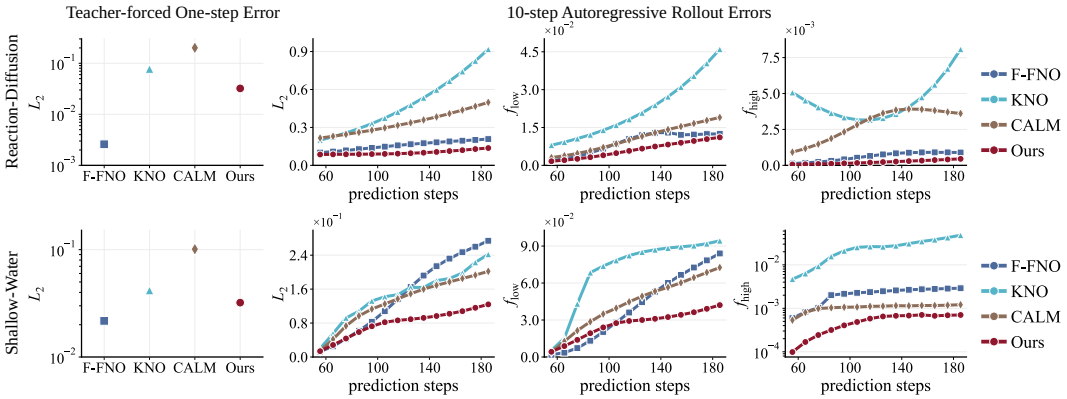


Figure 4: One-step prediction accuracy and autoregressive rollout errors on Reaction–Diffusion (top) and Shallow–Water (bottom). Left: teacher-forced one-step relative L_2 errors averaged over 140 extrapolation steps. Remaining columns: free-running L_2, f_{low}, and f_{high} errors evaluated over consecutive 10-step intervals (prediction steps 51–190). Logarithmic vertical scales are used for the one-step panels and the Shallow–Water f_{high} panel.

5.3 Disentangling Local Accuracy and Rollout Robustness

To examine whether long-horizon improvements arise from more accurate local predictions or greater robustness under repeated propagation, we extend the extrapolation horizon from 40 to 140 steps on independently simulated SW and RD test trajectories. All methods retain their original checkpoints trained for the 50-step prediction task. We compare teacher-forced one-step predictions using ground-truth contexts with free-running autoregressive rollouts without ground-truth correction. Detailed evaluation protocols are provided in Appendix D.2.

Figure 4 compares teacher-forced one-step accuracy with autoregressive rollout errors evaluated over consecutive 10-step intervals. Although F-FNO achieves the lowest one-step L_2 errors on both systems, this advantage does not persist under extended autoregressive prediction. On RD, SSP maintains the lowest L_2, f_{low}, and f_{high} errors throughout the evaluated horizon, with smaller error increases under repeated propagation. On SW, F-FNO initially achieves lower L_2 and f_{low} errors, but SSP progressively overtakes it as the rollout extends, while consistently maintaining the lowest high-frequency error. This crossover highlights the distinction between local prediction accuracy and the accumulation of errors under recursive feedback.

The spectral results further indicate that SSP better preserves both dominant and fine-scale structures over extended horizons. In particular, its sustained advantage in f_{high} suggests reduced high-frequency distortion during repeated propagation. The SW f_{high} panel uses a logarithmic vertical scale, so its visual slope reflects changes in log-error rather than the absolute error growth rate. Over the final 50 extrapolation steps, SSP reduces relative L_2 by 39.4% on SW and 37.9% on RD compared with the strongest baseline for each system. Together, these findings provide empirical evidence that SSP's long-horizon gains cannot be explained solely by one-step prediction accuracy and are associated with improved robustness under recursive propagation.

5.4 Ablation studies

We conduct ablation studies on NS, SW, and RD to assess the contributions of the analysis–propagation–synthesis components. We independently remove the linear modal backbone K_θ, nonlinear spectral closure G_ψ, and projector pair $\mathcal{P}, \mathcal{R}$. We further examine two architectural variants: $T \to C$ Enc, which stacks temporal inputs along the channel dimension before encoding, and Conv Enc-Dec, which replaces the spectral-factorized encoder–decoder with convolutional modules under the same resolution schedule.

Table 3 shows that the full model achieves the lowest errors across all reported metrics. Removing either K_θ or G_ψ degrades physical-space and spectral accuracy, supporting their complementary roles in modal evolution and nonlinear correction. The relative contributions vary across systems: removing G_ψ causes greater degradation in relative L_2 on SW, whereas removing K_θ has a larger effect on RD. Removing $\mathcal{P}, \mathcal{R}$ also increases errors across all benchmarks, supporting the benefit of separating spatial representation from compact temporal propagation. The $T \to C$ Enc variant performs substantially worse, highlighting the importance of frame-wise spatial encoding followed by explicit propagation in Φ. Replacing the spectral encoder–decoder leads to some performance degradation, particularly on NS and RD, while the convolutional variant remains competitive with several representative baselines on SW and RD, suggesting that the structured propagation mechanism retains its effectiveness with alternative analysis–synthesis designs.

We further examine the training objectives by independently removing $\mathcal{L}_{\mathrm{phy}}$, $\mathcal{L}_{\mathrm{lat}}$, and $\mathcal{L}_{\mathrm{orth}}$ across long-horizon forecasting and temporal extrapolation. Removing $\mathcal{L}_{\mathrm{phy}}$ causes the largest degradation, confirming its central role in physical-space rollout supervision. The latent rollout loss provides additional benefits, particularly for extrapolation, while the orthogonality loss yields smaller but consistent improvements by regularizing the backbone–closure updates. Detailed results are provided in Appendix D.3 (Tables 35 and 36).

6 Conclusion

This work proposes SSP for long-horizon PDE forecasting, an analysis–propagation–synthesis framework that separates time-consistent spatial representation, compact propagation, and physical-field synthesis. SSP structures latent spectral evolution with a frequency-conditioned linear backbone and a nonlinear closure over retained modes. Experiments on three PDE benchmarks demonstrate improved long-horizon accuracy, spectral preservation, and beyond-horizon robustness, supporting structured latent spectral propagation as an effective principle for long-term physical field prediction. Ablation studies further validate the complementary contributions of compact projection, structured modal evolution, and nonlinear spectral correction to accurate recursive forecasting. Additional evaluations on coupled physical fields, higher spatial resolutions, and distribution shifts demonstrate the effectiveness of SSP across the tested physical and numerical settings.

The current implementation relies on regular spatial grids and does not directly accommodate irregular meshes or sparse spatial observations. Moreover, SSP does not explicitly enforce physical conservation laws, boundary conditions, or spectral-radius constraints, leaving stronger guarantees of long-horizon stability for future work.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant numbers T2525012, U25B6013, and 92467301). The authors declare no competing financial interests.

References

[1] Y. Yuan, X. Tang, W. Zhou, W. Pan, X. Li, H.-T. Zhang, H. Ding, and J. Goncalves, "Data driven discovery of cyber physical systems," *Nature communications*, vol. 10, no. 1, p. 4894, 2019.

[2] E. De Bézenac, A. Pajot, and P. Gallinari, "Deep learning for physical processes: Incorporating prior scientific knowledge," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124009, 2019.

[3] S. L. Brunton and J. N. Kutz, "Machine learning for partial differential equations," *arXiv preprint arXiv:2303.17078*, 2023.

[4] L. Lu, P. Jin, and G. E. Karniadakis, "Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators," *arXiv preprint arXiv:1910.03193*, 2019.

[5] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Learning maps between function spaces with applications to pdes," *Journal of Machine Learning Research*, vol. 24, no. 89, pp. 1–97, 2023.

[6] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," *arXiv preprint arXiv:2010.08895*, 2020.

[7] B. Raonic, R. Molinaro, T. De Ryck, T. Rohner, F. Bartolucci, R. Alaifari, S. Mishra, and E. De Bézenac, "Convolutional neural operators for robust and accurate learning of pdes," *Advances in Neural Information Processing Systems*, vol. 36, pp. 77187–77200, 2023.

[8] Z. Hao, Z. Wang, H. Su, C. Ying, Y. Dong, S. Liu, Z. Cheng, J. Song, and J. Zhu, "Gnot: A general neural operator transformer for operator learning," in *International conference on machine learning*, pp. 12556–12569, PMLR, 2023.

[9] Z. Sun, Y. Yang, and S. Yoo, "A neural pde solver with temporal stencil modeling," in *International Conference on Machine Learning*, pp. 33135–33155, PMLR, 2023.

[10] J. Brandstetter, D. Worrall, and M. Welling, "Message passing neural pde solvers," *arXiv preprint arXiv:2202.03376*, 2022.

[11] W. Xu, Y. Lu, and L. Wang, "Transfer learning enhanced deeponet for long-time prediction of evolution equations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 10629–10636, 2023.

[12] Z. Ye, C.-S. Zhang, and W. Wang, "Recurrent neural operators: Stable long-term pde prediction," *arXiv preprint arXiv:2505.20721*, 2025.

[13] J. Mikhaeil, Z. Monfared, and D. Durstewitz, "On the difficulty of learning chaotic dynamics with rnns," *Advances in neural information processing systems*, vol. 35, pp. 11297–11312, 2022.

[14] P. Lippe, B. Veeling, P. Perdikaris, R. Turner, and J. Brandstetter, "Pde-refiner: Achieving accurate long rollouts with neural pde solvers," *Advances in Neural Information Processing Systems*, vol. 36, pp. 67398–67433, 2023.

[15] Z. Wu, S. Zhang, S. He, S. Wang, M. Zhu, A. Jiao, L. Lu, and D. van Dijk, "Coast: Intelligent time-adaptive neural operators," in *2nd AI for Math Workshop@ ICML 2025*, 2025.

[16] T. Wu, T. Maruyama, and J. Leskovec, "Learning to accelerate partial differential equations via latent global evolution," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2240–2253, 2022.

[17] Z. Li, S. Patil, F. Ogoke, D. Shu, W. Zhen, M. Schneier, J. R. Buchanan Jr, and A. B. Farimani, "Latent neural pde solver: A reduced-order modeling framework for partial differential equations," *Journal of Computational Physics*, vol. 524, p. 113705, 2025.

[18] T. Wang and C. Wang, "Latent neural operator for solving forward and inverse pde problems," *Advances in Neural Information Processing Systems*, vol. 37, pp. 33085–33107, 2024.

[19] L. Serrano, T. X. Wang, E. Le Naour, J.-N. Vittaut, and P. Gallinari, "Aroma: Preserving spatial structure for latent pde modeling with local neural fields," *Advances in Neural Information Processing Systems*, vol. 37, pp. 13489–13521, 2024.

[20] J. Hagnberger, D. Musekamp, and M. Niepert, "Calm-pde: Continuous and adaptive convolutions for latent space modeling of time-dependent pdes," *arXiv preprint arXiv:2505.12944*, 2025.

[21] B. Alkin, A. Fürst, S. Schmid, L. Gruber, M. Holzleitner, and J. Brandstetter, "Universal physics transformers: A framework for efficiently scaling neural operators," *Advances in Neural Information Processing Systems*, vol. 37, pp. 25152–25194, 2024.

[22] W. Xiong, X. Huang, Z. Zhang, R. Deng, P. Sun, and Y. Tian, "Koopman neural operator as a mesh-free solver of non-linear partial differential equations," *Journal of Computational Physics*, vol. 513, p. 113194, 2024.

[23] Y. Jin, A. Cong, L. Hou, Q. Gao, X. Ge, C. Zhu, Y. Feng, and J. Li, "Invertible koopman neural operator for data-driven modeling of partial differential equations," *arXiv preprint arXiv:2503.19717*, 2025.

[24] S. Peitz, H. Harder, F. Nüske, F. Philipp, M. Schaller, and K. Worthmann, "Equivariance and partial observations in koopman operator theory for partial differential equations," *arXiv preprint arXiv:2307.15325*, 2023.

[25] D. Rodriguez-Lara, I. Alvarez-Rios, and F. S. Guzman, "Numerical solution partial differential equations using the discrete fourier transform," *arXiv preprint arXiv:2412.12308*, 2024.

[26] G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson, "U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow," *Advances in Water Resources*, vol. 163, p. 104180, 2022.

[27] M. A. Rahman, Z. E. Ross, and K. Azizzadenesheli, "U-no: U-shaped neural operators," *arXiv preprint arXiv:2204.11127*, 2022.

[28] A. Tran, A. Mathews, L. Xie, and C. S. Ong, "Factorized fourier neural operators," *arXiv preprint arXiv:2111.13802*, 2021.

[29] P. Jin, S. Meng, and L. Lu, "Mionet: Learning multiple-input operators via tensor product," *SIAM Journal on Scientific Computing*, vol. 44, no. 6, pp. A3490–A3514, 2022.

[30] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar, "Physics-informed neural operator for learning partial differential equations," *ACM/IMS Journal of Data Science*, vol. 1, no. 3, pp. 1–27, 2024.

[31] Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar, "Fourier neural operator with learned deformations for pdes on general geometries," *Journal of Machine Learning Research*, vol. 24, no. 388, pp. 1–26, 2023.

[32] P. Y. Chen, J. Xiang, D. H. Cho, Y. Chang, G. Pershing, H. T. Maia, M. M. Chiaramonte, K. Carlberg, and E. Grinspun, "Crom: Continuous reduced-order modeling of pdes using implicit neural representations," *arXiv preprint arXiv:2206.02607*, 2022.

[33] B. Kramer, B. Peherstorfer, and K. E. Willcox, "Learning nonlinear reduced models from data with operator inference," *Annual Review of Fluid Mechanics*, vol. 56, no. 1, pp. 521–548, 2024.

[34] N. H. Dashtbayaz, H. Salehipour, A. Butscher, and N. Morris, "Physics-informed reduced order modeling of time-dependent pdes via differentiable solvers," *arXiv preprint arXiv:2505.14595*, 2025.

[35] K. Tiwari, N. Dutta, N. Krishnan, *et al.*, "Latent mamba operator for partial differential equations," *arXiv preprint arXiv:2505.19105*, 2025.

[36] S. A. McQuarrie, P. Khodabakhshi, and K. E. Willcox, "Nonintrusive reduced-order models for parametric partial differential equations via data-driven operator inference," *SIAM Journal on Scientific Computing*, vol. 45, no. 4, pp. A1917–A1946, 2023.

[37] N. Franco, A. Manzoni, and P. Zunino, "A deep learning approach to reduced order modelling of parameter dependent partial differential equations," *Mathematics of Computation*, vol. 92, no. 340, pp. 483–524, 2023.

[38] Z. Zeng, Z. Yue, A. Mauroy, J. Gonçalves, and Y. Yuan, "A sampling theorem for exact identification of continuous-time nonlinear dynamical systems," *IEEE Transactions on Automatic Control*, vol. 69, no. 12, pp. 8402–8417, 2024.

[39] Z. Zeng, J. Liu, and Y. Yuan, "A generalized nyquist-shannon sampling theorem using the koopman operator," *IEEE Transactions on Signal Processing*, vol. 72, pp. 3595–3610, 2024.

[40] L. N. Trefethen, *Spectral methods in MATLAB*. SIAM, 2000.

[41] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral methods*, vol. 285. Springer, 2006.

[42] P. Sagaut, *Large eddy simulation for incompressible flows: an introduction*. Springer, 2006.

[43] S. B. Pope, *Turbulent flows*. Cambridge university press, Cambridge, 2000.

[44] A. J. Chorin, A. P. Kast, and R. Kupferman, "Optimal prediction of underresolved dynamics," *Proceedings of the National Academy of Sciences*, vol. 95, no. 8, pp. 4094–4098, 1998.

[45] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[46] M. Takamoto, T. Praditia, R. Leiteritz, D. MacKinlay, F. Alesiani, D. Pflüger, and M. Niepert, "Pdebench: An extensive benchmark for scientific machine learning," *Advances in neural information processing systems*, vol. 35, pp. 1596–1611, 2022.

A Full Method and Architecture Details

A.1 Data Preparation and Preprocessing

For each PDE benchmark, raw trajectories are converted into fixed-length input windows and future rollout targets. Let $u_t \in \mathbb{R}^{d_u \times N_x \times N_y}$ denote the physical state at time t, where d_u is the number of solution variables and $N_x \times N_y$ is the spatial grid resolution. Given an input length T, the model input is $u_{t-T+1:t} := (u_{t-T+1}, \dots, u_t)$, and subsequent windows are used as autoregressive training targets. In batched form, the input tensor is $u_{\text{in}} \in \mathbb{R}^{B \times T \times d_u \times N_x \times N_y}$.

All physical variables are normalized per channel using statistics computed only from the training split. For each physical channel c, the mean μ_c and standard deviation σ_c are computed over all training trajectories, time indices, and spatial grid locations. The normalized field is $u_{t,c}^{\text{norm}} = (u_{t,c} - \mu_c)/(\sigma_c + \epsilon)$, where ϵ is used for numerical stability. The same μ_c and σ_c are fixed for validation and test trajectories, preventing information leakage from non-training data.

We append normalized spatial coordinates to every input frame. The coordinate channels are generated from the grid resolution by uniformly sampling $[0, 1]$ along each spatial axis, producing $\gamma_x, \gamma_y \in \mathbb{R}^{B \times T \times 1 \times N_x \times N_y}$ after broadcasting across batch and time. The encoder input is $\widehat{u}_{\text{in}} = \text{concat}(u_{\text{in}}^{\text{norm}}, \gamma_x, \gamma_y) \in \mathbb{R}^{B \times T \times (d_u+2) \times N_x \times N_y}$. The coordinate channels are used only as geometric conditioning variables; the model predicts only the d_u normalized physical variables, which are mapped back to physical units by inverse channel-wise normalization for evaluation and visualization.

A.2 Operator Factorization and End-to-End Data Flow

SSP implements a finite-window evolution map as the composition $\widehat{\mathcal{G}}_T = \mathcal{D} \circ \mathcal{R} \circ \Phi \circ \mathcal{P} \circ \mathcal{E}$. Given an input window $u_{t-T+1:t}$, the encoder $\mathcal{E}$ and projector $\mathcal{P}$ are applied frame-wise to obtain a compact state sequence. The structured propagator Φ is then applied once to the full compact sequence and returns the future compact sequence. Finally, the inverse projector $\mathcal{R}$ and decoder $\mathcal{D}$ are applied frame-wise to synthesize the predicted physical window $\widetilde{u}_{t+1:t+T}$. Thus, spatial analysis and synthesis are implemented per frame, whereas temporal advancement is isolated inside Φ.

A.2.1 Encoder $\mathcal{E}$

The encoder $\mathcal{E}$ is implemented as a frame-wise cascaded factorized-spectral network. Given the coordinate-augmented input $\widehat{u}_{\text{in}} \in \mathbb{R}^{B \times T \times (d_u+2) \times N_x \times N_y}$, the batch and time dimensions are merged before encoding, so the network operates on tensors of shape $(BT) \times (d_u + 2) \times N_x \times N_y$. This keeps representation construction frame-wise and leaves temporal coupling to the propagator Φ. The

Table 4: End-to-end data flow of SSP.

Map	Input	Output	Implementation role
$\mathcal{E}$	coordinate-augmented physical frame	shared spatial representation	frame-wise analysis encoder
$\mathcal{P}$	shared spatial representation	compact propagation state	pointwise projection to propagation coordinates
Φ	compact state sequence	future compact state sequence	sequence-wise structured spectral propagation
$\mathcal{R}$	propagated compact state	synthesis spatial representation	pointwise lifting to decoder coordinates
$\mathcal{D}$	synthesis spatial representation	predicted physical frame	frame-wise physical-field decoder

first layer is a padded 3×3 convolution with stride s_0, followed by the Gaussian Error Linear Unit (GELU) activation. It maps the coordinate-augmented physical variables to C_s latent channels and performs spatial downsampling when $s_0 > 1$.

The encoder body consists of $L_{\mathcal{E}}$ factorized spectral blocks. For an input feature map $X^{(b)} \in \mathbb{R}^{BT \times C_b \times H_b \times W_b}$, each block applies one-dimensional real-valued fast Fourier transforms (rFFTs) separately along the width and height axes. The width branch retains $m^{\text{eff}}_{\mathcal{E},\text{w}}$ modes and mixes them with learned complex weights $W^{(b)}_{\text{w}} \in \mathbb{C}^{C_b \times C_{b+1} \times m_{\mathcal{E},\text{w}}}$. The height branch is implemented by transposing the spatial axes, applying the same rFFT–mixing–inverse-rFFT routine, and using weights $W^{(b)}_{\text{h}} \in \mathbb{C}^{C_b \times C_{b+1} \times m_{\mathcal{E},\text{h}}}$. The two inverse-transformed responses are summed in the latent spatial domain. At each resolution, the retained modes are clipped by the available Fourier frequencies: $m^{\text{eff}}_{\mathcal{E},\text{w}} = \min(m_{\mathcal{E},\text{w}}, \lfloor W_b/2 \rfloor + 1)$ and $m^{\text{eff}}_{\mathcal{E},\text{h}} = \min(m_{\mathcal{E},\text{h}}, \lfloor H_b/2 \rfloor + 1)$. This factorized construction follows the F-FNO design and avoids a full two-dimensional spectral kernel over all retained mode pairs.

After spectral mixing, each block applies a channel-mixing feed-forward module consisting of Group Normalization, a 1×1 expansion convolution, GELU, and a 1×1 projection convolution. The block output is formed by adding a residual branch to the processed spectral response. If the block preserves both resolution and channel width, the residual branch is the identity; if the resolution or channel width changes, the residual branch uses a padded 3×3 stride convolution, while the spectral branch is downsampled by average pooling with the same stride before addition. After the cascade of factorized spectral blocks, a lightweight channel attention module recalibrates the encoded feature channels at the final latent resolution. The resulting tensor is reshaped back to $h_{t-T+1:t} \in \mathbb{R}^{B \times T \times C_s \times n_x \times n_y}$ and passed to the projector $\mathcal{P}$.

A.2.2 Projector $\mathcal{P}$

The projector $\mathcal{P}$ is implemented as a frame-wise pointwise channel projection. It receives $h_{t-T+1:t} \in \mathbb{R}^{B \times T \times C_s \times n_x \times n_y}$ from the encoder and outputs the compact propagation state $z_{t-T+1:t} \in \mathbb{R}^{B \times T \times C_z \times n_x \times n_y}$. In implementation, the batch and time dimensions are merged before projection, so $\mathcal{P}$ operates on tensors of shape $(BT) \times C_s \times n_x \times n_y$, and the output is then reshaped back to the temporal layout.

The projector is a 1×1 convolution without normalization or nonlinear activation. At each latent grid location (i, j), it applies the same learned affine channel map, $z_j(:, i, j) = W_{\mathcal{P}} h_j(:, i, j) + b_{\mathcal{P}}$, with $W_{\mathcal{P}} \in \mathbb{R}^{C_z \times C_s}$ and $b_{\mathcal{P}} \in \mathbb{R}^{C_z}$. Since the kernel size is 1×1, $\mathcal{P}$ does not mix neighboring spatial locations and does not resample the latent grid.

We keep the latent spatial resolution unchanged in $\mathcal{P}$ because the subsequent propagator Φ performs structured evolution on spatial Fourier modes. Additional spatial downsampling at this stage would reduce the available frequency support, alter the latent phase structure, and make the retained band Λ depend on an extra resolution change immediately before propagation. Instead, the compactness of z_t is imposed along the channel axis: $\mathcal{P}$ changes the representation basis from the reconstruction-oriented encoder features to propagation-oriented compact coordinates while preserving the spatial grid on which rFFT2 is applied. When $C_z < C_s$, this layer also acts as a channel bottleneck; when $C_z = C_s$, it remains a learned change of basis between the spatial representation and the compact propagation state.

A.2.3 Structured Propagator Φ

The structured propagator Φ receives the compact state sequence $z_{t-T+1:t} \in \mathbb{R}^{B \times T \times C_z \times n_x \times n_y}$ and outputs the propagated compact sequence $\widetilde{z}_{t+1:t+T} \in \mathbb{R}^{B \times T \times C_z \times n_x \times n_y}$. It is implemented in

the spatial Fourier domain of the compact state. The propagator first applies a two-dimensional real-valued fast Fourier transform with orthonormal normalization, $\widehat{z} = \mathrm{rFFT2}(z)$, yielding $\widehat{z} \in \mathbb{C}^{B\times T\times C_z\times n_x\times(\lfloor n_y/2\rfloor+1)}$. The update is restricted to the retained spectral band $\Lambda = (m_x, m_y)$. Since rFFT2 stores only nonnegative frequencies along the last spatial axis, the implementation extracts two retained blocks: the positive height-frequency block $\widehat{z}^{+}_{\Lambda} = \widehat{z}[:,:,:,: m_x,: m_y]$ and the negative height-frequency block $\widehat{z}^{-}_{\Lambda} = \widehat{z}[:,:,:,-m_x :,: m_y]$. The reported (m_x, m_y) values are chosen so that the retained blocks fit the rFFT layout. Both blocks are evolved by the same propagation rule. Coefficients outside Λ are set to zero in the output spectrum, and inverse rFFT2 maps the updated spectrum back to the compact spatial grid.

Frequency-conditioned linear backbone K_θ. The linear backbone is parameterized by channel-wise complex temporal matrices and frequency-dependent gates. For each compact-state channel ℓ, SSP learns a complex matrix $\bar{K}_\ell \in \mathbb{C}^{T\times T}$, which maps the temporal vector of a retained Fourier mode to its propagated temporal vector. The real part of $\bar{K}_\ell$ is initialized as an identity matrix, while the imaginary part is initialized with a small perturbation. For each retained frequency $k = (k_x, k_y)$, the implementation constructs the normalized frequency feature $\xi(k) = (k_x, k_y, \|k\|, \cos\theta_k, \sin\theta_k)$, where the signed k_x distinguishes the positive and negative height-frequency blocks and k_y is nonnegative under rFFT2. A shared gate MLP a_θ, with GELU activations and a zero-initialized final layer, produces the bounded channel-wise modulation $M_\theta(k,\ell) = 1 + \beta\tanh(a_\theta(\xi(k))_\ell)$. The effective temporal operator for mode k and channel ℓ is $K_\theta^{(k,\ell)} = M_\theta(k,\ell)\bar{K}_\ell$. In implementation, this operation is applied by an Einstein contraction over the temporal dimension, while sharing $\{\bar{K}_\ell\}_{\ell=1}^{C_z}$ across all retained frequencies.

Nonlinear spectral closure G_ψ. The nonlinear closure G_ψ acts on each retained complex spectral block $\widehat{z}^{\pm}_{\Lambda} \in \mathbb{C}^{B\times T\times C_z\times m_x\times m_y}$. The implementation splits the complex coefficients into real and imaginary parts, merges the time and channel dimensions, and forms a real-valued tensor in $\mathbb{R}^{B\times(2TC_z)\times m_x\times m_y}$. This tensor is processed on the retained frequency plane by a shallow convolutional network consisting of padded 3×3 Conv2d layers with GELU activations, followed by a 1×1 Conv2d that maps back to $2TC_z$ channels. The output is split back into real and imaginary parts and reshaped to the original complex block layout. The final convolution is zero-initialized and the closure output is scaled by a small factor, so G_ψ starts as a near-zero residual correction and does not disrupt the identity-initialized linear backbone at the beginning of training.

Residual Euler propagation. Each retained block is evolved by fixed-step explicit Euler integration. Given $q^{(0)} = \widehat{z}^{\pm}_{\Lambda}$, the s-th substep computes the linear displacement $\Delta_K^{(s)} = K_\theta(q^{(s)}) - q^{(s)}$ and the nonlinear closure displacement $\Delta_G^{(s)} = G_\psi(q^{(s)})$. The retained coefficients are updated as

$$q^{(s+1)} = q^{(s)} + \Delta\tau\left(\alpha\Delta_K^{(s)} + \lambda_g\Delta_G^{(s)}\right), \qquad s = 0,\dots,N_{\mathrm{sub}}-1.$$

The same routine is applied to the positive and negative retained blocks. During integration, Φ also accumulates the orthogonality loss $\mathcal{L}_{\mathrm{orth}}$ between $\Delta_K^{(s)}$ and $\Delta_G^{(s)}$, following the definition in the main text. After the final substep, the evolved blocks $q^{(N_{\mathrm{sub}})}$ are inserted into the corresponding locations of the output spectrum, and inverse rFFT2 produces $\widetilde{z}_{t+1:t+T}$.

A.2.4 Inverse Projector $\mathcal{R}$

The inverse projector $\mathcal{R}$ maps the propagated compact sequence $\widetilde{z}_{t+1:t+T} \in \mathbb{R}^{B\times T\times C_z\times n_x\times n_y}$ back to the spatial-representation width used by the decoder. As in $\mathcal{P}$, the batch and time dimensions are merged before applying the layer, so $\mathcal{R}$ operates on tensors of shape $(BT)\times C_z\times n_x\times n_y$. The output is reshaped to $\widetilde{h}_{t+1:t+T} \in \mathbb{R}^{B\times T\times C_s\times n_x\times n_y}$ and passed to $\mathcal{D}$.

The module is implemented as a 1×1 convolution without normalization or nonlinear activation. At each latent grid location (i,j), it applies $\widetilde{h}_j(:,i,j) = W_{\mathcal{R}}\widetilde{z}_j(:,i,j) + b_{\mathcal{R}}$, where $W_{\mathcal{R}} \in \mathbb{R}^{C_s\times C_z}$ and $b_{\mathcal{R}} \in \mathbb{R}^{C_s}$. Thus, $\mathcal{R}$ preserves the latent spatial grid and only changes the channel basis. We use the term inverse projector to indicate its role as the synthesis-side lifting counterpart of $\mathcal{P}$, rather than to imply an exact algebraic inverse. In this way, $\mathcal{P}$ defines the propagation-oriented compact coordinates, while $\mathcal{R}$ restores a decoder-oriented representation for physical-field synthesis.

A.2.5 Decoder $\mathcal{D}$

The decoder $\mathcal{D}$ maps the lifted spatial representation $\widetilde{h}_{t+1:t+T} \in \mathbb{R}^{B\times T\times C_s\times n_x\times n_y}$ to the predicted physical sequence $\widetilde{u}_{t+1:t+T} \in \mathbb{R}^{B\times T\times d_u\times N_x\times N_y}$. As in the encoder, the batch and time dimensions are merged before decoding, so $\mathcal{D}$ operates frame-wise on tensors of shape $C_s \times n_x \times n_y$. The output is reshaped back to the temporal layout after decoding. The decoder first applies a lightweight channel attention module at the latent resolution, which recalibrates the lifted feature channels before spatial synthesis.

The decoder body follows the reverse of the encoder resolution schedule and is implemented as a cascade of factorized upsampling blocks. Each block contains two complementary branches. The global branch applies a factorized spectral operator: one-dimensional rFFTs are performed independently along the two spatial axes, retained modes are mixed by learned complex weights, and inverse rFFTs map the responses back to the spatial latent grid. The resulting spectral response is processed by Group Normalization, a 1×1 expansion convolution, the Sigmoid Linear Unit (SiLU) activation, and a 1×1 projection convolution, with a residual skip connection. When the block changes spatial resolution, the spectral branch and the skip branch are upsampled by bilinear interpolation before fusion. This branch provides global and mid-frequency spatial refinement during synthesis, complementing the retained spectral evolution performed by Φ.

The local branch applies a residual convolutional refinement module to the same feature map, after bilinear upsampling when the current block increases spatial resolution. This branch uses spatial convolutions to refine local structures and residual details that are not explicitly propagated as retained spectral modes. The outputs of the global spectral branch and the local convolutional branch are summed and passed through SiLU. After all upsampling and refinement blocks, a final padded 3×3 convolution maps the decoder features to the d_u physical variables. The decoder does not output coordinate channels; it produces only normalized physical fields, which are transformed back to physical units by the inverse channel-wise normalization for evaluation and visualization.

The role of $\mathcal{D}$ is therefore synthesis rather than temporal propagation. Since $\mathcal{P}$ and $\mathcal{R}$ preserve the latent spatial grid, the decoder receives a propagated representation that still contains spatial phase information. The factorized spectral branch uses this information to restore coherent global and mid-frequency structure, while the local branch refines spatially localized residual components. The convolutional encoder–decoder ablation in the main text shows that SSP remains competitive when these spectral synthesis blocks are replaced by convolutional alternatives, indicating that the primary gain is not solely due to the encoder–decoder architecture. Nevertheless, the full spectral decoder improves reconstruction fidelity, especially for spatially organized mid- and high-frequency error components.

A.3 Training Objective and Loss Implementation

SSP is trained end-to-end with autoregressive rollout supervision. For each training sample, let $u^{(0)}$ denote the normalized input window and $\{u^{(r)}\}_{r=1}^{N_{\mathrm{roll}}}$ denote the normalized future target windows. At rollout step r, the current physical window $x^{(r)}$ is concatenated with the coordinate grid γ, passed through $\mathcal{D} \circ \mathcal{R} \circ \Phi \circ \mathcal{P} \circ \mathcal{E}$, and fed back as the next input. The reconstruction, physical rollout, and latent rollout terms are implemented as mean squared error, $\mathrm{MSE}(a, b) = |\Omega_a|^{-1}\|a - b\|_F^2$, where $|\Omega_a|$ is the number of compared tensor entries. We use MSE rather than an unnormalized squared L_2 loss so that the loss scale is not directly determined by the spatial resolution, window length, or number of physical channels.

Reconstruction MSE. The reconstruction loss supervises the direct encoder–decoder path without temporal propagation. With $\widehat{u}^{(0)} = \mathrm{concat}(u^{(0)}, \gamma)$, it is defined as

$$\mathcal{L}_{\mathrm{rec}} = \mathrm{MSE}\left(\mathcal{D}(\mathcal{E}(\widehat{u}^{(0)})), u^{(0)}\right).$$

This term enforces that the shared spatial representation produced by $\mathcal{E}$ retains the information needed for physical-field synthesis. It stabilizes the analysis and synthesis modules before their representations are used by the propagation path.

Physical rollout MSE. The physical rollout loss is the main prediction objective. At rollout step r, SSP predicts $\widetilde{u}^{(r+1)}$ from the current input window and compares it with the corresponding

ground-truth future window $u^{(r+1)}$:

$$\mathcal{L}_{\text{phy}}^{(r)} = \text{MSE}\left(\widetilde{u}^{(r+1)}, u^{(r+1)}\right).$$

This loss is applied after the full analysis–propagation–synthesis pipeline and therefore directly penalizes accumulated autoregressive prediction error in the physical solution space.

Latent rollout MSE. The latent rollout loss directly supervises the compact state produced by Φ. For each target window $u^{(r+1)}$, we form its coordinate-augmented version $\widehat{u}_\star^{(r+1)} = \text{concat}(u^{(r+1)}, \gamma)$ and encode the corresponding compact target state $z_\star^{(r+1)} = \mathcal{P}(\mathcal{E}(\widehat{u}_\star^{(r+1)}))$. Let $\widetilde{z}^{(r+1)}$ be the compact state output by Φ. We use

$$\mathcal{L}_{\text{lat}}^{(r)} = \text{MSE}\left(\widetilde{z}^{(r+1)}, z_\star^{(r+1)}\right).$$

This term provides direct supervision to the structured propagator, rather than forcing all latent propagation errors to be corrected only by $\mathcal{R}$ and $\mathcal{D}$.

Orthogonality / decoupling loss. The orthogonality loss follows the definition in the main text and is applied inside Φ. Let $\Delta_K = K_\theta(\widehat{z}_\Lambda) - \widehat{z}_\Lambda$ denote the linear backbone displacement and $\Delta_G = G_\psi(\widehat{z}_\Lambda)$ denote the nonlinear closure displacement. We use

$$\ell_{\text{orth}}(\Delta_K, \Delta_G) = \frac{|\langle \Delta_K, \Delta_G \rangle|^2}{(\|\Delta_K\|_2^2 + \epsilon)(\|\Delta_G\|_2^2 + \epsilon)}.$$

For each rollout step r, the local penalty $\ell_{\text{orth}}^{(r)}$ is averaged over the two retained frequency blocks and N_{sub} Euler substeps. The training objective accumulates these penalties across rollout steps:

$$\mathcal{L}_{\text{orth}} = \sum_{r=0}^{N_{\text{roll}}-1} \ell_{\text{orth}}^{(r)}.$$

This regularizer discourages the nonlinear closure from duplicating the linear backbone update direction.

Normality loss. The normality loss regularizes the channel-wise temporal matrices in K_θ. For $\{\bar{K}_\ell\}_{\ell=1}^{C_z}$, we use

$$\mathcal{L}_{\text{norm}} = \frac{1}{C_z} \sum_{\ell=1}^{C_z} \left\| \bar{K}_\ell \bar{K}_\ell^H - \bar{K}_\ell^H \bar{K}_\ell \right\|_F^2.$$

This term encourages the learned temporal matrices to be closer to normal operators, reducing non-normal transient amplification during repeated rollout. The stability interpretation of this regularizer is discussed in Section A.6.3.

Total objective. The final training objective combines physical accuracy, latent propagation supervision, and structural regularization:

$$\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \frac{1}{N_{\text{roll}}} \sum_{r=0}^{N_{\text{roll}}-1} \left(\lambda_{\text{phy}}\mathcal{L}_{\text{phy}}^{(r)} + \lambda_{\text{lat}}\mathcal{L}_{\text{lat}}^{(r)}\right) + \lambda_{\text{orth}}\mathcal{L}_{\text{orth}} + \lambda_{\text{norm}}\mathcal{L}_{\text{norm}}. \tag{3}$$

The weights λ_{rec}, λ_{phy}, λ_{lat}, λ_{orth}, and λ_{norm} are fixed for each run and reported in the training configuration. The MSE terms supervise reconstruction, physical rollout, and compact-state propagation, whereas $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{norm}}$ constrain the internal structure of the spectral propagator.

A.4 Model Configuration and Parameter Description

This section summarizes the model and training configurations used in the experiments. Table 5 defines the symbols used in the configuration tables. Tables 6, 7, and 8 report the dataset-specific architecture, propagation, and training settings, respectively.

Table 5: Parameter glossary for SSP.

Parameter	Module	Description
d_u	Data	Number of physical solution variables in u_t.
T	Data / Φ	Input window length; also the temporal dimension of the compact state sequence.
$N_x \times N_y$	Data	Spatial resolution of physical input and output fields.
$\mathbf{s}_{\mathcal{E}}$	$\mathcal{E}, \mathcal{D}$	Encoder stride schedule; the decoder uses the reversed schedule for upsampling.
$n_x \times n_y$	$\mathcal{E}, \Phi, \mathcal{D}$	Latent spatial resolution after $\mathcal{E}$ and before $\mathcal{D}$.
C_s	$\mathcal{E}, \mathcal{R}, \mathcal{D}$	Channel width of the shared spatial representation.
C_z	$\mathcal{P}, \Phi, \mathcal{R}$	Channel width of the compact propagation state.
$L_{\mathcal{E}}$	$\mathcal{E}$	Number of factorized spectral encoder blocks, excluding the initial convolutional lift.
$L_{\mathcal{D}}$	$\mathcal{D}$	Number of factorized upsampling/refinement decoder blocks.
$m_{\mathcal{E}}$	$\mathcal{E}$	Per-axis retained Fourier modes in the factorized spectral encoder blocks.
$m_{\mathcal{D}}$	$\mathcal{D}$	Per-axis retained Fourier modes in the factorized spectral decoder blocks.
(m_x, m_y)	Φ	Retained spectral band size used by structured propagation in Λ.
N_{sub}	Φ	Number of explicit Euler substeps used in residual spectral propagation.
$\Delta\tau$	Φ	Internal latent-time step size for each Euler substep.
α	Φ	Weight of the linear displacement $K_\theta(\widehat{z}_\Lambda) - \widehat{z}_\Lambda$.
λ_g	Φ	Weight of the nonlinear closure displacement $G_\psi(\widehat{z}_\Lambda)$.
β	K_θ	Gate scale in $M_\theta(k,\ell) = 1 + \beta \tanh(a_\theta(\xi(k))_\ell)$.
C_a	K_θ	Hidden width of the frequency-gate MLP a_θ.
C_ψ	G_ψ	Hidden channel width of the convolutional closure network.
L_ψ	G_ψ	Number of 3×3 Conv2d–GELU blocks in G_ψ, excluding the final 1×1 output convolution.
s_ψ	G_ψ	Output scale applied to the nonlinear closure.
N_{roll}	Training	Number of autoregressive rollout windows used during training.
λ_{rec}	Loss	Weight of the reconstruction MSE loss.
λ_{phy}	Loss	Weight of the physical rollout MSE loss.
λ_{lat}	Loss	Weight of the compact-state rollout MSE loss.
λ_{orth}	Loss	Weight of the backbone–closure decoupling penalty.
λ_{norm}	Loss	Weight of the temporal-matrix normality penalty.

Table 6: Dataset-specific architecture configuration of SSP.

Dataset	d_u	T	$N_x \times N_y$	$\mathbf{s}_{\mathcal{E}}$	$n_x \times n_y$	C_s	C_z	$L_{\mathcal{E}}$	$L_{\mathcal{D}}$
Navier–Stokes	1	4	64×64	$(1,1,1,1)$	64×64	64	32	3	5
Shallow–Water	1	10	128×128	$(2,1,1,1)$	64×64	32	16	3	5
Reaction–Diffusion	2	10	128×128	$(2,2,1,1)$	32×32	64	32	3	5

A.5 Pseudocode

Algorithms 1 and 2 summarize the autoregressive training procedure and the structured spectral propagation routine. During inference, the same rollout loop is used without target encoding, loss accumulation, or parameter updates.

Algorithm 1 Autoregressive training of SSP

Input: normalized input window $u^{(0)}$, target rollout windows $\{u^{(r)}\}_{r=1}^{N_{\mathrm{roll}}}$, coordinate grid $\gamma = (\gamma_x, \gamma_y)$
Parameters: $\Theta = \{\mathcal{E}, \mathcal{P}, \Phi, \mathcal{R}, \mathcal{D}, K_\theta, G_\psi\}$
Hyperparameters: N_{roll}, λ_{rec}, λ_{phy}, λ_{lat}, λ_{orth}, λ_{norm}
Output: training objective $\mathcal{L}$ and updated parameters Θ

$\widehat{u}^{(0)} \leftarrow \mathrm{concat}(u^{(0)}, \gamma)$.
$h_{\mathrm{rec}} \leftarrow \mathcal{E}(\widehat{u}^{(0)})$; $\widetilde{u}_{\mathrm{rec}} \leftarrow \mathcal{D}(h_{\mathrm{rec}})$.
$\mathcal{L}_{\mathrm{rec}} \leftarrow \mathrm{MSE}(\widetilde{u}_{\mathrm{rec}}, u^{(0)})$.
Initialize $\mathcal{L}_{\mathrm{phy}} \leftarrow 0$, $\mathcal{L}_{\mathrm{lat}} \leftarrow 0$, and $\mathcal{L}_{\mathrm{orth}} \leftarrow 0$.
Set rollout input $x^{(0)} \leftarrow u^{(0)}$.
for $r = 0, \ldots, N_{\mathrm{roll}} - 1$ **do**

Table 7: Dataset-specific structured-propagation configuration of SSP.

Dataset	$m_{\mathcal{E}}$	$m_{\mathcal{D}}$	(m_x, m_y)	N_{sub}	$\Delta\tau$	α	λ_g	β	C_a	C_ψ	L_ψ	s_ψ
Navier–Stokes	12	12	$(16, 16)$	4	0.25	1	1	0.1	64	128	2	0.01
Shallow–Water	12	12	$(24, 24)$	4	0.25	1	1	0.1	64	128	2	0.01
Reaction–Diffusion	12	12	$(16, 16)$	4	0.25	1	1	0.1	64	128	2	0.01

Table 8: Dataset-specific training configuration of SSP.

Dataset	N_{roll}	Batch size	Epochs	Optimizer	LR	Scheduler	$(\lambda_{\mathrm{rec}}, \lambda_{\mathrm{phy}}, \lambda_{\mathrm{lat}}, \lambda_{\mathrm{orth}}, \lambda_{\mathrm{norm}})$
Navier–Stokes	3	10	250	AdamW	10^{-3}	Cosine warm restart	$(1, 1, 1, 0.01, 0.01)$
Shallow–Water	9	2	250	AdamW	10^{-3}	Cosine warm restart	$(1, 1, 1, 0.01, 0.01)$
Reaction–Diffusion	9	2	250	AdamW	10^{-3}	Cosine warm restart	$(1, 1, 1, 0.01, 0.01)$

$\widehat{x}^{(r)} \leftarrow \mathrm{concat}(x^{(r)}, \gamma)$.
$h^{(r)} \leftarrow \mathcal{E}(\widehat{x}^{(r)})$; $z^{(r)} \leftarrow \mathcal{P}(h^{(r)})$.
$(\widetilde{z}^{(r+1)}, \ell_{\mathrm{orth}}^{(r)}) \leftarrow \Phi(z^{(r)}; K_\theta, G_\psi, \Lambda)$.
$\widetilde{h}^{(r+1)} \leftarrow \mathcal{R}(\widetilde{z}^{(r+1)})$; $\widetilde{u}^{(r+1)} \leftarrow \mathcal{D}(\widetilde{h}^{(r+1)})$.
$\mathcal{L}_{\mathrm{phy}} \leftarrow \mathcal{L}_{\mathrm{phy}} + \mathrm{MSE}(\widetilde{u}^{(r+1)}, u^{(r+1)})$.
$\widehat{u}_\star^{(r+1)} \leftarrow \mathrm{concat}(u^{(r+1)}, \gamma)$.
$z_\star^{(r+1)} \leftarrow \mathcal{P}(\mathcal{E}(\widehat{u}_\star^{(r+1)}))$.
$\mathcal{L}_{\mathrm{lat}} \leftarrow \mathcal{L}_{\mathrm{lat}} + \mathrm{MSE}(\widetilde{z}^{(r+1)}, z_\star^{(r+1)})$.
$\mathcal{L}_{\mathrm{orth}} \leftarrow \mathcal{L}_{\mathrm{orth}} + \ell_{\mathrm{orth}}^{(r)}$.
Feed back prediction: $x^{(r+1)} \leftarrow \widetilde{u}^{(r+1)}$.
end for
Compute $\mathcal{L}_{\mathrm{norm}}$ from $\{\bar{K}_\ell\}_{\ell=1}^{C_z}$ as defined in Section A.3.
Form the total objective

$$\mathcal{L} = \lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} + \frac{\lambda_{\mathrm{phy}}\mathcal{L}_{\mathrm{phy}} + \lambda_{\mathrm{lat}}\mathcal{L}_{\mathrm{lat}}}{N_{\mathrm{roll}}} + \lambda_{\mathrm{orth}}\mathcal{L}_{\mathrm{orth}} + \lambda_{\mathrm{norm}}\mathcal{L}_{\mathrm{norm}}.$$

Update Θ by backpropagation with the optimizer.
return $\mathcal{L}$ and updated Θ.

Algorithm 2 Structured spectral propagation in Φ

Input: compact state sequence $z \in \mathbb{R}^{B\times T\times C_z\times n_x\times n_y}$
Parameters: temporal matrices $\{\bar{K}_\ell\}_{\ell=1}^{C_z}$, gate MLP a_θ, closure G_ψ
Hyperparameters: retained band $\Lambda = (m_x, m_y)$, N_{sub}, $\Delta\tau$, α, λ_g, β
Output: propagated compact sequence $\widetilde{z}$ and local decoupling loss ℓ_{orth}

$\widehat{z} \leftarrow \mathrm{rFFT2}(z)$.
Initialize $\widehat{z}_{\mathrm{out}} \leftarrow 0$ with the same spectral shape as $\widehat{z}$; initialize $\ell_{\mathrm{orth}} \leftarrow 0$.
for each retained block $b \in \{+, -\}$ **do**
  Extract retained block: $\widehat{z}_\Lambda^b \leftarrow \Pi_\Lambda^b(\widehat{z})$.
  Build signed frequency features $\xi^b(k) = (k_x, k_y, \|k\|, \cos\theta_k, \sin\theta_k)$.
  Compute gates $M_\theta^b(k, \ell) = 1 + \beta\tanh(a_\theta(\xi^b(k))_\ell)$.
  Initialize $q^{(0)} \leftarrow \widehat{z}_\Lambda^b$.
  for $s = 0, \ldots, N_{\mathrm{sub}} - 1$ **do**
    Apply the gated temporal backbone $K_\theta^b(q^{(s)})$ using $K_\theta^{b,(k,\ell)} = M_\theta^b(k, \ell)\bar{K}_\ell$.
    $\Delta_K^{(s)} \leftarrow K_\theta^b(q^{(s)}) - q^{(s)}$.
    $\Delta_G^{(s)} \leftarrow G_\psi(q^{(s)})$.

Accumulate

$$\ell_{\mathrm{orth}} \leftarrow \ell_{\mathrm{orth}} + \frac{|\langle \Delta_K^{(s)}, \Delta_G^{(s)} \rangle|^2}{(\|\Delta_K^{(s)}\|_2^2 + \epsilon)(\|\Delta_G^{(s)}\|_2^2 + \epsilon)}.$$

Update $q^{(s+1)} \leftarrow q^{(s)} + \Delta\tau(\alpha\Delta_K^{(s)} + \lambda_g \Delta_G^{(s)})$.

end for

Insert $q^{(N_{\mathrm{sub}})}$ into the corresponding retained block of $\widehat{z}_{\mathrm{out}}$.

end for

Average ℓ_{orth} over retained blocks and Euler substeps.

$\widetilde{z} \leftarrow \mathrm{irFFT2}(\widehat{z}_{\mathrm{out}})$.

return $\widetilde{z}, \ell_{\mathrm{orth}}$.

A.6 Propagation Design and Stability Analysis

A.6.1 Fourier-domain Example: Linear Wave Equation

A simple linear PDE illustrates why modeling temporal evolution in the spectral domain is natural. Consider the linear wave equation on a periodic spatial domain,

$$\partial_{tt} u(x,t) = c^2 \Delta u(x,t),$$

where c is the wave speed. Applying the spatial Fourier transform gives, for each Fourier mode k,

$$\partial_{tt} \widehat{u}(k,t) = -c^2 \|k\|^2 \widehat{u}(k,t).$$

Thus, the original spatio-temporal PDE is converted into independent temporal systems indexed by frequency. Introducing $\widehat{v}(k,t) = \partial_t \widehat{u}(k,t)$, each mode satisfies the first order linear ODE

$$\frac{d}{dt}\begin{bmatrix}\widehat{u}(k,t)\\ \widehat{v}(k,t)\end{bmatrix} = A(k)\begin{bmatrix}\widehat{u}(k,t)\\ \widehat{v}(k,t)\end{bmatrix}, \qquad A(k) = \begin{bmatrix}0 & 1\\ -c^2\|k\|^2 & 0\end{bmatrix}.$$

The corresponding finite-time evolution is

$$\begin{bmatrix}\widehat{u}(k,t+\Delta t)\\ \widehat{v}(k,t+\Delta t)\end{bmatrix} = \exp(\Delta t A(k))\begin{bmatrix}\widehat{u}(k,t)\\ \widehat{v}(k,t)\end{bmatrix}.$$

This example shows that a spatial Fourier mode can have its own temporal evolution rule and that the frequency magnitude $\|k\|$ directly controls the temporal oscillation. Therefore, a frequency-conditioned temporal operator is a natural structure for spectral propagation.

SSP uses this principle in a learned latent representation rather than in the raw physical variables. The propagator Φ first restricts the compact state to the retained spectral band Λ, then applies a frequency-conditioned temporal backbone K_θ to the temporal vector of each retained mode. For nonlinear PDEs, the Fourier modes are generally no longer closed under independent linear evolution: nonlinear products in physical space induce mode interactions in Fourier space. This motivates the additional nonlinear closure G_ψ, which supplements the structured linear modal evolution with residual corrections for nonlinear mode coupling and finite-band truncation effects. The next subsection formalizes this spectral propagation as a latent-time ODE discretization.

A.6.2 Latent-time ODE Modeling in Retained Spectral Space

The structured propagator Φ can be viewed as learning an internal continuous dynamics in the retained spectral latent space. After the analysis maps $\mathcal{E}$ and $\mathcal{P}$, the compact states provide a finite-dimensional coordinate representation of the solution trajectory. We therefore model the propagation of the retained spectral coefficients $q(\tau) := \widehat{z}_\Lambda(\tau)$ through an auxiliary latent-time variable τ. This latent time is used only inside the propagator and should not be identified with the physical simulation time t.

In continuous form, the retained spectral state is assumed to evolve according to a structured latent vector field

$$\frac{dq}{d\tau} = f_\Theta(q) = \mathcal{A}_\theta q + \lambda_g G_\psi(q),$$

Table 9: Effect of normality regularization on Shallow–Water long-horizon rollout. We compare SSP trained with and without $\mathcal{L}_{\mathrm{norm}}$. All metrics are errors; lower is better. Relative improvement is computed with respect to the variant without $\mathcal{L}_{\mathrm{norm}}$.

Metric	w/o $\mathcal{L}_{\mathrm{norm}}$	w/ $\mathcal{L}_{\mathrm{norm}}$	Improvement
L_2	6.82E−4	5.97E−4	12.40%
$E_{\max}$	1.04E−2	8.22E−3	20.63%
f_{low}	4.27E−5	2.30E−5	46.02%
f_{mid}	4.63E−5	3.45E−5	25.51%
f_{high}	3.92E−5	3.49E−5	11.15%
f_{mse}	4.03E−5	3.41E−5	15.38%

Table 10: Per-batch consistency of the gains from normality regularization on Shallow–Water long-horizon rollout. We report the number of test batches improved by $\mathcal{L}_{\mathrm{norm}}$ and the average relative improvement across batches.

Metric	Improved batches	Mean batch-wise improvement
L_2	50/50	12.45%
$E_{\max}$	50/50	20.51%
f_{low}	50/50	45.09%
f_{mid}	50/50	25.56%
f_{high}	50/50	11.16%
f_{mse}	50/50	15.53%

where $\mathcal{A}_\theta$ denotes the linear modal generator and G_ψ denotes the nonlinear spectral closure. This is analogous to a Neural ODE on the retained latent spectrum, but with a structured decomposition of the vector field: the dominant modal evolution is represented by a linear operator, while the nonlinear closure accounts for residual mode interactions and finite-band truncation effects.

Instead of parameterizing the infinitesimal generator $\mathcal{A}_\theta$ directly, SSP learns a finite-window temporal map K_θ acting on the temporal vector of each retained Fourier mode. This naturally leads to the residual form used in the implementation. Let $\mathcal{F}_\eta$ be the flow map of the linear subsystem $\frac{dq}{d\tau} = \mathcal{A}_\theta q$ over an internal time interval η. Its generator satisfies

$$\mathcal{A}_\theta q = \frac{d}{d\eta}\mathcal{F}_\eta(q)\Big|_{\eta=0} = \lim_{\eta\to 0}\frac{\mathcal{F}_\eta(q) - q}{\eta}.$$

When the learned temporal map K_θ is used as a finite-step approximation of $\mathcal{F}_\eta$, the generator can be approximated by the forward difference

$$\mathcal{A}_\theta q \approx \alpha\left(K_\theta(q) - q\right),$$

where α absorbs the effective inverse latent-time interval and the scale of the learned linear displacement. Substituting this approximation into the latent ODE gives

$$\frac{dq}{d\tau} = \alpha\left(K_\theta(q) - q\right) + \lambda_g G_\psi(q).$$

Thus, the term $K_\theta(q) - q$ is not an additional prediction target; it is the finite-flow displacement used to approximate the latent linear generator.

The implementation discretizes this latent ODE by fixed-step explicit Euler integration. Starting from $q^{(0)} = \widehat{z}_\Lambda$, the retained coefficients are updated by

$$q^{(s+1)} = q^{(s)} + \Delta\tau\left[\alpha\left(K_\theta(q^{(s)}) - q^{(s)}\right) + \lambda_g G_\psi(q^{(s)})\right], \qquad s = 0, \ldots, N_{\mathrm{sub}} - 1.$$

Here $\Delta\tau$ is the internal step size in latent space. It controls the numerical composition of the learned vector field inside Φ, rather than the physical temporal spacing of the PDE trajectory.

This residual parameterization is useful for autoregressive rollout. If $K_\theta \approx I$ and $G_\psi \approx 0$ at initialization, then $f_\Theta(q) \approx 0$, and the initial propagator is close to the identity flow in latent space. Ignoring the closure term for clarity, one Euler substep reduces to

$$q^{(s+1)} = \left[(1 - \alpha\Delta\tau)I + \alpha\Delta\tau K_\theta\right] q^{(s)}.$$

The state is therefore moved along a controlled latent displacement instead of being directly replaced by $K_\theta q^{(s)}$. This relaxed update is better suited to repeated composition, where abrupt finite-step transitions may amplify small prediction errors over long rollouts.

A.6.3 Normality Regularization and Transient Growth

The normality regularizer is applied to the channel-wise temporal matrices $\{\bar{K}_\ell\}_{\ell=1}^{C_z}$ in the linear backbone. Its purpose is not to directly constrain the eigenvalues of $\bar{K}_\ell$, but to reduce the non-normality of the learned temporal maps. For a complex matrix A, normality means $AA^H = A^H A$. A normal matrix is unitarily diagonalizable, i.e., there exist a unitary matrix U and a diagonal matrix Λ such that $A = U\Lambda U^H$. Consequently, repeated applications of A satisfy

$$\|A^n\|_2 = \|\Lambda^n\|_2 = \max_i |\lambda_i(A)|^n.$$

Thus, for normal matrices, growth under repeated composition is governed by the magnitudes of the eigenvalues.

This property is relevant to the frequency-conditioned Euler update used by Φ. For mode k and channel ℓ, ignoring the nonlinear closure, the effective linear substep matrix is

$$S_{k,\ell} = (1 - \alpha\Delta\tau)I + \alpha\Delta\tau M_\theta(k,\ell)\bar{K}_\ell.$$

Since $M_\theta(k,\ell)$ is scalar-valued, normality of $\bar{K}_\ell$ also implies normality of $S_{k,\ell}$. For matrix-level diagnostics, we additionally consider the ungated case $M_\theta(k,\ell) = 1$. The corresponding substep matrix is

$$S_\ell = I + \alpha\Delta\tau(\bar{K}_\ell - I) = (1 - \alpha\Delta\tau)I + \alpha\Delta\tau\bar{K}_\ell.$$

When $\bar{K}_\ell$ is normal, S_ℓ is also normal, since S_ℓ is an affine polynomial of $\bar{K}_\ell$. The eigenvalues of S_ℓ are therefore given by

$$\lambda_i(S_\ell) = 1 + \alpha\Delta\tau(\lambda_i(\bar{K}_\ell) - 1).$$

Consequently,

$$\|S_\ell^n\|_2 = \max_i \left|1 + \alpha\Delta\tau(\lambda_i(\bar{K}_\ell) - 1)\right|^n.$$

In practice, Φ applies N_{sub} residual Euler substeps within one latent propagation block. We therefore define the effective linear propagation matrix

$$A_{\mathrm{eff},\ell} = S_\ell^{N_{\mathrm{sub}}}.$$

This matrix summarizes the composed linear backbone for channel ℓ over one propagation block. It excludes the nonlinear spectral closure and is used only as a diagnostic object, not as the full nonlinear rollout operator of SSP. For normal $\bar{K}_\ell$, both S_ℓ and $A_{\mathrm{eff},\ell}$ remain normal, so the amplification of the linear part is determined by the mapped eigenvalues of $\bar{K}_\ell$, rather than by non-orthogonal eigenvectors.

For non-normal matrices, eigenvalues alone may not characterize short-term growth. Even when the eigenvalues are moderate, non-orthogonal eigenvectors can produce transient amplification under repeated multiplication. This effect is undesirable in autoregressive rollout because small latent errors may be temporarily amplified before the eigenvalue asymptotics dominate. We therefore penalize the departure from normality by

$$\mathcal{L}_{\mathrm{norm}} = \frac{1}{C_z}\sum_{\ell=1}^{C_z} \left\|\bar{K}_\ell\bar{K}_\ell^H - \bar{K}_\ell^H\bar{K}_\ell\right\|_F^2.$$

This loss encourages the temporal matrices to have more orthogonal spectral structure and reduces a source of non-normal transient growth.

The regularizer should be interpreted as a structural bias rather than a complete stability guarantee. It does not by itself bound the spectral radius of $\bar{K}_\ell$, nor does it ensure that the Euler update S_ℓ is contractive. In SSP, rollout stability is supported jointly by identity initialization of $\bar{K}_\ell$, bounded frequency gates, finite-band spectral propagation, residual Euler updates, and the normality penalty.

To isolate the effect of the normality regularizer, we compare SSP trained with and without $\mathcal{L}_{\mathrm{norm}}$ on the Shallow–Water long-horizon rollout task. Tables 9 and 10 show that $\mathcal{L}_{\mathrm{norm}}$ improves the main

Table 11: Matrix-level diagnostics for the normality regularizer on Shallow–Water. We compare SSP trained with and without $\mathcal{L}_{\text{norm}}$. Statistics are computed over channel-wise temporal matrices. Lower η_{norm} and η_{schur} indicate weaker non-normality and weaker Schur off-diagonal coupling.

Operator	Setting	η_{norm} mean	η_{schur} mean	ρ mean	ρ max
$\bar{K}_\ell$	w/o $\mathcal{L}_{\text{norm}}$	5.58E−2	1.77E−1	1.32	1.77
	w/ $\mathcal{L}_{\text{norm}}$	**5.00E−5**	**2.63E−3**	**1.15**	**1.33**
S_ℓ	w/o $\mathcal{L}_{\text{norm}}$	3.66E−3	4.51E−2	1.07	1.19
	w/ $\mathcal{L}_{\text{norm}}$	**3.00E−6**	**6.52E−4**	**1.04**	**1.08**
A_{eff}	w/o $\mathcal{L}_{\text{norm}}$	6.17E−2	1.81E−1	1.34	2.02
	w/ $\mathcal{L}_{\text{norm}}$	**5.00E−5**	**2.61E−3**	**1.16**	**1.37**

trajectory and spectral metrics, especially f_{low}, and f_{mid}. The gains are consistent across test batches: L_2, $E_{\max}$, f_{low}, f_{mid}, f_{high}, and f_{mse} improve on all 50 batches.

Matrix-level diagnostics in Table 11 empirically verify that the regularizer reduces non-normality in the learned temporal matrices. Here, η_{norm} denotes the normalized normality residual, and η_{schur} denotes the normalized Schur off-diagonal energy. For $\bar{K}_\ell$, η_{norm} drops from $5.58\text{E}{-2}$ to $5.00\text{E}{-5}$, and η_{schur} drops from $1.77\text{E}{-1}$ to $2.63\text{E}{-3}$. Similar reductions are observed for the effective matrices $A_{\text{eff},\ell}$, whose maximum spectral radius decreases from 2.02 to 1.37. Consistently, Figure 5 shows that $\mathcal{L}_{\text{norm}}$ yields a more compact eigenvalue distribution for $A_{\text{eff},\ell}$. These diagnostics support the role of $\mathcal{L}_{\text{norm}}$ as a structural bias that reduces non-normal coupling and potential transient amplification, rather than as a standalone stability guarantee.

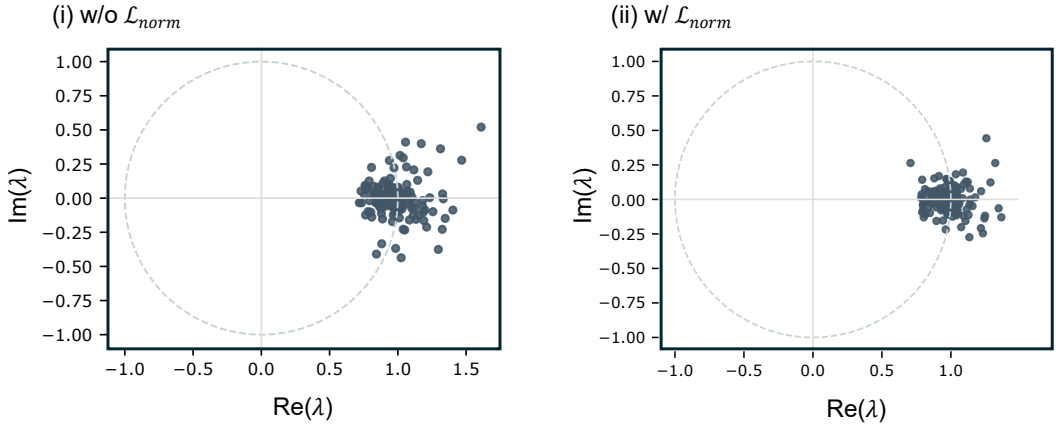


Figure 5: Eigenvalue distributions of the ungated effective linear propagation matrices A_{eff} on Shallow–Water. The dashed circle denotes the unit circle $|\lambda| = 1$. The results characterize the shared temporal backbone without frequency modulation or nonlinear closure.

Table 12: Relative reduction induced by $\mathcal{L}_{\text{norm}}$ in matrix-level diagnostics. Reductions are computed against the model trained without $\mathcal{L}_{\text{norm}}$.

Operator	η_{norm} reduction	η_{schur} reduction	$\rho_{\max}$ reduction
$\bar{K}_\ell$	99.91%	98.51%	24.78%
S_ℓ	99.92%	98.56%	9.23%
A_{eff}	99.92%	98.56%	32.10%

B Dataset details

2D Incompressible Navier–Stokes Dataset. The Navier–Stokes equations are widely used to describe the dynamics of viscous incompressible fluids. Following the data generation setting of

FNO, we use the two-dimensional incompressible Navier–Stokes dataset released with the FNO PDE benchmark[2]. The equation is considered in the vorticity form on the unit torus with periodic boundary conditions:

$$\begin{aligned} \partial_t \omega(t,x) + \mathbf{u}(t,x) \cdot \nabla \omega(t,x) &= \nu \Delta \omega(t,x) + f(x), && x \in (0,1)^2,\ t \in (0,T], \\ \nabla \cdot \mathbf{u}(t,x) &= 0, && x \in (0,1)^2,\ t \in [0,T], \\ \omega(0,x) &= \omega_0(x), && x \in (0,1)^2. \end{aligned} \tag{4}$$

Here, $\omega = \nabla \times \mathbf{u}$ denotes the vorticity, $\mathbf{u}$ is the velocity field, ν is the viscosity coefficient, ω_0 is the initial vorticity, and f is the forcing term. The velocity field is recovered from the stream function ψ by

$$-\Delta \psi = \omega, \qquad \mathbf{u} = \left(\frac{\partial \psi}{\partial y}, -\frac{\partial \psi}{\partial x} \right). \tag{5}$$

The initial vorticity fields are sampled from a Gaussian random field, and the forcing term is defined as

$$f(x,y) = 0.1 \left[\sin\left(2\pi(x+y)\right) + \cos\left(2\pi(x+y)\right) \right]. \tag{6}$$

The resulting dynamics contain nonlinear advection, viscous diffusion, and external forcing, making the dataset suitable for evaluating neural operator models on fluid dynamics prediction.

The generated dataset contains 1200 trajectories, and each trajectory consists of 20 temporal snapshots. Each snapshot is a single-channel vorticity field on a 64×64 spatial grid. Therefore, the dataset can be represented as

$$\mathcal{D} = \left\{ \omega_i^{0:19} \right\}_{i=1}^{1200}, \qquad \omega_i^t \in \mathbb{R}^{1 \times 64 \times 64}. \tag{7}$$

The full tensor shape is $[N, T, C, H, W] = [1200, 20, 1, 64, 64]$.

2D Shallow–Water Dataset. The shallow water equations are commonly used to describe free-surface flows whose horizontal length scale is much larger than the vertical water depth. In this work, we use a processed shallow water dataset generated from PDEBench[3]. The data correspond to a two-dimensional radial dam-break scenario, where the initial water height forms a circular high-water region in the center of the domain. A standard form of the two-dimensional shallow water equations is given by

$$\begin{aligned} \partial_t h + \partial_x (hu) + \partial_y (hv) &= 0, \\ \partial_t (hu) + \partial_x \left(hu^2 + \frac{1}{2} g h^2 \right) + \partial_y (huv) &= -gh \partial_x b, \\ \partial_t (hv) + \partial_x (huv) + \partial_y \left(hv^2 + \frac{1}{2} g h^2 \right) &= -gh \partial_y b, \end{aligned} \tag{8}$$

where h denotes the water height, u and v are the horizontal velocity components, g is the gravitational acceleration, and b denotes the bathymetry. In the radial dam-break setting, the initial condition is controlled by a dam radius sampled from

$$r \sim \mathcal{U}(0.3, 0.7), \tag{9}$$

which introduces variations across different trajectories. This dataset contains wave propagation, nonlinear transport, and dam-break dynamics, making it suitable for evaluating the spatio-temporal prediction ability of neural operator models.

The processed dataset used in this work contains 1000 trajectories, and each trajectory consists of 101 temporal snapshots. The current exported version is a single-variable dataset, where only the water depth h is retained at each time step. Each snapshot is stored as a `float32` array with spatial resolution 128×128. Therefore, the dataset can be represented as

$$\mathcal{D} = \left\{ q_i^{0:100} \right\}_{i=1}^{1000}, \qquad q_i^t \in \mathbb{R}^{1 \times 128 \times 128}. \tag{10}$$

The full tensor shape is $[N, T, C, H, W] = [1000, 101, 1, 128, 128]$.

[2] `https://drive.google.com/drive/folders/1UnbQh2WWc6knEHbLn-ZaXrKUZhp7pjt-?usp=sharing`

[3] `https://github.com/pdebench/PDEBench`

2D Reaction–Diffusion Dataset. The reaction–diffusion equation describes the spatio-temporal evolution of interacting variables under both local reaction and spatial diffusion effects. In this work, we use a processed two-dimensional reaction–diffusion dataset generated from PDEBench[4]. The system contains two nonlinearly coupled variables, namely the activator $u = u(t, x, y)$ and the inhibitor $v = v(t, x, y)$. The governing equations are written as

$$\begin{aligned} \partial_t u &= D_u \left(\partial_{xx} u + \partial_{yy} u\right) + R_u(u, v), \\ \partial_t v &= D_v \left(\partial_{xx} v + \partial_{yy} v\right) + R_v(u, v), \end{aligned} \tag{11}$$

where D_u and D_v are the diffusion coefficients of the activator and inhibitor, respectively. The reaction terms follow the FitzHugh–Nagumo form:

$$\begin{aligned} R_u(u, v) &= u - u^3 - k - v, \\ R_v(u, v) &= u - v, \end{aligned} \tag{12}$$

with $k = 5 \times 10^{-3}$, $D_u = 1 \times 10^{-3}$, and $D_v = 5 \times 10^{-3}$. The simulation is defined on the spatial domain $(x, y) \in (-1, 1)^2$ with no-flow Neumann boundary conditions. This dataset contains nonlinear coupling between two variables and diffusion-driven pattern formation dynamics, making it suitable for evaluating neural operator models on multi-channel spatio-temporal prediction.

The processed dataset used in this work contains 1000 trajectories, and each trajectory consists of 101 temporal snapshots. Each snapshot contains two variables and is stored as a `float32` array with shape $(2, 128, 128)$. Therefore, the dataset can be represented as

$$\mathcal{D} = \left\{ \mathbf{q}_i^{0:100} \right\}_{i=1}^{1000}, \qquad \mathbf{q}_i^t \in \mathbb{R}^{2 \times 128 \times 128}, \tag{13}$$

where the two channels correspond to the activator and inhibitor variables. The full tensor shape is $[N, T, C, H, W] = [1000, 101, 2, 128, 128]$

Numerical solvers and temporal sampling. We retain the original FNO and PDEBench data-generation implementations. Below, δt denotes an internal integration step and Δt_{save} denotes the interval between saved frames.

NS. We use the FNO vorticity–streamfunction pseudo-spectral solver with $2/3$ de-aliasing, Crank–Nicolson treatment of viscous diffusion, and explicit treatment of nonlinear advection, at $\nu = 10^{-5}$. The simulation grid is 256×256, spatially subsampled to 64×64 for learning and evaluation. The internal step is $\delta t = 10^{-4}$ and $\Delta t_{\text{save}} = 1$. Initial vorticity is sampled using the original FNO Gaussian random-field generator with $\alpha = 2.5$ and $\tau = 7$, including its original normalization and zero-mean convention. [5]

SW. The baseline radial dam break uses a flat bed on $[-2.5, 2.5]^2$, $g = 1$, $h_{\text{in}} = 2$, $h_{\text{out}} = 1$, and zero initial momenta. The dam is centered at $(0, 0)$ with $r \sim \mathcal{U}(0.3, 0.7)$. We retain PDEBench's PyClaw finite-volume solver (`ClawSolver2D`), its Roe Riemann solver with an entropy fix, the monotonized-central (MC) limiter, extrapolation boundary conditions, and the original CFL-based time-step control. At 128×128 resolution, $\Delta x = \Delta y = 5/128$ and $\Delta t_{\text{save}} = 0.01$. [6]

RD. We retain PDEBench's cell-centered finite-volume discretization with a five-point Laplacian and no-flow Neumann boundaries. At 128×128 resolution on $[-1, 1]^2$, $\Delta x = \Delta y = 2/128$. Initial activator and inhibitor values are sampled independently from standard normal distributions at the grid cells using the original sampler. Time integration uses SciPy's adaptive RK45 method through `solve_ivp`, with the unmodified defaults $\text{rtol} = 10^{-3}$ and $\text{atol} = 10^{-6}$; $\Delta t_{\text{save}} = 0.05$. The SW and RD saved-frame intervals are output sampling intervals, not fixed internal integration steps. [7]

[4] `https://github.com/pdebench/PDEBench`

[5] FNO solver and initial-condition generator: `https://github.com/scaomath/fourier_neural_operator/tree/master/data_generation/navier_stokes`.

[6] PDEBench SW solver and configuration: `https://github.com/pdebench/PDEBench/blob/main/pdebench/data_gen/src/sim_radial_dam_break.py` and `https://github.com/pdebench/PDEBench/blob/main/pdebench/data_gen/configs/radial_dam_break.yaml`.

[7] PDEBench RD solver and configuration: `https://github.com/pdebench/PDEBench/blob/main/pdebench/data_gen/src/sim_diff_react.py` and `https://github.com/pdebench/PDEBench/blob/main/pdebench/data_gen/configs/diff-react.yaml`.

Table 13: Common experimental settings for all compared methods. The listed input lengths apply to methods other than F-FNO, which uses only the final frame of the corresponding history window. Forecast origins and target frames are identical across methods.

Setting	SW / SW-Ext.	RD / RD-Ext.	NS
Input Length T_{in}	10	10	4
Supervised Prediction Length	90 / 50	90 / 50	12
Test Rollout Length	90	90	12
Evaluation Range	steps 1–90 / 51–90	steps 1–90 / 51–90	steps 1–12
Spatial Resolution	128×128	128×128	64×64
Number of Channels	1	2	1
Number of Trajectories	1000	1000	1200
Frames per Trajectory	101	101	20
Train / Val-Test Split	$900/100$	$900/100$	$1000/200$
Single-Frame Shape	$(128, 128)$	$(2, 128, 128)$	$(64, 64)$

External Assets and Licenses All datasets used in this work are publicly available scientific simulation datasets. We do not use private, scraped, human-subject, or personally identifiable data. The Shallow–Water and Reaction–Diffusion datasets are processed from PDEBench, whose dataset collection is hosted on DaRUS under the Creative Commons Attribution 4.0 International (CC BY 4.0) license.

The Navier–Stokes dataset is obtained from the public Fourier Neural Operator (FNO) repository[8], which is released under the MIT License. The repository provides the Burgers, Darcy flow, and Navier–Stokes benchmark datasets used in the original FNO paper, together with the corresponding data-generation scripts. In this work, we use the provided Navier–Stokes benchmark data only for scientific evaluation, and cite the original FNO paper and repository as the data source. We do not redistribute the raw third-party datasets as part of this submission.

C Experiment details

C.1 Experimental Setup

Benchmark settings. Table 13 summarizes the common data and evaluation settings used for all compared methods. For SW and RD, methods other than F-FNO use 10 history frames and predict 90 future steps. For the original NS benchmark, they use 4 history frames and predict 12 future steps. F-FNO follows its original single-frame Markov formulation [28] and uses only the final frame of the corresponding history window. All methods share the same forecast origin, defined as the time of the final observed frame, and predict the same subsequent target frames. During free-running evaluation, predictions are fed back without ground-truth correction. The SW/RD extrapolation setting uses a 50-step supervised prediction horizon and a 90-step test rollout, with errors evaluated on prediction steps 51–90. All methods use the same train/validation-test splits, spatial resolutions, channel numbers, and evaluation horizons under each setting.

Following the general paradigm of operator learning, we partition each benchmark dataset into a training set and a held-out set. In our experiments, the held-out set is shared between validation and final testing, without reserving a separate validation subset. Checkpoint selection is based exclusively on the training loss, with prediction-loss terms evaluated over the full supervised prediction horizon. For each run, we retain the checkpoint with the lowest training loss and use it for all reported evaluations. Validation/test errors are not used for checkpoint selection.

The supplementary experiments follow the corresponding NS or SW architecture and optimization configurations reported in Appendix A.4. The latent channel widths, retained Fourier modes, internal Euler substeps, optimizer, learning rate, batch size, training epochs, and loss weights remain unchanged. Task-dependent adjustments are limited to the input/output dimensions, data configurations, and number of training rollout windows N_{roll}. Specifically, $N_{\text{roll}} = 7$ for the 70-step NS experiment and $N_{\text{roll}} = 5$ for the 50-step SW extrapolation settings. Table 14 summarizes these configurations.

[8] `https://github.com/scaomath/fourier_neural_operator/tree/master`

For the extended NS experiment, we generate 1,200 trajectories with 80 temporal snapshots each, following the original physical and numerical simulation settings. Methods other than F-FNO increase their history length from 4 to 10 frames, while F-FNO retains its single-frame input. All methods are retrained for the extended 70-step prediction task with aligned forecast origins and target frames. The temporal dimensions of the propagator are adjusted accordingly, while the remaining architecture and optimization hyperparameters follow the standard NS configuration.

For the full-state SW experiment, we retain the original SW model configuration and jointly predict three physical variables, (h, hu, hv), instead of a single scalar field. The encoder input and decoder output channels are adjusted accordingly. For the higher-resolution SW experiment, we use the same governing equations, physical parameters, and data-generation procedure as the original SW benchmark, increasing only the spatial resolution from 128×128 to 256×256. Both SW settings use a 50-step supervised prediction horizon and a 90-step autoregressive evaluation, with errors reported over steps 51–90. Methods other than F-FNO use 10 history frames, whereas F-FNO uses only the final frame; forecast origins and target frames are aligned across methods. All compared methods are retrained under their respective settings.

The number of training rollout windows N_{roll} is adjusted to match the supervised prediction horizon, with $T_{\mathrm{sup}} = TN_{\mathrm{roll}}$, where T denotes the temporal window length. Specifically, the extended NS experiment uses $T = 10$ and $N_{\mathrm{roll}} = 7$ for 70 supervised prediction steps. The full-state and higher-resolution SW experiments use $T = 10$ and $N_{\mathrm{roll}} = 5$ for 50 supervised prediction steps. The standard SW/RD temporal extrapolation experiments likewise use $N_{\mathrm{roll}} = 5$. All other applicable architecture and optimization hyperparameters follow the corresponding NS or SW configurations in Appendix A.4.

Simulation and data-generation settings. The supplementary simulations inherit the physical and numerical settings in Appendix B, except for the explicitly varied quantities. For extended NS, we extend the simulation end time from 20 to 80 without changing the internal step or saved-frame interval, obtaining 80 snapshots at physical times $t = 1, \ldots, 80$. The 1,200 trajectories are split into 1,000 training and 200 held-out trajectories, with the same split used by all compared methods. All methods are retrained to predict the same 70 future frames, using 10 history frames except for F-FNO, which uses only the final frame.

For extended SW and RD, we generate 100 additional test trajectories per system with new initial conditions drawn from the original benchmark distributions. The SW simulation end time is extended from 1 to 2 with 200 output intervals, and the RD end time from 5 to 10 with 201 output times. Both simulations produce 201 snapshots including the initial state. We discard the final snapshot and retain the first 200, at $t_j = j\Delta t_{\mathrm{save}}$, $j = 0, \ldots, 199$: $[0, 1.99]$ for SW and $[0, 9.95]$ for RD. All methods use these same test trajectories and reuse their original checkpoints for the 50-step supervised prediction task, without retraining or adaptation. The 10-frame history and the single-frame exception for F-FNO follow the input protocol above. The 190-step prediction is evaluated on steps 51–190, indexed as extrapolation steps 1–140. We use per-trajectory generation seeds $s \in \{2000, 2001, \ldots, 2099\}$, with one seed per trajectory for each system. These seeds were not used for training or validation. The initial-condition samplers, physical parameters, numerical solvers, and saved-frame intervals remain unchanged; only the simulated time horizon is extended. All compared methods are evaluated on the same extended test trajectories.

Full-state SW retains (h, hu, hv) from the same simulator instead of only h. Higher-resolution SW changes the grid to 256×256, giving $\Delta x = \Delta y = 5/256$, while retaining the numerical scheme, CFL-based time-step control, physical parameters, and saved-frame interval. The SW distribution-shift tests change only the stated dam geometry, gravity, or initial inner water depth. Their remaining numerical settings and 201-to-200 snapshot truncation follow the extended SW protocol.

C.2 Evaluation Metrics

Let $\mathcal{I}$ denote the set of evaluated prediction steps. In this subsection, $T = |\mathcal{I}|$ denotes the number of evaluated snapshots, distinct from the model input-window length. The predicted and ground-truth trajectories restricted to $\mathcal{I}$ are denoted by $\hat{\mathbf{u}}, \mathbf{u} \in \mathbb{R}^{B \times T \times C \times H \times W}$, where B is the batch size, C is the number of physical variables, and $H \times W$ is the spatial resolution. We denote the prediction error by $\mathbf{e} = \hat{\mathbf{u}} - \mathbf{u}$. All metrics are evaluated on the selected interval $\mathcal{I}$, which corresponds to either the full evaluation horizon or a specified subinterval.

Table 14: Configurations of the supplementary experiments. Frames denote retained snapshots per trajectory. Evaluation steps are indexed from the first predicted frame. The Input column lists the history length for methods other than F-FNO, which uses only the final frame; forecast origins and target frames are aligned across methods. "Retrain" denotes training under the corresponding configuration; "Reuse" denotes evaluation with the original checkpoint. Each SW distribution-shift configuration has 100 test trajectories. For the extended SW/RD and shifted SW tests, prediction steps 51–190 correspond to extrapolation steps 1–140.

Experiment	d_u	Resolution	Traj./Frames	Input	Supervised	Rollout	Eval. steps	Weights
Extended NS	1	64×64	1200/80	10	70	70	1–70	Retrain
Full-state SW	3	128×128	1000/100	10	50	90	51–90	Retrain
High-res. SW	1	256×256	1000/100	10	50	90	51–90	Retrain
Extended SW test	1	128×128	100/200	10	50	190	51–190	Reuse
Extended RD test	2	128×128	100/200	10	50	190	51–190	Reuse
Shifted SW test (each)	1	128×128	100/200	10	50	190	51–190	Reuse

For models trained with normalization, predictions are first transformed back to the physical scale before metric computation.

The mean squared error (MSE) measures the average pointwise squared deviation over all batches, time steps, channels, and spatial locations:

$$\mathrm{MSE} = \frac{1}{BTCHW} \sum_{b=1}^{B} \sum_{t=1}^{T} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \left(\hat{u}_{b,t,c,i,j} - u_{b,t,c,i,j}\right)^2 .$$

A smaller MSE indicates better overall pointwise accuracy in the physical space.

The relative L_2 error evaluates the global trajectory error normalized by the magnitude of the ground-truth trajectory:

$$L_2 = \frac{1}{B} \sum_{b=1}^{B} \frac{\|\hat{\mathbf{u}}_b - \mathbf{u}_b\|_2}{\|\mathbf{u}_b\|_2 + \varepsilon},$$

where ε is a small constant for numerical stability. This metric reflects the overall relative prediction error and is useful for comparing results across different PDE systems or scales. For relative L_2, the norm is computed over all temporal, channel, and spatial dimensions within $\mathcal{I}$ for each trajectory, followed by averaging across samples. Full-horizon and interval-wise results use the same definition with different evaluation intervals; interval-wise values are therefore distinct from arithmetic averages of per-frame relative L_2 errors.

The maximum error $E_{\max}$ means batch-level worst-case pointwise error, measuring the largest pointwise deviation at each time step and physical channel, followed by averaging over time and channels:

$$E_{\max} = \frac{1}{CT} \sum_{c=1}^{C} \sum_{t=1}^{T} \max_{b,i,j} \left|\hat{u}_{b,t,c,i,j} - u_{b,t,c,i,j}\right| .$$

This metric emphasizes worst-case local errors and is sensitive to sharp artifacts, local instability, or extreme pointwise deviations.

Since $E_{\max}$ takes a maximum over samples within each batch, all methods are evaluated with a fixed batch size of 2. Within each benchmark setting, the same ordered test set is loaded with shuffling disabled, ensuring identical sample order and batch composition across methods. Reported $E_{\max}$ values are obtained by averaging the batch-level results over all test batches.

The boundary root mean squared error (BRMS) measures the prediction error on the spatial boundary:

$$\mathrm{BRMS} = \frac{1}{BCT} \sum_{b=1}^{B} \sum_{c=1}^{C} \sum_{t=1}^{T} \left[\frac{1}{|\partial\Omega|} \sum_{(i,j)\in\partial\Omega} \left(\hat{u}_{b,t,c,i,j} - u_{b,t,c,i,j}\right)^2 \right]^{1/2},$$

where $\partial\Omega$ denotes the four boundary edges of the two-dimensional grid and $|\partial\Omega| = 2H + 2W - 4$. BRMS reflects how well the model preserves boundary behavior, which is important for PDE systems where boundary regions strongly affect the subsequent evolution.

In addition to physical-space metrics, we also report Fourier-domain errors to evaluate spectral fidelity. For each predicted field, a two-dimensional FFT is applied only along the spatial dimensions:

$$\widehat{\mathbf{e}}^{F}_{b,t,c} = \mathcal{F}_{x,y}\left(\hat{\mathbf{u}}_{b,t,c} - \mathbf{u}_{b,t,c}\right).$$

The Fourier coefficients are grouped into radial frequency bins. For a frequency radius bin k, define

$$\mathcal{R}_k = \left\{(p,q) : \left\lfloor \sqrt{p^2+q^2} \right\rfloor = k \right\}.$$

The radial Fourier error is computed as

$$E_F(c,k,t) = \frac{1}{HW}\left[\frac{1}{B}\sum_{b=1}^{B}\sum_{(p,q)\in\mathcal{R}_k}\left|\widehat{e}^{F}_{b,t,c,p,q}\right|^2\right]^{1/2}.$$

Following the implementation, the frequency bands are divided using $k_{\text{low}} = 4$ and $k_{\text{high}} = 12$. The low-, mid-, and high-frequency errors are then defined as

$$f_{\text{low}} = \frac{1}{CT|\mathcal{K}_{\text{low}}|}\sum_{c,t}\sum_{k<4} E_F(c,k,t),$$

$$f_{\text{mid}} = \frac{1}{CT|\mathcal{K}_{\text{mid}}|}\sum_{c,t}\sum_{4\le k<12} E_F(c,k,t),$$

$$f_{\text{high}} = \frac{1}{CT|\mathcal{K}_{\text{high}}|}\sum_{c,t}\sum_{k\ge 12} E_F(c,k,t).$$

Here, f_{low} mainly reflects errors in large-scale dominant structures, f_{mid} measures errors in intermediate spatial patterns, and f_{high} captures fine-scale details, local oscillations, and high-frequency artifacts.

Finally, the overall Fourier error is reported as

$$f_{\text{mse}} = \frac{1}{CTK}\sum_{c=1}^{C}\sum_{t=1}^{T}\sum_{k=0}^{K-1} E_F(c,k,t),$$

where $K = \min(H/2, W/2)$ is the number of radial frequency bins. This metric summarizes the overall spectral discrepancy between the predicted and ground-truth trajectories. For all metrics used in this work, lower values indicate better prediction performance.

For teacher-forced one-step evaluation, we compute the relative error independently at each prediction step and then average over the evaluated steps:

$$L_2^{\text{TF}} = \frac{1}{B|\mathcal{I}|}\sum_{b=1}^{B}\sum_{t\in\mathcal{I}}\frac{\|\hat{u}_{b,t} - u_{b,t}\|_2}{\|u_{b,t}\|_2 + \epsilon}.$$

The norms in this expression are computed over the physical channels and spatial dimensions of a single frame. This differs from the interval-wise trajectory relative L_2, which aggregates the temporal dimension before normalization.

C.3 Computational Resources

All experiments were conducted on NVIDIA RTX 4090 GPUs with 24GB memory. Unless otherwise stated, each training run used a single GPU. For the long-horizon prediction experiments, the main SSP model required approximately 15 hours for Navier–Stokes, 20 hours for Shallow–Water, and 20 hours for Reaction–Diffusion per random seed. For the temporal extrapolation experiments, the SSP model required approximately 13 hours on Shallow–Water and Reaction–Diffusion per random seed.

GPU-hours are computed as wall-clock training time multiplied by the number of GPUs used. Inference-time memory and latency are reported separately in Table 24.

Table 15: Hyperparameters and model weight sizes of CALM-PDE.

Parameter	SW / SW-Ext.	RD / RD-Ext.	NS
Encoder Channels	[32, 64, 128]	[32, 64, 128]	[32, 64, 128]
Decoder Channels	[64, 32]	[64, 32]	[64, 32]
Encoder Query Points	[1024, 256, 16]	[1024, 256, 16]	[1024, 256, 16]
Decoder Query Points	[256, 1024]	[256, 1024]	[256, 1024]
Encoder Receptive Fields	[0.01, 0.01, 0.2]	[0.01, 0.01, 0.2]	[0.01, 0.01, 0.2]
Decoder Receptive Fields	[0.2, 0.01]	[0.2, 0.01]	[0.2, 0.01]
Encoder Softmax Temperatures	[1.0, 1.0, 0.1]	[1.0, 1.0, 0.1]	[1.0, 1.0, 0.1]
Decoder Softmax Temperatures	[0.1, 1.0]	[0.1, 1.0]	[0.1, 1.0]
Latent Dimension	128	128	128
Neural ODE Hidden Dimension	256	256	256
Attention Heads	8	8	8
Batch Size	2	2 / 4	8
Epochs	500	500	500
Learning Rate	5×10^{-4} / 2.5×10^{-4}	5×10^{-4}	2.5×10^{-4}
Weight size (MB)	8.96	9.01	8.94

C.4 CALM

CALM-PDE. CALM-PDE is a latent-space neural surrogate for time-dependent PDEs, which represents physical fields with latent points and models the temporal evolution in the compressed latent space [20]. Its encoder and decoder are built on continuous adaptive convolutions, allowing the model to map between physical fields and latent representations before and after latent dynamics modeling. Different from our method, which explicitly structures the compact propagator in the spectral domain using a frequency-conditioned linear modal backbone and a nonlinear spectral closure, CALM-PDE mainly relies on continuous-convolution-based latent representation and a general latent dynamics model.

Reproduction settings of CALM-PDE. We follow their original setup and keep the training strategy. CALM-PDE was reproduced based on its official implementation, with only the data interface and evaluation scripts adapted to our benchmark format. All experiments use the common data splits, input lengths, supervised prediction horizons, test horizons, spatial resolutions, and evaluation ranges summarized in Table 13. For extrapolation experiments, the architecture and main training configuration are kept consistent with the corresponding standard setting, while the evaluation horizon follows Table 13. The detailed hyperparameters and model weight sizes are summarized in Table 15.

C.5 LNO

LNO. LNO is a latent-space neural operator that learns PDE solution operators in a compact, data-driven latent representation rather than directly in the original geometric space [18]. Its core module is Physics-Cross-Attention (PhCA), which maps input samples from the geometric space to a learnable latent space, applies Transformer blocks to model the operator in the latent space, and decodes the latent representation back to queried physical positions. Different from our method, which explicitly structures latent propagation in the spectral domain with a frequency-conditioned linear modal backbone and a nonlinear spectral closure, LNO mainly relies on attention-based latent encoding and Transformer-based latent operator learning without imposing an explicit frequency-wise propagation structure.

Reproduction settings of LNO. We follow their original setup and keep the training strategy. LNO was reproduced based on the PyTorch implementation for the forward problem, with the core model structure kept unchanged and only the data interface, normalization, split files, and evaluation scripts adapted to our benchmark format. All experiments use the common data splits, input lengths, supervised prediction horizons, test horizons, spatial resolutions, and evaluation ranges summarized in Table 13. RD and RD-Ext. are reported as "/" because our local adaptation did not produce stable and reliable results on the multi-variable Reaction–Diffusion benchmark. The detailed hyperparameters and model weight sizes are summarized in Table 16.

Table 16: Hyperparameters and model weight sizes of LNO.

Parameter	SW / SW-Ext.	RD / RD-Ext.	NS
Latent Tokens	128	/	128
Embedding Dimension	192	/	192
Attention Blocks	8	/	8
Attention Heads	8	/	8
Branch Projector	$12 \to 192 \to 192$	/	$6 \to 192 \to 192$
Trunk Projector	$2 \to 192 \to 192$	/	$2 \to 192 \to 192$
Attention Projector	$192 \to 192 \to 128$	/	$192 \to 192 \to 128$
Decoder MLP	$192 \to 192 \to 10$	/	$192 \to 192 \to 4$
Attention Type	Scaled Dot-Product	/	Scaled Dot-Product
Position Input	(x, y)	/	(x, y)
Batch Size	4	/	8
Epochs	500	/	500
Learning Rate	1×10^{-3}	/	1×10^{-3}
Weight size (MB)	11.40	/	11.39

Table 17: Hyperparameters and model weight sizes of U-FNO.

Parameter	SW / SW-Ext.	RD / RD-Ext.	NS
Modes_x	12	12	12
Modes_y	12	12	12
Modes_t	12	12	12
Width	36 / 28	28 / 28	28
Fourier Blocks	6	6	6
U-Fourier Blocks	3	3	3
U-Net Branch	last 3	last 3	last 3
U-Net Channels	36 / 28	28 / 28	28
U-Net Depth	3 / 3	3 / 3	3 / 3
Input Features	$10 + 3$	$10 \times 2 + 3$	$4 + 3$
Output Channels	1	2	1
Batch Size	1 / 8	2 / 4	2
Epochs	500	500	500
Learning Rate	1×10^{-3}	1×10^{-3}	1×10^{-3}
Weight size (MB)	437.96 / 264.95	264.95	243.27

C.6 U-FNO

U-FNO. U-FNO is an enhanced Fourier-neural-operator-based model that augments the original FNO architecture with local convolutional feature extraction [26]. Its core component is the U-Fourier layer, where a U-Net branch is combined with the Fourier layer to capture local and high-frequency information that may be insufficiently represented by truncated Fourier modes. Different from our method, which explicitly builds a compact spectral propagator with a frequency-conditioned linear modal backbone and a nonlinear spectral closure, U-FNO mainly improves the representation capacity of Fourier operator layers and does not impose an explicit structured modal propagation law.

Reproduction settings of U-FNO. We follow their original setup and keep the training strategy. U-FNO was reproduced based on the public implementation, with the data interface, normalization, input/output channel construction, and evaluation scripts adapted to our benchmark format. All experiments use the common data splits, input lengths, supervised prediction horizons, test horizons, spatial resolutions, and evaluation ranges summarized in Table 13. Standard and extrapolation experiments use separate model configurations, as reported in Table 17, with evaluation horizons following Table 13. The detailed hyperparameters and model weight sizes are summarized in Table 17.

Table 18: Hyperparameters and model weight sizes of F-FNO.

Parameter	SW / SW-Ext.	RD / RD-Ext.	NS
Modes	16	16	16
Width	64	64	64
Layers	16	16	4
Fourier Factorization	Yes	Yes	Yes
Input Dimension	3	4	3
Output Dimension	1	2	1
MLP Factor	4	4	4
Weight Normalization	Yes	Yes	Yes
Gain	0.1	0.1	0.1
Dropout	0.0	0.0	0.0
Input Dropout	0.0	0.0	0.0
Batch Size	24	8	8
Epochs	101	101	101
Learning Rate	2.5×10^{-3}	2.5×10^{-3}	2.5×10^{-3}
Weight size (MB)	9.40	9.40	4.70

C.7 F-FNO

F-FNO. F-FNO is a spectral neural operator and an improved variant of FNO for learning PDE solution maps in Fourier space [28]. Its core mechanism is to factorize Fourier transforms along different spatial dimensions and combine separable spectral layers with improved residual connections and pointwise feed-forward layers, which improves the scalability of Fourier operator networks. Different from our method, which explicitly models compact latent propagation with a frequency-conditioned linear modal backbone and a nonlinear spectral closure, F-FNO mainly improves the spectral convolution layer itself and does not impose an explicit structured modal propagation law.

Reproduction settings of F-FNO. We follow their original setup and keep the training strategy. The original study reports that additional history frames did not improve performance on TorusLi [28]; we therefore retain its single-frame Markov input configuration. F-FNO was reproduced based on the FourierFlow-style implementation of Factorized Fourier Neural Operator, with the data interface, normalization, input/output channel construction, checkpoint loading, and evaluation scripts adapted to our benchmark format. F-FNO uses only the final frame of the history window used by the other methods; the input dimensions in Table 18 correspond to the physical variables at that frame plus the two spatial coordinates. All methods share the same forecast origins and target frames, as well as the data splits, supervised prediction horizons, test horizons, spatial resolutions, and evaluation ranges summarized in Table 13. For extrapolation experiments, the model configuration follows the corresponding standard setting, while the evaluation horizon follows Table 13. The detailed hyperparameters and model weight sizes are summarized in Table 18.

C.8 FNO

FNO. FNO is a Fourier-based neural operator that learns mappings between function spaces for PDE solution approximation [6]. Its core component is the Fourier layer, which transforms the lifted feature field into Fourier space, applies learnable linear transformations on a truncated set of low-frequency modes, and combines the result with a pointwise transformation in physical space. Different from our method, which separates spatial representation from compact recurrent propagation and explicitly structures the retained spectral modes with a frequency-conditioned linear modal backbone and a nonlinear spectral closure, FNO mainly learns a general spectral convolutional operator without an explicit structured modal propagation rule.

Reproduction settings of FNO. We follow their original setup and keep the training strategy. FNO was reproduced based on the two-dimensional Fourier Neural Operator implementation, with the data interface, normalization, input/output channel construction, checkpoint loading, and evaluation scripts adapted to our benchmark format. All experiments use the common data splits, input lengths, supervised prediction horizons, test horizons, spatial resolutions, and evaluation ranges summarized

Table 19: Hyperparameters and model weight sizes of FNO.

Parameter	SW / SW-Ext.	RD / RD-Ext.	NS
Modes_x	12	12	12
Modes_y	12	12	12
Width	20	20	20
Fourier Layers	4	4	4
Input Dimension	12	22	6
Projection Head	$20 \to 128 \to 1$	$20 \to 128 \to 2$	$20 \to 128 \to 1$
Batch Size	8	12	5
Epochs	500	500	500
Learning Rate	1×10^{-3}	1×10^{-3}	1×10^{-3}
Weight size (MB)	10.60	10.60	11.14

Table 20: Hyperparameters and model weight sizes of KNO.

Parameter	SW / SW-Ext.	RD / RD-Ext.	NS
Operator Size o	32	32	32
Modes f	16	16	16
Decompose	8	8	8
t_{len}	10	20	4
Autoencoder	Conv2d / MLP	MLP	MLP
High-frequency Complement	1×1 Conv	1×1 Conv	1×1 Conv
Batch Size	8	8	16
Epochs	500	500	500
Learning Rate	1×10^{-3}	1×10^{-3}	1×10^{-3}
Weight size (MB)	2.11	2.11	2.11

in Table 13. For extrapolation experiments, the model configuration follows the corresponding standard setting, while the evaluation horizon follows Table 13. The detailed hyperparameters and model weight sizes are summarized in Table 19.

C.9 KNO

KNO. KNO is a Koopman-theory-based neural operator for learning the evolution of nonlinear PDE systems [22]. Its core mechanism is to map the PDE state into an observable space, apply Fourier transform, approximate Koopman evolution on truncated low-frequency modes, and complement the discarded high-frequency information with an additional convolutional branch. Different from our method, which constructs a compact spectral propagation state and evolves it with a frequency-conditioned modal backbone and a nonlinear spectral closure, KNO mainly relies on observable-space Koopman linearization and low-/high-frequency fusion.

Reproduction settings of KNO. We follow their original setup and keep the training strategy. KNO was reproduced using the local KoopmanLab implementation, with the data interface, checkpoint loading, split files, and evaluation scripts adapted to our benchmark format. All experiments use the common data splits, input lengths, supervised prediction horizons, test horizons, spatial resolutions, and evaluation ranges summarized in Table 13. For extrapolation experiments, the model configuration follows the corresponding standard setting, while the evaluation horizon follows Table 13. The detailed hyperparameters and model weight sizes are summarized in Table 20.

C.10 UNO

UNO. UNO is a U-shaped neural operator designed to learn mappings between infinite-dimensional function spaces with improved scalability and memory efficiency [27]. Its core mechanism is to progressively contract the spatial representation in the encoder path, expand it back in the decoder path, and connect matching resolutions with skip connections, while using Fourier-based integral operator layers as the main building blocks. Different from our method, which explicitly constructs

Table 21: Hyperparameters and model weight sizes of UNO.

Parameter	SW / SW-Ext.	RD / RD-Ext.	NS
Modes	$[[8,8],[8,8],[4,4],[8,8],[8,8]]$		
Width	64	64	64
Lifting Dimension	256	256	256
Projection Dimension	256	256	256
Operator Blocks	5	5	5
Encoder / Bottleneck / Decoder Blocks	$2/1/2$	$2/1/2$	$2/1/2$
U-shaped Scaling	$[[1,1],[0.5,0.5],[1,1],[2,2],[1,1]]$		
Horizontal Skips	$\{4:0,\ 3:1\}$		
Domain Padding	0.2	0.2	0.2
Activation	GELU	GELU	GELU
Batch Size	1	16	10
Epochs	500	500	500
Learning Rate	1×10^{-3}	1×10^{-3}	1×10^{-3}
Weight size (MB)	5.45	5.45	5.68

a compact spectral propagator with frequency-conditioned linear modal evolution and nonlinear spectral closure, UNO mainly improves the global encoder–decoder architecture of neural operators without imposing an explicit structured modal propagation law.

Reproduction settings of UNO. We follow their original setup and keep the training strategy. UNO was reproduced using an implementation based on the U-shaped neural operator architecture, with the data interface, normalization, input/output channel construction, checkpoint loading, and evaluation scripts adapted to our benchmark format. All experiments use the common data splits, input lengths, supervised prediction horizons, test horizons, spatial resolutions, and evaluation ranges summarized in Table 13. For extrapolation experiments, the model configuration follows the corresponding standard setting, while the evaluation horizon follows Table 13. The detailed hyperparameters and model weight sizes are summarized in Table 21.

D Additional Experiment

D.1 Long-horizon prediction experiment details

Table 22 provides the full long-horizon forecasting results on the Navier–Stokes benchmark over rollout steps 1–12. Compared with the compact results reported in the main text, this table additionally includes MSE, BRMS, and the mid-/high-frequency spectral errors, giving a more complete view of both physical-space accuracy and spectral consistency. Our method achieves the best MSE, relative L_2, BRMS, f_{low}, f_{mid}, and f_{high}, and obtains the second-best $E_{\max}$, with a parameter-weight storage size of only 11.41 MB. Compared with F-FNO, the strongest baseline in terms of MSE, relative L_2, BRMS, and spectral errors, our method reduces MSE from 1.00×10^{-2} to 6.74×10^{-3}, relative L_2 from 7.57×10^{-2} to 5.50×10^{-2}, and BRMS from 6.89×10^{-2} to 5.74×10^{-2}, corresponding to relative improvements of approximately 32.6%, 27.3%, and 16.7%, respectively. In the frequency domain, our method reduces f_{low}, f_{mid}, and f_{high} by approximately 49.6%, 17.3%, and 6.7% compared with F-FNO, indicating stronger preservation of both dominant and finer spectral components during autoregressive rollout. Although KNO obtains the lowest $E_{\max}$, its much larger relative L_2, BRMS, and spectral errors indicate weaker global trajectory alignment and poorer spectral consistency. F-FNO and CALM remain competitive on several metrics, but their errors are consistently higher than ours in the main physical-space and spectral-domain measures. Overall, these results show that the proposed structured propagator improves Navier–Stokes long-horizon forecasting by reducing global rollout error, suppressing background error accumulation, and preserving the multi-scale spectral organization of the flow dynamics.

Figure 9 further compares the predicted vorticity fields and pointwise absolute-error maps at different rollout steps. The ground-truth sequence exhibits progressively sharpened vortex bands and coherent large-scale flow structures, making this benchmark sensitive to both phase drift and local amplitude error. Several baselines can reproduce the coarse flow pattern at early steps, but their error maps

become increasingly structured as the rollout proceeds, especially near high-gradient vortex regions. FNO, UNO, LNO, CALM, and U-FNO show visible error accumulation in later snapshots, while KNO also presents non-negligible residual structures despite its favorable $E_{\max}$. In contrast, our method produces consistently weaker error maps across the displayed rollout steps, indicating better alignment with the evolving vorticity field and stronger suppression of accumulated local errors. This qualitative comparison supports the quantitative results in Table 22, showing that the proposed propagator improves both global trajectory accuracy and visual fidelity for Navier–Stokes forecasting.

Table 23 reports the full long-horizon forecasting results on the Shallow–Water benchmark over rollout steps 1–90. This setting is substantially longer than the Navier–Stokes rollout and is especially sensitive to accumulated phase error, wavefront drift, and spectral distortion under autoregressive prediction. Our method achieves the best MSE, relative L_2, $E_{\max}$, f_{low}, f_{mid}, and f_{high}, with a parameter-weight storage size of only 5.90 MB. Compared with F-FNO, which is the strongest baseline in terms of MSE, relative L_2, $E_{\max}$, and high-frequency error, our method reduces MSE from 1.09×10^{-6} to 4.78×10^{-7}, relative L_2 from 9.28×10^{-4} to 6.56×10^{-4}, and $E_{\max}$ from 1.69×10^{-2} to 9.72×10^{-3}, corresponding to relative improvements of approximately 56.1%, 29.3%, and 42.5%, respectively. In the frequency domain, our method reduces f_{low}, f_{mid}, and f_{high} by approximately 63.8%, 64.3%, and 4.0% compared with F-FNO, indicating stronger preservation of the dominant wave structure and mid-frequency spectral components during the 90-step rollout. Although F-FNO obtains the lowest BRMS, its larger global trajectory and worst-case pointwise errors indicate that superior boundary accuracy does not necessarily translate into more accurate long-horizon forecasting over the full spatial domain. CALM and LNO achieve competitive low-frequency errors, but their physical-space errors remain higher, indicating weaker reconstruction of the full wave field. KNO exhibits much larger errors in both physical and spectral metrics, suggesting that its learned latent evolution is less stable on this wave-dominated benchmark. Overall, the results show that the proposed structured propagator provides a favorable balance between compactness, global trajectory accuracy, worst-case error control, and spectral preservation for long-horizon Shallow–Water prediction.

Figure 6 further visualizes the prediction snapshots and pointwise absolute-error maps from $t = 10$ to $t = 90$. The ground-truth sequence shows a radially expanding wave structure with a clear annular profile, which requires the model to maintain both phase consistency and radial symmetry over a long rollout horizon. Most baseline methods reproduce the coarse wave pattern at early steps, but their error maps reveal increasingly structured artifacts, such as ring-shaped residuals, grid-like error patterns, and wavefront misalignment as prediction proceeds. In contrast, our method remains visually closer to the ground truth across the full rollout and produces substantially weaker absolute-error maps, with fewer structured residuals around the moving wavefront. This qualitative comparison is consistent with the results in Table 23, confirming that the proposed propagator better controls error accumulation and preserves the dominant wave morphology during long-horizon autoregressive forecasting.

Table 25 presents the full long-horizon forecasting results on the Reaction–Diffusion benchmark over rollout steps 1–90. Different from the wave-dominated Shallow–Water case, this benchmark contains two coupled variables and stronger nonlinear reaction dynamics, making long-horizon rollout sensitive to both cross-channel error accumulation and spectral distortion. LNO fails to train reliably in this setting, which is likely related to the stronger multivariable coupling in the Reaction–Diffusion system. Our method achieves the best performance across all reported metrics, including MSE, relative L_2, $E_{\max}$, BRMS, f_{low}, f_{mid}, and f_{high}, with a parameter-weight storage size of only 5.00 MB. Compared with the strongest baseline for each metric, our method reduces MSE from 3.10×10^{-6} to 5.26×10^{-7}, relative L_2 from 1.46×10^{-2} to 7.46×10^{-3}, $E_{\max}$ from 2.02×10^{-2} to 6.71×10^{-3}, and BRMS from 3.71×10^{-3} to 9.61×10^{-4}. These correspond to relative improvements of approximately 83.0%, 48.9%, 66.8%, and 74.1%, respectively, showing a substantial reduction in both global and local physical-space errors. In the frequency domain, our method reduces f_{low}, f_{mid}, and f_{high} by approximately 21.3%, 75.8%, and 67.1%, respectively, relative to the strongest baseline for each metric, indicating more accurate preservation of both dominant and fine-scale spectral components. Although U-FNO obtains the second-best relative L_2 and F-FNO is competitive on most other metrics, their worst-case, background, and spectral errors remain clearly higher than those of our method. CALM obtains a competitive low-frequency error, but its physical-space errors are much larger, indicating that the learned latent dynamics do not sufficiently control nonlinear rollout error in this coupled system. Overall, these results demonstrate that the proposed structured propagator is particularly effective for multivariable Reaction–Diffusion forecasting, where compact

propagation states, frequency-conditioned modal evolution, and nonlinear spectral closure jointly suppress error accumulation during long autoregressive rollout.

Figures 7 and 8 further show qualitative comparisons for the two Reaction–Diffusion channels. The ground-truth sequences exhibit gradually evolving local patterns, so accurate rollout requires preserving both the spatial distribution of the reaction structures and the coupling consistency between channels. FNO and KNO show increasingly pronounced structured residuals as the rollout proceeds, indicating accumulated pattern distortion and phase mismatch in the predicted reaction fields. UNO and CALM reduce part of the error but still produce visible residual textures in later steps, especially around regions with sharper local variation. U-FNO remains visually competitive, but the quantitative results in Table 25 show that its worst-case and spectral errors are still higher than those of our method. In contrast, our method maintains low-magnitude and weakly structured error maps across both channels, which indicates better suppression of background error accumulation and more stable preservation of coupled reaction–diffusion patterns. Together, the quantitative and visual results support the importance of combining a compact propagation state with frequency-conditioned modal evolution and nonlinear spectral closure for multivariable reaction–diffusion dynamics.

Table 22: Long-horizon prediction results on the Navier–Stokes benchmark. Errors are computed over rollout steps 1–12. Best results are in **bold**, and second-best results are underlined.

Method	Weight Size	MSE	L2	$E_{\max}$	BRMS	f_{low}	f_{mid}	f_{high}
FNO	11.14MB	3.62E-02	1.42E-01	6.68E-01	1.46E-01	5.94E-02	5.90E-02	2.63E-02
U-FNO	243.27MB	3.04E-02	1.50E-01	1.44E+00	1.53E-01	2.38E-02	2.23E-02	9.26E-03
F-FNO	4.70MB	1.00E-02	7.57E-02	6.87E-01	6.89E-02	1.09E-02	8.81E-03	5.54E-03
UNO	5.68MB	3.28E-02	1.36E-01	1.19E+00	1.55E-01	2.16E-02	2.31E-02	9.91E-03
KNO	2.11MB	1.03E-02	7.77E-02	**3.82E-01**	9.43E-02	5.60E-02	4.42E-02	2.73E-02
LNO	11.39MB	3.13E-02	1.32E-01	1.16E+00	1.39E-01	2.18E-02	2.20E-02	9.83E-03
CALM	8.94MB	1.23E-02	8.82E-02	9.96E-01	1.13E-01	1.40E-02	1.47E-02	6.84E-03
Ours	11.41MB	**6.74E-03**	**5.50E-02**	5.38E-01	**5.74E-02**	**5.49E-03**	**7.29E-03**	**5.17E-03**

Table 23: Long-horizon prediction results on the Shallow Water benchmark. Errors are computed over rollout steps 1–90. Best results are in **bold**, and second-best results are underlined.

Method	Weight Size	MSE	L2	$E_{\max}$	BRMS	f_{low}	f_{mid}	f_{high}
FNO	10.60MB	9.60E-05	5.10E-03	8.00E-02	1.60E-03	3.37E-04	3.63E-04	4.05E-04
U-FNO	437.96MB	3.50E-05	4.99E-03	6.04E-02	1.08E-03	1.36E-03	4.18E-04	1.17E-04
F-FNO	9.40MB	1.09E-06	9.28E-04	1.69E-02	**5.04E-06**	6.99E-05	1.10E-04	4.03E-05
UNO	5.45MB	4.00E-06	1.76E-03	1.71E-02	2.24E-04	7.40E-05	1.47E-04	8.40E-05
KNO	2.11MB	1.04E-04	9.76E-03	3.12E-02	7.88E-03	3.89E-03	2.47E-03	2.12E-03
LNO	11.40MB	5.96E-06	2.31E-03	2.11E-02	1.69E-03	6.12E-05	7.40E-05	1.07E-04
CALM	8.96MB	6.04E-06	2.25E-03	2.41E-02	1.29E-04	3.61E-05	1.22E-04	1.32E-04
Ours	5.90MB	**4.78E-07**	**6.56E-04**	**9.72E-03**	1.91E-04	**2.53E-05**	**3.93E-05**	**3.87E-05**

Extended Navier–Stokes rollout. The original NS benchmark contains only 20 temporal snapshots per trajectory, limiting the evaluation of long-horizon autoregressive error accumulation. To extend the evaluation, we numerically simulate 1,200 trajectories, each containing 80 snapshots, at viscosity $\nu = 10^{-5}$ using the same physical and numerical data-generation settings as the original benchmark. SSP, F-FNO, KNO, and CALM are retrained on the extended dataset for the same 70-step prediction task. SSP, KNO, and CALM receive the first 10 frames, whereas F-FNO receives only the tenth frame. All methods autoregressively predict the same subsequent 70 frames.

Table 26 reports the errors over the full 70-step rollout, while Table 27 reports the same metrics evaluated separately over three successive rollout intervals. SSP achieves the lowest errors across all seven full-horizon metrics, reducing relative L_2 from 1.56×10^{-1} (CALM) to 1.10×10^{-1}, a relative reduction of 29.5%. It also obtains the lowest L_2, $E_{\max}$, and f_{low} errors in all three rollout intervals and exhibits the smallest absolute increase between the first and final intervals for these metrics. These observations provide empirical evidence of reduced error accumulation over the evaluated 70-step horizon. The solver, initial-condition sampler, saved-frame interval, simulation time range,

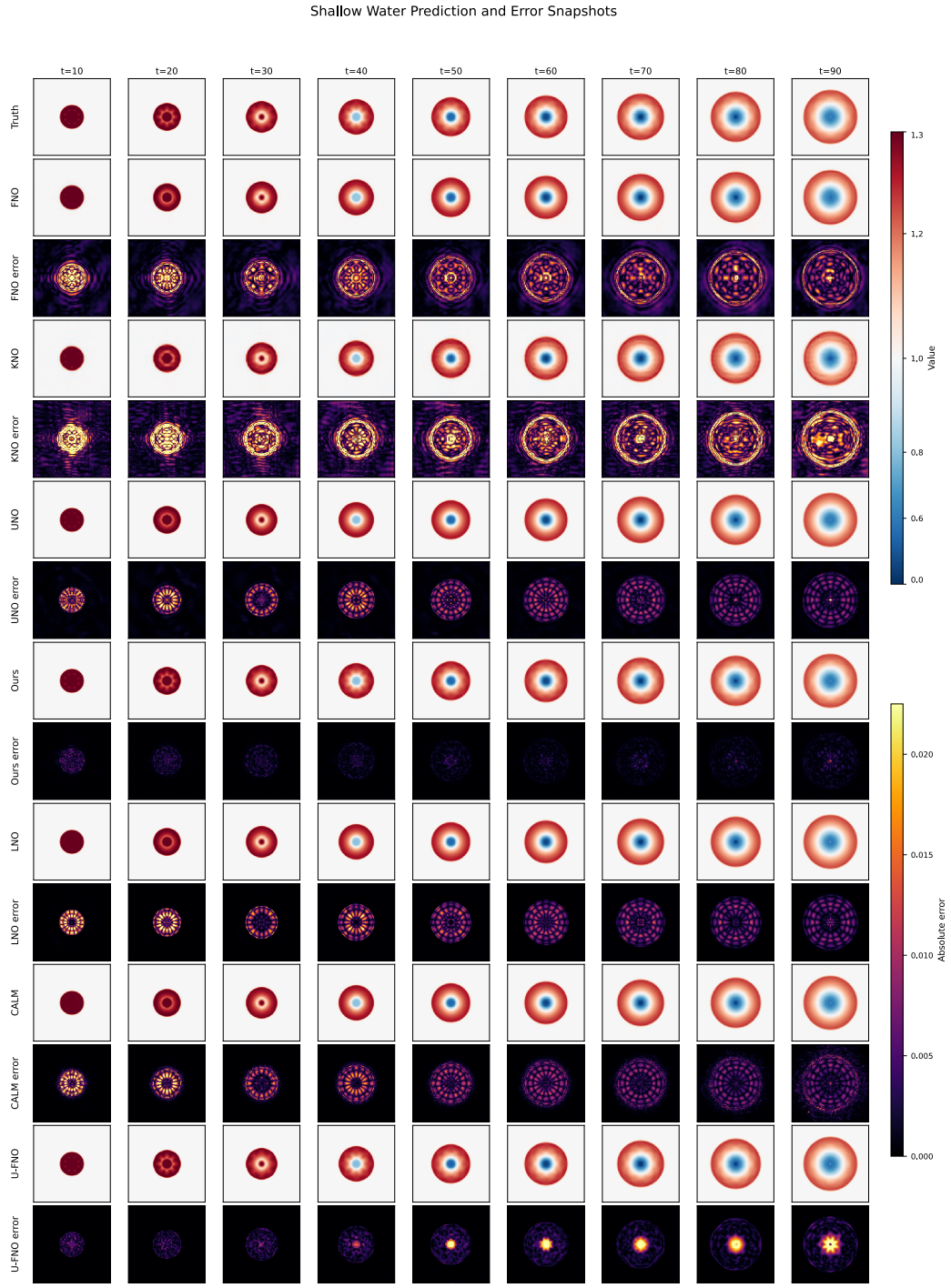


Figure 6: Ground-truth states and absolute-error maps on the Shallow–Water benchmark across representative rollout steps from $t = 10$ to $t = 90$. Each pair of rows shows one method's predicted state and its corresponding pointwise absolute-error map, with the ground truth shown in the first row.

Table 24: Inference efficiency on Shallow–Water under the same autoregressive rollout setting with batch size 1. Memory denotes peak GPU usage with gradients disabled, and time denotes total inference time. Best results are in **bold**, and second-best results are underlined.

Metric	FNO	U-FNO	F-FNO	UNO	KNO	LNO	CALM	Ours
Weight Size (MB)	10.60	437.96	9.40	5.45	**2.11**	11.40	8.96	5.90
Inference memory (GB)	**0.04**	2.72	0.08	0.09	0.09	0.12	2.04	0.17
Inference time (s)	13.78	9.99	15.17	**4.06**	20.20	8.73	13.70	9.55

Table 25: Long-horizon prediction results on the Reaction–Diffusion benchmark. Errors are computed over rollout steps 1–90. LNO fails to train reliably on this dataset, likely due to stronger multivariable coupling. Best results are in **bold**, and second-best results are underlined.

Method	**Weight Size**	**MSE**	**L2**	$E_{\max}$	**BRMS**	f_{low}	f_{mid}	f_{high}
FNO	10.60MB	5.11E-04	2.57E-01	6.72E-02	1.71E-02	1.28E-02	6.33E-03	5.17E-03
U-FNO	264.95MB	1.12E-04	1.46E-02	3.53E-02	2.81E-02	1.88E-04	1.14E-03	4.49E-04
F-FNO	9.40MB	3.10E-06	1.52E-02	2.02E-02	3.71E-03	2.69E-04	2.14E-04	4.04E-05
UNO	5.45MB	2.63E-03	8.38E-02	3.29E-01	7.60E-02	4.18E-03	5.87E-03	1.73E-03
KNO	2.11MB	4.96E-04	2.51E-01	6.85E-02	1.74E-02	1.09E-02	5.84E-03	3.46E-03
LNO	11.40MB	–	–	–	–	–	–	–
CALM	9.01MB	2.28E-04	1.72E-01	2.30E-01	6.66E-02	5.40E-04	7.46E-04	8.76E-04
Ours	5.00MB	**5.26E-07**	**7.46E-03**	**6.71E-03**	**9.61E-04**	**1.48E-04**	**5.17E-05**	**1.33E-05**

and data split are specified in Appendices B and C.1; the extension changes the trajectory duration rather than the internal integration step.

D.2 Extrapolation experiment details

Table 28 reports temporal extrapolation results on the Shallow–Water benchmark, where errors are evaluated only on rollout steps 51–90 beyond the supervised training horizon. This setting is more challenging than standard long-horizon prediction because the model must continue rolling out after the range directly constrained during training, so unstable propagators can rapidly accumulate phase errors, background artifacts, or spectral drift. FNO and LNO suffer from severe error growth, while UNO exhibits numerical divergence, indicating that their learned updates are not reliable under repeated autoregressive composition beyond the training horizon. F-FNO achieves the best MSE, relative L_2, and f_{low}, showing strong global extrapolation accuracy and good preservation of the dominant low-frequency component. However, its larger $E_{\max}$, BRMS, f_{mid}, f_{high}, and f_{mse} indicate that its globally accurate predictions still contain noticeable local deviations, weaker boundary consistency, and fine-scale spectral distortion. CALM remains stable and obtains the second-best $E_{\max}$, f_{high}, and f_{mse}, but its MSE, relative L_2, BRMS, and mid-frequency error are clearly worse than those of the strongest methods. Our method nearly matches F-FNO on MSE and relative L_2, and obtains the second-best f_{low}, indicating comparable large-scale extrapolation quality. More importantly, it achieves the best $E_{\max}$, BRMS, f_{mid}, f_{high}, and f_{mse}, with a particularly large reduction in BRMS. These results suggest that although F-FNO is slightly better in global and low-frequency extrapolation accuracy, our method provides more balanced beyond-horizon behavior by maintaining comparable large-scale prediction quality while substantially improving local stability, boundary-error control, and mid-to-high-frequency spectral consistency.

To further evaluate SSP under coupled multivariable dynamics, we consider the full three-variable Shallow–Water system, jointly predicting water depth h and horizontal momenta hu and hv. We numerically generate 1,000 trajectories with 100 snapshots each at a spatial resolution of 128×128. All methods are retrained with three physical input and output channels with a 50-step supervised prediction horizon and a 90-step autoregressive test rollout, using 10 history frames except for F-FNO, which uses only the final frame. As reported in Table 29, SSP achieves the lowest errors across all seven metrics for each variable over prediction steps 51–90, demonstrating its effectiveness in forecasting coupled physical states beyond the original single-field setting.

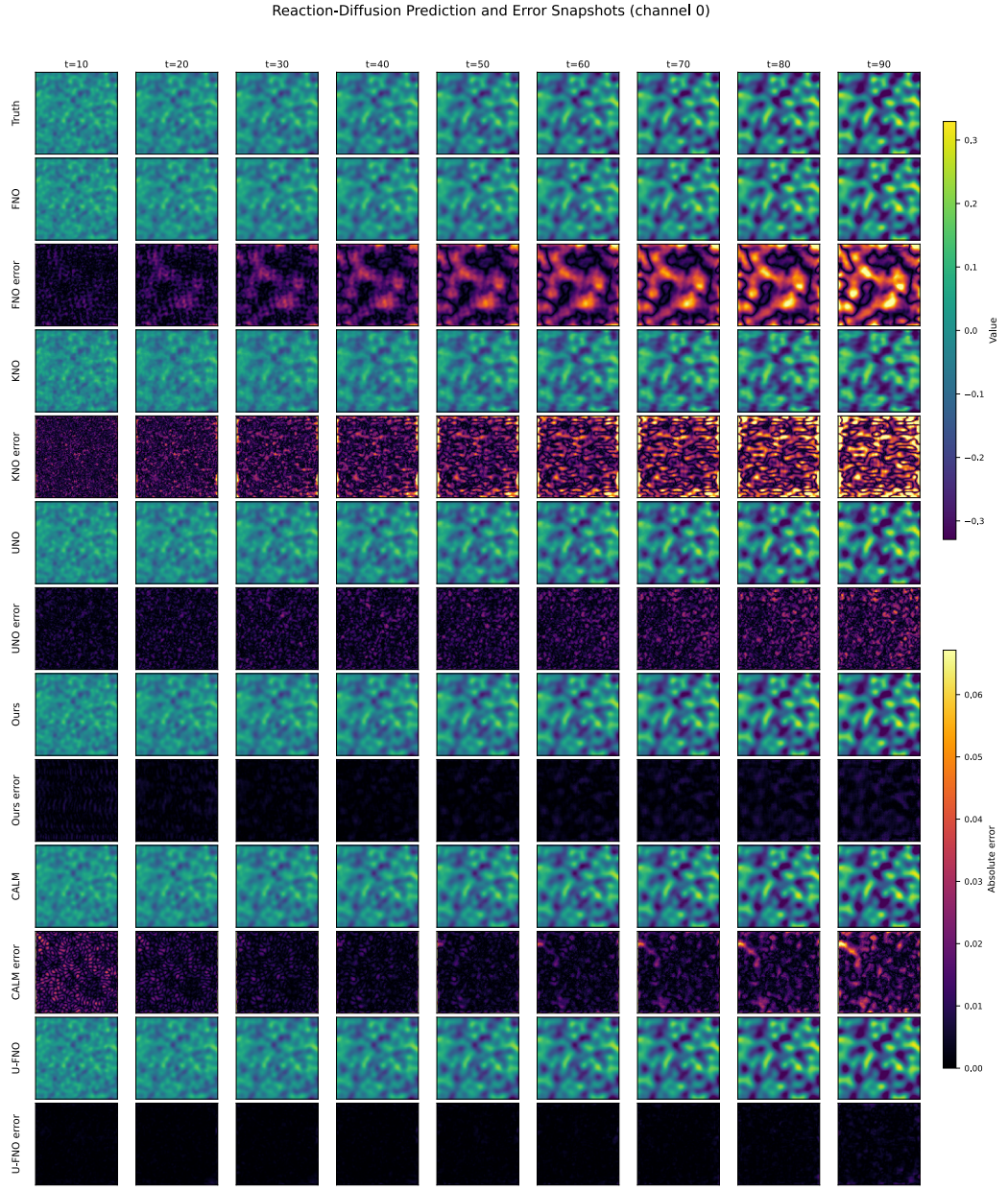


Figure 7: Ground-truth states and absolute-error maps on the Reaction–Diffusion benchmark for channel 0 across representative rollout steps from $t = 10$ to $t = 90$. Each pair of rows shows one method's predicted state and its corresponding pointwise absolute-error map, with the ground truth shown in the first row.

Table 30 reports temporal extrapolation results on the Reaction–Diffusion benchmark, with errors computed only on rollout steps 51–90 beyond the supervised training horizon. This experiment evaluates whether the learned propagator remains reliable when recursively applied outside the time interval directly seen during training, under stronger multivariable coupling than in the Shallow–Water case. LNO fails to converge on this benchmark. Among the baselines that produce valid results, F-FNO is the strongest competitor and achieves the second-best performance on most metrics, but it still has larger global error, weaker boundary consistency, and stronger spectral distortion than our method. UNO obtains the second-best relative L_2, yet its much larger $E_{\max}$, BRMS, and spectral errors indicate unstable local behavior, weaker boundary consistency, and severe error accumulation during extrapolation. FNO performs moderately but remains clearly worse than F-FNO and our

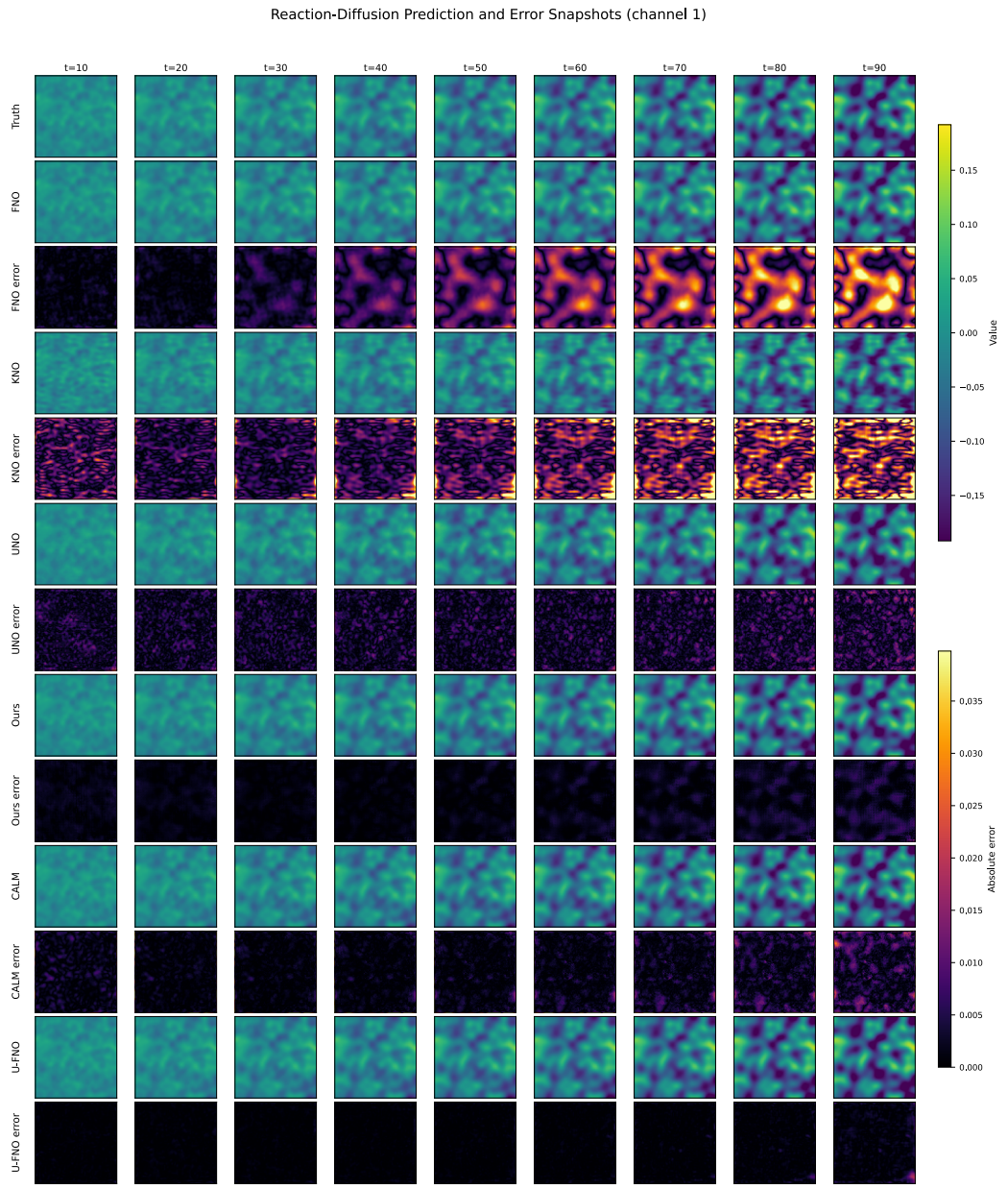


Figure 8: Ground-truth states and absolute-error maps on the Reaction–Diffusion benchmark for channel 1 across representative rollout steps from $t = 10$ to $t = 90$. Each pair of rows shows one method's predicted state and its corresponding pointwise absolute-error map, with the ground truth shown in the first row.

method across all metrics, while U-FNO, KNO, and CALM exhibit substantially larger physical-space and spectral errors. Our method achieves the best MSE, relative L_2, $E_{\max}$, BRMS, f_{low}, f_{mid}, f_{high}, and f_{mse}, showing consistent improvements in both physical-space and spectral-domain measures. The large reduction in BRMS suggests better boundary consistency during extrapolation, while the lower mid- and high-frequency errors indicate better preservation of fine-scale reaction–diffusion patterns during beyond-horizon rollout. Overall, these results show that our method delivers the most stable and accurate temporal extrapolation performance on this coupled nonlinear benchmark.

To distinguish local prediction accuracy from error accumulation under repeated propagation, we further evaluate SSP, F-FNO, KNO, and CALM on extended Shallow–Water (SW) and Reaction–

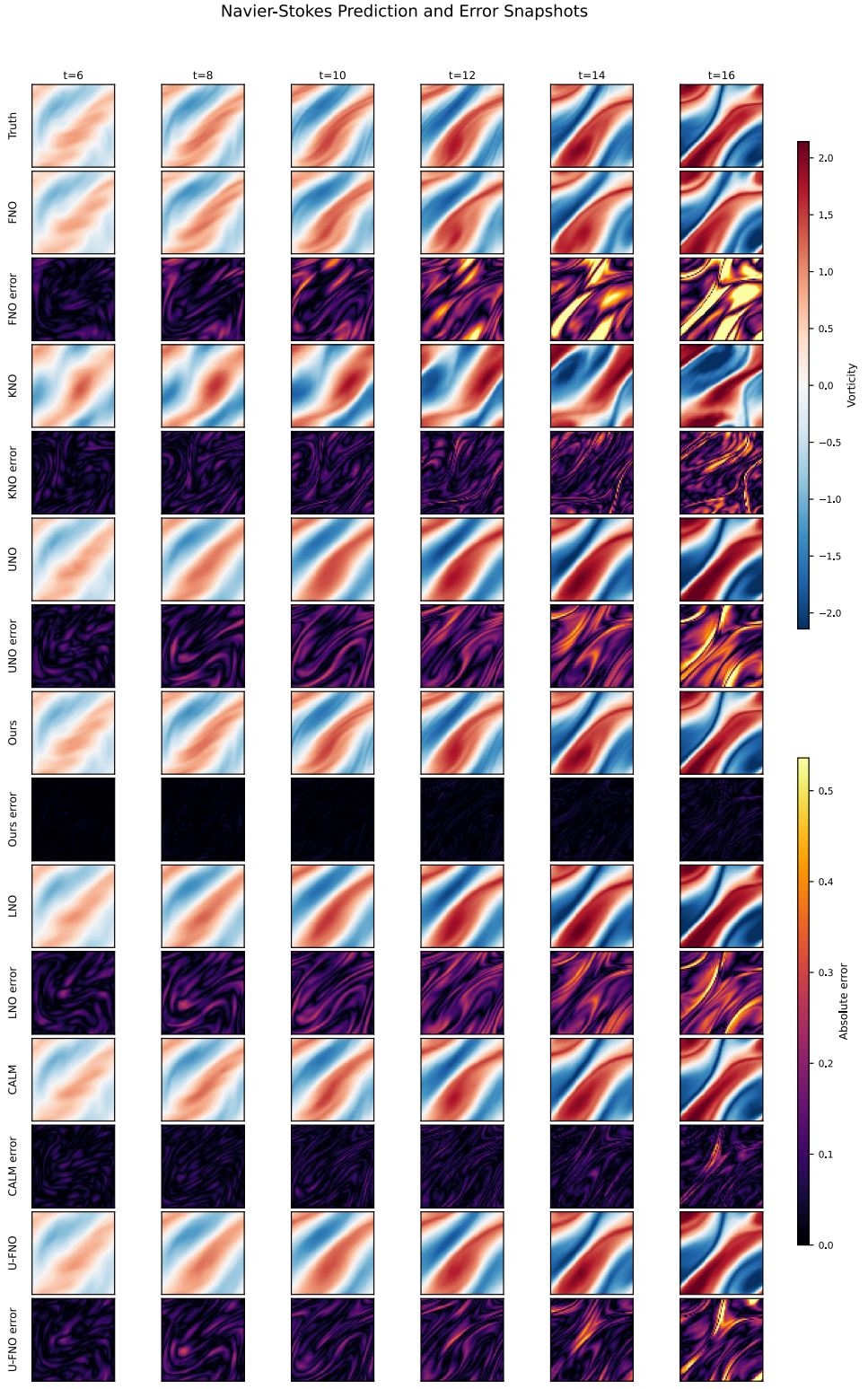


Figure 9: Ground-truth states and absolute-error maps on the Navier-Stokes benchmark across representative rollout steps at $t = 6, 8, 10, 12, 14, 16$. Each pair of rows shows one method's predicted state and its corresponding pointwise absolute-error map, with the ground truth shown in the first row.

Table 26: Full-horizon errors over the extended 70-step Navier–Stokes rollout at $\nu = 10^{-5}$. All methods are retrained and evaluated over the same 70 future frames. F-FNO uses only the final frame of the 10-frame history used by the other methods. Best results are bold and second-best results are underlined.

Method	L_2	$E_{\max}$	BRMS	f_{low}	f_{mid}	f_{high}	f_{mse}
F-FNO	2.23E-1	1.05E+0	2.14E-1	4.38E-2	2.95E-2	1.06E-2	1.95E-2
KNO	1.79E-1	9.49E-1	1.82E-1	9.62E-2	5.18E-2	3.02E-2	4.39E-2
CALM	<u>1.56E-1</u>	<u>7.04E-1</u>	<u>1.59E-1</u>	<u>1.92E-2</u>	<u>2.22E-2</u>	<u>9.23E-3</u>	<u>1.37E-2</u>
SSP	**1.10E-1**	**4.11E-1**	**1.01E-1**	**1.03E-2**	**1.26E-2**	**4.42E-3**	**7.20E-3**

Table 27: Interval-wise prediction errors over the extended 70-step Navier–Stokes autoregressive rollout. Each metric is computed over the corresponding interval using the same definition as in the full-horizon evaluation. Best results are in **bold**, and second-best results are <u>underlined</u>.

Interval	Metric	F-FNO	KNO	CALM	SSP
1–30	L_2	1.14E-1	1.33E-1	<u>9.11E-2</u>	**8.54E-2**
	$E_{\max}$	3.32E-1	3.39E-1	<u>2.25E-1</u>	**1.45E-1**
	f_{low}	1.71E-2	5.07E-2	<u>6.68E-3</u>	**2.34E-3**
31–50	L_2	1.69E-1	1.52E-1	<u>9.36E-2</u>	**9.11E-2**
	$E_{\max}$	9.05E-1	6.84E-1	<u>5.55E-1</u>	**4.67E-1**
	f_{low}	3.45E-2	7.88E-2	<u>1.72E-2</u>	**7.85E-3**
51–70	L_2	3.81E-1	2.08E-1	<u>1.84E-1</u>	**1.42E-1**
	$E_{\max}$	2.26E+0	2.13E+0	<u>1.57E+0</u>	**7.54E-1**
	f_{low}	9.30E-2	1.82E-1	<u>4.01E-2</u>	**2.46E-2**

Diffusion (RD) test trajectories. For each system, we numerically generate 100 additional trajectories with unseen initial conditions, each containing 200 temporal snapshots at a spatial resolution of 128×128, following the original physical and numerical settings. All methods retain their original checkpoints for the 50-step supervised prediction task, without retraining or adaptation. SSP, KNO, and CALM use the first 10 frames, whereas F-FNO uses only the tenth frame. All methods predict the same subsequent 190 steps, including 140 extrapolation steps beyond the supervised horizon. Numerical differences from the standard 40-step evaluation in Section 5.2 reflect the use of independently simulated test trajectories with initial conditions sampled under different random seeds.

We first evaluate teacher-forced one-step accuracy by providing the exact ground-truth context required by each model and predicting only the next physical state. Table 31 reports the mean errors over extrapolation steps 1–140. F-FNO achieves the lowest one-step relative L_2 errors on both systems, indicating stronger local prediction accuracy.

We then evaluate free-running autoregressive prediction, where model outputs are recursively fed back without ground-truth correction. Table 32 reports prediction errors evaluated separately over three extrapolation intervals, while Figure 4 presents the corresponding rollout metrics over consecutive 10-step intervals. All interval-wise metrics follow the definitions in Appendix C.2. On SW, SSP achieves lower errors as the rollout extends, reducing the final-interval relative L_2 by 39.4% compared with CALM, the strongest baseline in that interval. On RD, SSP achieves the lowest errors across all three intervals and reduces the final-interval relative L_2 by 37.9% compared with F-FNO. These results provide empirical evidence that SSP's long-horizon advantage extends beyond one-step prediction accuracy and is associated with reduced error accumulation under recursive propagation. These trajectories use the original PDEBench initial-condition samplers and numerical settings in Appendix B. The extended time grids and removal of the final simulated frame are specified in Appendix C.1; the saved-frame intervals and solver integration settings are unchanged.

Table 28: Temporal extrapolation results on the Shallow Water benchmark. Errors are computed only over rollout steps 51–90, which lie beyond the supervised training horizon. Best results are in **bold**, and second-best results are underlined.

Method	**MSE**	**L2**	$E_{\max}$	**BRMS**	f_{low}	f_{mid}	f_{high}	f_{mse}
FNO	1.31E+05	1.83E+02	3.29E+02	4.86E+01	1.16E+01	7.30E+00	6.63E-01	2.17E+00
U-FNO	1.25E+00	1.28E+00	1.02E+01	1.37E-02	2.40E-01	8.51E-02	8.07E-03	3.22E-02
F-FNO	**2.16E-03**	**4.36E-02**	4.69E-01	6.51E-03	**7.71E-03**	5.04E-03	1.68E-03	2.48E-03
UNO	2.29E+23	3.56E+11	4.21E+11	1.47E+11	6.01E+10	9.50E+09	1.26E+09	5.97E+09
KNO	7.35E-03	8.19E-02	9.96E-01	1.90E-02	4.71E-02	1.94E-02	8.99E-03	1.27E-02
LNO	9.96E+02	1.71E+00	2.96E+02	1.13E+01	2.43E+00	1.05E+00	1.18E+00	1.24E+00
CALM	4.91E-03	6.41E-02	3.68E-01	2.17E-02	1.56E-02	5.36E-03	7.98E-04	2.29E-03
Ours	2.21E-03	4.49E-02	**2.55E-01**	**5.46E-04**	1.24E-02	**2.02E-03**	**2.59E-04**	**1.24E-03**

Table 29: Temporal extrapolation on the fully coupled three-variable Shallow–Water system. All methods are retrained with three physical channels and evaluated on prediction steps 51–90. Best results for each variable are in **bold**, and second-best results are underlined.

Variable	Method	L_2	$E_{\max}$	BRMS	f_{low}	f_{mid}	f_{high}	f_{mse}
h	F-FNO	8.31E-2	3.20E+0	2.93E-2	1.96E-2	1.60E-2	2.73E-3	5.44E-3
	KNO	9.65E-2	3.90E-1	1.40E-2	3.67E-2	1.32E-2	7.69E-3	1.02E-2
	CALM	5.59E-2	3.64E-1	8.27E-3	1.14E-2	4.80E-3	9.17E-4	2.06E-3
	SSP	**4.17E-2**	**3.52E-1**	**2.13E-3**	**1.03E-2**	**2.17E-3**	**2.70E-4**	**1.13E-3**
hu	F-FNO	2.58E-1	1.78E+0	1.63E-2	1.07E-2	1.27E-2	2.01E-3	3.89E-3
	KNO	2.35E-1	2.39E-1	2.93E-3	2.58E-2	9.08E-3	5.20E-3	6.97E-3
	CALM	4.91E-1	2.28E-1	5.60E-3	7.02E-3	3.52E-3	6.27E-4	1.39E-3
	SSP	**1.55E-1**	**1.74E-1**	**9.77E-4**	**5.00E-3**	**1.88E-3**	**2.14E-4**	**7.21E-4**
hv	F-FNO	2.43E-1	1.71E+0	2.35E-2	1.05E-2	1.28E-2	1.97E-3	3.86E-3
	KNO	2.07E-1	2.16E-1	3.63E-3	1.46E-2	1.07E-2	6.12E-3	7.22E-3
	CALM	4.50E-1	2.03E-1	7.11E-3	6.83E-3	3.34E-3	6.53E-4	1.38E-3
	SSP	**1.53E-1**	**1.80E-1**	**1.60E-3**	**4.92E-3**	**2.01E-3**	**1.98E-4**	**7.20E-4**

Table 30: Temporal extrapolation results on the Reaction–Diffusion benchmark, with errors computed jointly over channels u and v. Errors are computed only over rollout steps 51–90, which lie beyond the supervised training horizon. LNO failed to converge on this dataset, plausibly due to multivariable coupling. Best results are in **bold**, and second-best results are underlined.

Method	**MSE**	**L2**	$E_{\max}$	**BRMS**	f_{low}	f_{mid}	f_{high}	f_{mse}
FNO	4.38E-04	1.79E-01	1.36E-01	3.32E-02	4.82E-03	1.85E-03	3.83E-04	8.44E-04
U-FNO	2.71E-03	4.53E-01	1.52E-01	5.87E-02	1.34E-02	4.40E-03	4.74E-04	1.77E-03
F-FNO	2.58E-04	1.39E-01	7.15E-02	2.03E-02	3.79E-03	1.62E-03	2.04E-04	6.05E-04
UNO	1.10E-02	1.33E-01	6.27E-01	1.55E-01	7.08E-03	1.51E-02	2.98E-03	4.74E-03
KNO	8.20E-04	2.47E-01	2.66E-01	7.34E-02	1.18E-02	7.16E-03	4.53E-03	5.31E-03
LNO	–	–	–	–	–	–	–	–
CALM	1.03E-03	2.81E-01	3.00E-01	9.31E-02	4.17E-03	2.98E-03	1.16E-03	1.58E-03
Ours	**1.21E-04**	**9.15E-02**	**4.07E-02**	**9.63E-03**	**2.26E-03**	**6.21E-04**	**9.00E-05**	**2.92E-04**

Table 31: Teacher-forced one-step prediction errors on the extended Shallow–Water and Reaction–Diffusion test sets, averaged over extrapolation steps 1–140. Best results are in **bold**, and second-best results are underlined.

System	Method	L_2	f_{low}	f_{high}
SW	F-FNO	**2.17E-2**	**3.29E-3**	3.67E-4
	KNO	4.17E-2	2.02E-2	**2.49E-4**
	CALM	1.01E-1	7.89E-3	1.33E-3
	SSP	3.21E-2	5.46E-3	2.70E-4
RD	F-FNO	**2.59E-3**	**5.77E-5**	**3.51E-5**
	KNO	7.73E-2	9.60E-4	1.19E-3
	CALM	2.01E-1	3.10E-3	2.80E-3
	SSP	3.22E-2	5.69E-4	1.21E-4

Table 32: Interval-wise prediction errors over the extended 140-step autoregressive extrapolation on Shallow–Water and Reaction–Diffusion. Best results are in **bold**, and second-best results are underlined.

System	Interval	Metric	F-FNO	KNO	CALM	SSP
SW	1–40	L_2	**4.29E-2**	8.29E-2	6.29E-2	4.42E-2
		f_{low}	**7.84E-3**	4.65E-2	1.51E-2	1.21E-2
		f_{high}	1.74E-3	8.52E-3	8.05E-4	**2.51E-4**
	41–90	L_2	1.41E-1	1.50E-1	1.43E-1	**8.43E-2**
		f_{low}	3.86E-2	8.14E-2	4.04E-2	**2.83E-2**
		f_{high}	2.39E-3	2.53E-2	1.39E-3	**5.66E-4**
	91–140	L_2	2.45E-1	2.07E-1	1.80E-1	**1.09E-1**
		f_{low}	7.42E-2	9.21E-2	5.76E-2	**3.68E-2**
		f_{high}	2.80E-3	3.91E-2	1.70E-3	**6.90E-4**
RD	1–40	L_2	1.14E-1	2.43E-1	2.38E-1	**8.71E-2**
		f_{low}	3.23E-3	9.96E-3	4.35E-3	**2.37E-3**
		f_{high}	2.12E-4	4.31E-3	1.36E-3	**8.66E-5**
	41–90	L_2	1.57E-1	4.27E-1	3.17E-1	**9.37E-2**
		f_{low}	1.01E-2	1.85E-2	1.00E-2	**5.73E-3**
		f_{high}	6.40E-4	3.30E-3	3.18E-3	**1.99E-4**
	91–140	L_2	1.95E-1	7.50E-1	4.39E-1	**1.21E-1**
		f_{low}	1.21E-2	3.60E-2	1.66E-2	**9.67E-3**
		f_{high}	8.97E-4	5.84E-3	3.79E-3	**3.72E-4**

D.3 Ablation details

Table 33 reports the ablation results on the Shallow–Water benchmark. The full model achieves the best performance across all physical-space and spectral metrics, including relative L_2, $E_{\max}$, BRMS, f_{low}, f_{mid}, f_{high}, and f_{mse}. This indicates that the proposed structured design provides the most consistent rollout behavior for long-horizon wave propagation. Removing the frequency-conditioned linear backbone K_θ mainly increases the low-frequency error, confirming its role in preserving the dominant modal structure of the wave field. Removing the nonlinear closure G_ψ leads to a more pronounced degradation in relative L_2, $E_{\max}$, and mid-frequency error, suggesting that nonlinear spectral correction is important for compensating residual modal interactions during recursive prediction. The variant without the projection and reconstruction modules P, R also performs worse than the full model, especially in BRMS and spectral errors, which supports the benefit of separating the rich spatial representation from the compact propagation state. The $T{\to}C$ encoder variant shows the largest degradation, indicating that a direct temporal-to-channel encoding strategy is insufficient for constructing propagation-compatible latent states in this benchmark. Among the convolutional replacements, the Conv Decoder remains the most competitive and obtains second-best results on most metrics, whereas replacing the encoder or both encoder and decoder causes larger error increases. These results suggest that the spectral-factorized encoder is particularly important for forming a stable latent propagation space, while the decoder admits slightly more architectural flexibility. Overall, the ablation study verifies that the linear modal backbone, nonlinear closure, compact projection–reconstruction pathway, and spectral analysis module provide complementary contributions to stable Shallow–Water rollout.

Table 34 reports the ablation results on the Reaction–Diffusion benchmark. The full model achieves the best performance across all physical-space and spectral metrics, including relative L_2, $E_{\max}$, BRMS, f_{low}, f_{mid}, f_{high}, and f_{mse}. This result indicates that the complete structured propagator is also effective for coupled two-channel reaction–diffusion dynamics, where both local pattern formation and spectral consistency need to be maintained during recursive rollout. Removing the frequency-conditioned linear backbone K_θ leads to clear degradation in both trajectory errors and frequency-domain errors, confirming that the modal linear component remains important for organizing the dominant evolution of the system. Removing the nonlinear closure G_ψ gives the second-best results on most metrics, but still performs consistently worse than the full model, suggesting that the closure provides useful correction for nonlinear modal interactions rather than being redundant. The variant without the projection and reconstruction modules P, R shows a substantial increase in BRMS and spectral errors, which demonstrates the importance of separating the rich spatial representation from the compact propagation state. The $T{\to}C$ encoder variant suffers the largest degradation in relative L_2, $E_{\max}$, BRMS, and low-frequency error, indicating that directly folding temporal information into channels is insufficient for constructing propagation-compatible latent states on this benchmark. The encoder and decoder ablations also increase the errors, with the joint removal of both modules producing a larger performance drop than removing either one alone. Overall, these results show that stable Reaction–Diffusion rollout benefits from the combined use of the linear modal backbone, nonlinear spectral closure, compact projection–reconstruction pathway, and spectral encoder–decoder design.

Table 37 reports the ablation results on the Navier–Stokes benchmark. The full model achieves the best performance across all physical-space and spectral metrics, including relative L_2, $E_{\max}$, BRMS, f_{low}, f_{mid}, f_{high}, and f_{mse}. This demonstrates that the complete analysis–propagation–synthesis design is effective for the more complex Navier–Stokes dynamics, where coherent vortex structures and multi-scale spectral components must be preserved during rollout. Removing the frequency-conditioned linear backbone K_θ consistently increases both trajectory errors and spectral errors, confirming its role in modeling the dominant modal evolution of retained spectral states. Removing the nonlinear closure G_ψ also degrades performance, although it remains the second-best variant on several metrics, suggesting that the closure provides useful but controlled correction for unresolved nonlinear interactions. The variant without the projection and reconstruction modules P, R performs worse than the full model on every metric, which supports the need to separate the rich spatial representation from the compact propagation state before recurrent evolution. The $T{\to}C$ encoder variant causes a clear increase in both physical-space and frequency-domain errors, indicating that directly folding temporal information into channel dimensions is insufficient for constructing stable propagation-compatible latent states. Compared with the encoder-only replacement, replacing the decoder by convolutional modules leads to larger degradation on most metrics, and replacing both

encoder and decoder produces the worst overall performance. This suggests that, for Navier–Stokes rollout, both spectral analysis and spectral-aware reconstruction are important, while the decoder is especially critical for recovering coherent flow structures from the propagated latent state. Overall, the ablation results verify that the spectral encoder–decoder design, compact projection–reconstruction pathway, frequency-conditioned linear backbone, and nonlinear spectral closure make complementary contributions to stable Navier–Stokes forecasting.

We independently remove the physical rollout loss $\mathcal{L}_{\mathrm{phy}}$, latent rollout loss $\mathcal{L}_{\mathrm{lat}}$, and orthogonality loss $\mathcal{L}_{\mathrm{orth}}$ to assess their contributions. Table 35 reports the results on long-horizon forecasting across NS, SW, and RD, while Table 36 presents temporal extrapolation results on SW and RD. Removing $\mathcal{L}_{\mathrm{phy}}$ leads to the largest degradation, while $\mathcal{L}_{\mathrm{lat}}$ improves extrapolation accuracy and $\mathcal{L}_{\mathrm{orth}}$ provides smaller but consistent gains. The effect of normality regularization is analyzed separately in Appendix A.6.3.

Table 33: Ablation study on Shallow–Water. Best results are in **bold**, and second-best results are underlined.

Method	L_2	$E_{\max}$	**BRMS**	f_{low}	f_{mid}	f_{high}	f_{mse}
Full	**6.56E-4**	**9.72E-3**	**1.91E-4**	**2.53E-5**	**3.93E-5**	**3.87E-5**	**3.79E-5**
w/o K_θ	9.09E-4	1.30E-2	2.36E-4	8.50E-5	7.30E-5	4.90E-5	5.43E-5
w/o G_ψ	1.78E-3	2.16E-2	3.04E-4	7.70E-5	1.92E-4	7.90E-5	9.30E-5
w/o $\mathcal{P}, \mathcal{R}$	1.05E-3	1.38E-2	3.48E-4	4.90E-5	8.30E-5	5.80E-5	6.06E-5
$T \to C$ Enc	2.06E-3	3.36E-2	5.14E-4	1.01E-4	2.64E-4	9.20E-5	1.14E-4
Conv Encoder	7.96E-4	1.15E-2	2.27E-4	3.30E-5	5.60E-5	4.50E-5	4.56E-5
Conv Decoder	7.23E-4	1.03E-2	2.24E-4	2.70E-5	4.00E-5	4.30E-5	4.16E-5
Conv Enc-Dec	1.08E-3	1.76E-2	3.74E-4	5.50E-5	7.80E-5	6.10E-5	6.28E-5

Table 34: Ablation study on Reaction–Diffusion. Best results are in **bold**, and second-best results are underlined.

Method	L_2	$E_{\max}$	**BRMS**	f_{low}	f_{mid}	f_{high}	f_{mse}
Full	**7.46E-3**	**6.71E-3**	**9.61E-4**	**1.48E-4**	**5.17E-5**	**1.33E-5**	**2.65E-5**
w/o K_θ	1.43E-2	1.15E-2	2.40E-3	4.80E-4	2.10E-4	8.50E-5	1.25E-4
w/o G_ψ	1.12E-2	8.12E-3	1.65E-3	2.90E-4	1.10E-4	3.50E-5	6.03E-5
w/o $\mathcal{P}, \mathcal{R}$	2.41E-2	2.20E-2	5.80E-3	1.40E-3	8.80E-4	3.80E-4	5.06E-4
$T \to C$ Enc	6.41E-2	3.99E-2	8.77E-3	2.24E-3	4.67E-4	7.70E-5	2.61E-4
Conv Encoder	1.38E-2	1.32E-2	2.30E-3	3.50E-4	2.10E-4	9.00E-5	1.21E-4
Conv Decoder	1.26E-2	1.15E-2	1.95E-3	2.90E-4	1.60E-4	6.50E-5	9.09E-5
Conv Enc-Dec	1.95E-2	1.68E-2	3.90E-3	8.50E-4	4.50E-4	1.90E-4	2.64E-4

Table 35: Loss-term ablation on long-horizon forecasting across three PDE benchmarks. Each variant removes one loss term from the full objective. Best results are in **bold**, and second-best results are underlined.

Method	Navier–Stokes			Shallow–Water			Reaction–Diffusion		
	L_2	$E_{\max}$	f_{low}	L_2	$E_{\max}$	f_{low}	L_2	$E_{\max}$	f_{low}
Full	**5.50E-2**	**5.38E-1**	**5.49E-3**	**6.56E-4**	**9.72E-3**	**2.53E-5**	**7.46E-3**	**6.71E-3**	**1.48E-4**
w/o $\mathcal{L}_{\mathrm{phy}}$	5.95E-1	3.48E+0	2.04E-1	6.01E-2	3.69E-1	1.52E-2	2.35E-1	1.05E-1	2.21E-2
w/o $\mathcal{L}_{\mathrm{lat}}$	6.04E-2	5.86E-1	6.15E-3	8.70E-4	1.20E-2	3.76E-5	1.51E-2	1.37E-2	5.70E-4
w/o $\mathcal{L}_{\mathrm{orth}}$	5.88E-2	5.46E-1	5.74E-3	7.62E-4	1.10E-2	2.99E-5	7.93E-3	6.90E-3	1.64E-4

D.4 Statistical Stability Across Random Seeds

To assess the sensitivity of SSP to random initialization and data-order randomness, we repeat the main experiments with three independent random seeds. For each run, the dataset split, model configuration, training protocol, and evaluation procedure are kept fixed, while only the random seed

Table 36: Loss-term ablation on temporal extrapolation for Shallow–Water and Reaction–Diffusion. Each variant removes one loss term from the full objective. Best results are in **bold**, and second-best results are underlined.

Method	Shallow–Water			Reaction–Diffusion		
	L_2	$E_{\max}$	f_{low}	L_2	$E_{\max}$	f_{low}
Full	**4.49E-2**	**2.55E-1**	**1.24E-2**	**9.15E-2**	**4.07E-2**	**2.26E-3**
w/o $\mathcal{L}_{\text{phy}}$	8.11E-1	9.68E+0	9.04E-1	9.79E-1	6.37E-1	2.59E-2
w/o $\mathcal{L}_{\text{lat}}$	5.86E-2	3.21E-1	2.48E-2	1.71E-1	8.55E-2	6.78E-3
w/o $\mathcal{L}_{\text{orth}}$	5.61E-2	3.06E-1	1.67E-2	1.08E-1	4.68E-2	2.83E-3

Table 37: Ablation study on Navier–Stokes. Best results are in **bold**, and second-best results are underlined.

Method	L_2	$E_{\max}$	**BRMS**	f_{low}	f_{mid}	f_{high}	f_{mse}
Full	**5.50E-2**	**5.38E-1**	**5.74E-2**	**5.49E-3**	**7.29E-3**	**5.17E-3**	**5.74E-3**
w/o K_θ	6.20E-2	6.48E-1	6.75E-2	6.41E-3	8.45E-3	5.98E-3	6.65E-3
w/o G_ψ	6.04E-2	5.93E-1	6.66E-2	5.92E-3	8.24E-3	5.93E-3	6.51E-3
w/o $\mathcal{P}, \mathcal{R}$	6.07E-2	6.21E-1	6.58E-2	6.11E-3	8.36E-3	5.77E-3	6.46E-3
$T \to C$ Enc	8.35E-2	9.43E-1	9.63E-2	8.18E-3	1.24E-2	8.20E-3	9.24E-3
Conv Encoder	7.11E-2	7.35E-1	7.62E-2	7.81E-3	1.03E-2	6.49E-3	7.60E-3
Conv Decoder	8.73E-2	8.98E-1	1.00E-1	1.06E-2	1.34E-2	7.34E-3	9.26E-3
Conv Enc-Dec	1.53E-1	1.48E+0	1.72E-1	2.57E-2	2.50E-2	1.04E-2	1.60E-2

is changed. We report the mean and standard deviation across the three runs. All reported quantities are error metrics, and the downward arrow indicates that lower values are better.

Table 38: Statistical stability of SSP on long-horizon forecasting tasks across three independent random seeds in physical space. We report mean $\pm$ standard deviation. The downward arrow indicates that lower values are better.

Dataset	MSE $\downarrow$	$L_2 \downarrow$	$E_{\max} \downarrow$	BRMS $\downarrow$
Navier–Stokes	$6.74\text{E}{-3} \pm 4.22\text{E}{-4}$	$5.50\text{E}{-2} \pm 2.30\text{E}{-3}$	$5.38\text{E}{-1} \pm 1.14\text{E}{-2}$	$5.74\text{E}{-2} \pm 2.78\text{E}{-3}$
Shallow–Water	$4.78\text{E}{-7} \pm 7.68\text{E}{-8}$	$6.56\text{E}{-4} \pm 5.47\text{E}{-5}$	$9.72\text{E}{-3} \pm 1.74\text{E}{-3}$	$1.91\text{E}{-4} \pm 1.38\text{E}{-5}$
Reaction–Diffusion	$5.26\text{E}{-7} \pm 2.40\text{E}{-7}$	$7.46\text{E}{-3} \pm 1.66\text{E}{-3}$	$6.71\text{E}{-3} \pm 6.08\text{E}{-4}$	$9.61\text{E}{-4} \pm 1.22\text{E}{-4}$

Table 39: Statistical stability of SSP on long-horizon forecasting tasks across three independent random seeds in the spectral domain. We report mean $\pm$ standard deviation. The downward arrow indicates that lower values are better.

Dataset	$f_{\text{low}} \downarrow$	$f_{\text{mid}} \downarrow$	$f_{\text{high}} \downarrow$	$f_{\text{mse}} \downarrow$
Navier–Stokes	$5.49\text{E}{-3} \pm 2.85\text{E}{-4}$	$7.29\text{E}{-3} \pm 4.02\text{E}{-4}$	$5.17\text{E}{-3} \pm 2.15\text{E}{-4}$	$5.74\text{E}{-3} \pm 2.67\text{E}{-4}$
Shallow–Water	$2.53\text{E}{-5} \pm 2.08\text{E}{-6}$	$3.93\text{E}{-5} \pm 5.03\text{E}{-6}$	$3.87\text{E}{-5} \pm 3.51\text{E}{-6}$	$3.79\text{E}{-5} \pm 3.51\text{E}{-6}$
Reaction–Diffusion	$1.48\text{E}{-4} \pm 2.96\text{E}{-5}$	$5.17\text{E}{-5} \pm 9.61\text{E}{-6}$	$1.33\text{E}{-5} \pm 1.16\text{E}{-6}$	$2.65\text{E}{-5} \pm 3.51\text{E}{-6}$

Tables 38 and 39 report the statistical stability on the long-horizon forecasting tasks. SSP shows low run-to-run variation on Navier–Stokes and Shallow–Water. For example, the relative L_2 errors are $5.50\text{E}{-2} \pm 2.30\text{E}{-3}$ on Navier–Stokes and $6.56\text{E}{-4} \pm 5.47\text{E}{-5}$ on Shallow–Water. Reaction–Diffusion exhibits relatively larger variation, with relative L_2 error $7.46\text{E}{-3} \pm 1.66\text{E}{-3}$, mainly due to one higher-error seed, but the absolute standard deviation remains small and no seed-level failure is observed. The spectral-domain metrics show the same trend, indicating that the observed improvements are not tied to a particular random initialization.

Tables 40 and 41 further report the results on temporal extrapolation beyond the supervised horizon. As expected, extrapolation produces larger seed-to-seed variability than supervised long-horizon forecasting because rollout errors are evaluated outside the training horizon. Nevertheless, SSP

remains statistically stable at the level of the main trajectory metrics, with relative L_2 error $4.49\text{E}{-2}\pm 6.67\text{E}{-3}$ on Shallow–Water and $9.15\text{E}{-2} \pm 1.04\text{E}{-2}$ on Reaction–Diffusion. The frequency metrics are more sensitive under extrapolation, especially on Reaction–Diffusion, where the spectral errors vary more across seeds. This suggests that extrapolated spectral details are more affected by optimization randomness than global physical-space trajectory errors, while the overall rollout accuracy remains consistent across independent runs.

Table 40: Statistical stability of SSP on extrapolation tasks across three independent random seeds in physical space. We report mean $\pm$ standard deviation. The downward arrow indicates that lower values are better.

Dataset	MSE ↓	L_2 ↓	$E_{\max}$ ↓	BRMS ↓
Shallow–Water	$2.21\text{E}{-3} \pm 6.79\text{E}{-4}$	$4.49\text{E}{-2} \pm 6.67\text{E}{-3}$	$2.55\text{E}{-1} \pm 2.15\text{E}{-2}$	$5.46\text{E}{-4} \pm 2.13\text{E}{-4}$
Reaction–Diffusion	$1.21\text{E}{-4} \pm 2.54\text{E}{-5}$	$9.15\text{E}{-2} \pm 1.04\text{E}{-2}$	$4.07\text{E}{-2} \pm 9.83\text{E}{-3}$	$9.63\text{E}{-3} \pm 1.33\text{E}{-3}$

Table 41: Statistical stability of SSP on extrapolation tasks across three independent random seeds in the spectral domain. We report mean $\pm$ standard deviation. The downward arrow indicates that lower values are better.

Dataset	f_{low} ↓	f_{mid} ↓	f_{high} ↓	f_{mse} ↓
Shallow–Water	$1.24\text{E}{-2} \pm 1.63\text{E}{-3}$	$2.02\text{E}{-3} \pm 3.63\text{E}{-4}$	$2.59\text{E}{-4} \pm 7.62\text{E}{-5}$	$1.24\text{E}{-3} \pm 2.09\text{E}{-4}$
Reaction–Diffusion	$2.26\text{E}{-3} \pm 4.17\text{E}{-4}$	$6.21\text{E}{-4} \pm 1.12\text{E}{-4}$	$9.00\text{E}{-5} \pm 1.57\text{E}{-5}$	$2.92\text{E}{-4} \pm 1.61\text{E}{-5}$

D.5 Higher-resolution Shallow–Water Evaluation

To evaluate SSP at higher spatial resolution, we numerically generate 1,000 Shallow–Water (SW) trajectories at 256×256 resolution, each containing 100 temporal snapshots, following the physical and numerical settings of the original benchmark. All compared methods are retrained at this resolution with a 50-step supervised prediction horizon. Methods other than F-FNO use 10 history frames, whereas F-FNO uses only the final frame. At inference, all methods autoregressively predict the same subsequent 90 frames, with extrapolation errors evaluated over steps 51–90.

Table 42 reports the results. SSP achieves the lowest errors across all seven metrics, including a relative L_2 error of 5.05×10^{-2} compared with 5.16×10^{-2} for CALM, the strongest baseline. These results demonstrate the effectiveness of SSP under the evaluated higher-resolution configuration. The higher-resolution data use the SW solver and temporal sampling in Appendix B, with cell spacing reduced from $5/128$ to $5/256$; the numerical scheme and CFL-based time-step control are unchanged.

Table 42: Temporal extrapolation on the Shallow–Water benchmark at 256×256 resolution, evaluated over prediction steps 51–90. All methods are retrained at the target resolution. Best results are in **bold**, and second-best results are underlined.

Method	L_2	$E_{\max}$	BRMS	f_{low}	f_{mid}	f_{high}	f_{mse}
F-FNO	7.24E-2	3.61E-1	3.50E-4	1.94E-2	6.75E-3	6.49E-4	1.62E-3
KNO	1.41E-1	1.54E+0	2.30E-1	2.46E-2	2.97E-2	1.61E-2	1.72E-2
CALM	5.16E-2	6.21E-1	1.47E-2	1.56E-2	5.54E-3	7.56E-4	1.52E-3
SSP	**5.05E-2**	**2.87E-1**	**2.92E-4**	**1.26E-2**	**3.60E-3**	**4.14E-4**	**9.93E-4**

D.6 Robustness to Distribution Shifts

We further assess SSP under distribution shifts on the two-dimensional Shallow–Water radial dam-break problem over the domain $[-2.5, 2.5]^2$. In the original setting, the dam is centered at $(0, 0)$, with radius $r \sim U(0.3, 0.7)$, gravitational acceleration $g = 1.0$, and initial inner water depth $h_{\text{in}} = 2.0$. All shifted test sets retain the baseline SW solver, boundary conditions, 128×128 grid, and $\Delta t_{\text{save}} = 0.01$ in Appendix B. Only the stated geometry or physical parameter is varied; frame retention follows the extended SW protocol in Appendix C.1.

Table 43: Autoregressive extrapolation under unseen Shallow–Water initial geometries. Off-center shifts the dam center to $(0.75, 0.75)$; enlarged radius samples $r \sim U(0.8, 1.0)$. Best results are in **bold**, and second-best results are underlined.

Shift	Interval	Metric	F-FNO	KNO	CALM	SSP
Off-center	1–40	L_2	2.75E-1	9.25E-2	1.28E-1	**7.85E-2**
		f_{low}	8.14E-2	2.98E-2	3.41E-2	**2.84E-2**
		f_{high}	1.54E-3	1.15E-3	1.11E-3	**9.13E-4**
	41–90	L_2	3.41E-1	1.58E-1	1.36E-1	**8.31E-2**
		f_{low}	1.02E-1	3.48E-2	3.32E-2	**3.09E-2**
		f_{high}	1.97E-3	2.74E-3	1.67E-3	**1.13E-3**
	91–140	L_2	3.62E-1	2.26E-1	1.40E-1	**8.40E-2**
		f_{low}	1.06E-1	4.08E-2	3.70E-2	**3.20E-2**
		f_{high}	2.02E-3	4.26E-3	2.12E-3	**1.06E-3**
Enlarged radius	1–40	L_2	1.39E-1	1.36E-1	1.38E-1	**7.32E-2**
		f_{low}	4.36E-2	4.24E-2	3.42E-2	**1.64E-2**
		f_{high}	1.60E-3	1.47E-3	1.85E-3	**7.64E-4**
	41–90	L_2	2.33E-1	2.64E-1	2.00E-1	**1.24E-1**
		f_{low}	6.24E-2	7.14E-2	4.91E-2	**3.75E-2**
		f_{high}	4.00E-3	2.45E-3	3.11E-3	**1.11E-3**
	91–140	L_2	3.04E-1	4.26E-1	2.47E-1	**1.58E-1**
		f_{low}	9.40E-2	9.28E-2	6.02E-2	**4.54E-2**
		f_{high}	1.90E-3	7.01E-3	3.57E-3	**7.05E-4**

We first consider two unseen initial geometries while keeping the physical parameters fixed: an off-center dam with center $(0.75, 0.75)$ and an enlarged dam radius sampled from $U(0.8, 1.0)$. For each setting, we simulate 100 test trajectories containing 200 snapshots at 128×128 resolution. All models retain their original checkpoints for the 50-step supervised prediction task, using 10-frame histories except for F-FNO, which retains its single-frame input, without retraining, fine-tuning, or test-time adaptation. We evaluate an additional 140 autoregressive steps beyond the supervised horizon.

Table 43 reports the interval-wise results. SSP achieves the lowest relative L_2, low-frequency, and high-frequency errors across all evaluated intervals under both geometric shifts. These results indicate improved robustness to the tested changes in initial geometry over extended autoregressive rollouts.

We further evaluate robustness to changes in gravitational acceleration and initial inner water depth, varying one quantity at a time while keeping the others fixed. Specifically, we consider $g \in \{0.5, 1.5, 2.0\}$ and $h_{\text{in}} \in \{1.5, 3.0\}$, compared with the training settings $g = 1.0$ and $h_{\text{in}} = 2.0$. For each setting, we generate 100 test trajectories with 200 snapshots. All methods retain their original checkpoints without retraining or adaptation and are evaluated over 140 extrapolation steps beyond the supervised horizon.

Table 44 reports the results. SSP achieves the lowest relative L_2 errors in four of the five settings and the lowest high-frequency errors across all settings. Under $h_{\text{in}} = 3.0$, CALM obtains a lower relative L_2 error, while SSP achieves the lowest spectral errors. These results demonstrate favorable robustness to the evaluated changes in physical parameters and initial conditions, with metric-dependent performance differences.

E Limitations and Future Work

SSP is currently designed and evaluated for long-horizon forecasting on fixed discretized grids. In the present implementation, the encoder and decoder operate on regular grid tensors, and the structured propagator applies rFFT2 on the latent spatial grid. Therefore, the current model is not yet a fully resolution-invariant or mesh-free operator, and it does not directly handle irregular meshes, point-cloud observations, or spatially sparse measurements without additional encoding and decoding mechanisms.

Table 44: Autoregressive extrapolation on Shallow–Water under changes in gravitational acceleration and initial inner water depth. Errors are evaluated over the full 140-step extrapolation horizon. Best results are in **bold**, and second-best results are underlined.

Setting	Metric	F-FNO	KNO	CALM	SSP
$g = 0.5$	L_2	3.55E-1	1.60E-1	1.17E-1	**1.02E-1**
	f_{low}	1.16E-1	3.03E-2	**2.48E-2**	2.64E-2
	f_{high}	4.70E-3	2.98E-3	1.66E-3	**9.38E-4**
$g = 1.5$	L_2	3.64E-1	1.77E-1	1.48E-1	**1.26E-1**
	f_{low}	1.19E-1	3.94E-2	3.90E-2	**3.48E-2**
	f_{high}	5.11E-3	3.04E-3	1.66E-3	**9.02E-4**
$g = 2.0$	L_2	3.67E-1	1.86E-1	1.51E-1	**1.39E-1**
	f_{low}	1.20E-1	4.74E-2	4.18E-2	**3.92E-2**
	f_{high}	5.20E-3	3.13E-3	1.64E-3	**8.42E-4**
$h_{\text{in}} = 1.5$	L_2	3.59E-1	1.32E-1	1.05E-1	**8.29E-2**
	f_{low}	1.16E-1	2.81E-2	2.54E-2	**2.05E-2**
	f_{high}	4.73E-3	2.45E-3	1.35E-3	**6.75E-4**
$h_{\text{in}} = 3.0$	L_2	3.95E-1	3.69E-1	**1.21E-1**	1.64E-1
	f_{low}	1.35E-1	8.41E-2	5.64E-2	**4.78E-2**
	f_{high}	6.14E-3	6.10E-3	2.39E-3	**1.65E-3**

This limitation mainly reflects the current implementation of the analysis and synthesis modules rather than a fundamental restriction of the structured propagation principle. A central design choice of SSP is to separate spatial analysis, latent propagation, and physical synthesis through $\mathcal{D} \circ \mathcal{R} \circ \Phi \circ \mathcal{P} \circ \mathcal{E}$. Since $\mathcal{E}$ and $\mathcal{D}$ do not carry the temporal advancement themselves, they can in principle be replaced by alternative geometric interfaces while keeping the middle propagator responsible for temporal evolution. This modularity suggests a path toward extending SSP beyond regular grids.

One future direction is to develop geometry-aware encoders and decoders for irregular inputs. For example, kernel-based encoders, Gaussian-kernel interpolation modules, graph neural operators, or point-cloud networks could be used to embed irregular observations into a shared latent coordinate system. The decoder could then be adapted symmetrically to reconstruct fields on irregular query locations or target meshes. Such a design would preserve the separation between geometric representation learning and latent temporal propagation, while allowing SSP to process non-grid data.

A second direction is to impose stronger mathematical structure on the propagator Φ. Because the temporal evolution is isolated inside the structured spectral propagator, it is possible to introduce additional constraints on the learned dynamics, such as stability constraints, conservation biases, stronger spectral radius constraints, explicit dissipativity conditions, and continuous-time consistency. These extensions may further connect SSP to continuous dynamical systems and improve its reliability for long-horizon prediction. We leave the development of mesh-free geometric interfaces and more strongly constrained continuous latent dynamics to future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contribution as a structured spectral propagation framework for long-horizon PDE forecasting, and the scope is evaluated on Navier–Stokes, Shallow–Water, and Reaction–Diffusion benchmarks. The claims are empirical and are supported by the long-horizon, extrapolation, and ablation experiments reported in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix E discusses the limitations of the proposed method, including its current restriction to regular grids, the lack of direct support for irregular meshes and sparse observations, and the absence of explicit conservation and spectral-radius guarantees.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results in the form of theorems or lemmas. The spectral decomposition and normality discussions are used as methodological motivation and structural intuition rather than as proved guarantees.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the model architecture, training objectives, benchmark definitions, preprocessing procedures, evaluation metrics, and baseline configurations. The appendices additionally document the protocols for extended NS rollouts, full-state and high-resolution SW, one-step and extended autoregressive evaluations, and distribution-shift tests.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The paper provides the model design, experimental configurations, and evaluation protocols, but the implementation and executable reproduction scripts are not yet publicly available. We plan to release the SSP implementation, training and evaluation scripts, and reproducibility instructions through a public GitHub repository following publication.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the training and test settings, including data splits, input lengths, supervised and test rollout horizons, spatial resolutions, normalization, optimizer, learning rate, batch size, epochs, architecture hyperparameters, and evaluation metrics in the main text and appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The appendix reports mean and standard deviation over three independent random seeds for the main long-horizon and extrapolation experiments. The error bars capture run-to-run variability due to initialization and data-order randomness.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C.3 reports the GPU model and memory capacity, per-run training times for the principal SSP experiments, and inference memory and latency measurements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is a computational method for PDE forecasting using scientific simulation benchmarks and does not involve human subjects, private data, surveillance, or other high-risk data sources. We have reviewed the NeurIPS Code of Ethics and believe the work conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The current paper focuses on technical contributions and limitations and does not include a dedicated discussion of positive and negative societal impacts. The method is intended for scientific simulation and forecasting, where potential impacts depend on the downstream application.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models or datasets such as large language models, image generators, scraped personal data, or other assets with substantial misuse risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original datasets and baseline implementations. Appendix B documents the sources and license information for the PDEBench datasets and the FNO repository, together with relevant access information. Third-party raw datasets are not redistributed.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The supplementary experiments use additional numerically simulated trajectories, with their configurations documented in the appendix. No standalone dataset, model checkpoint, or software package is released with the current camera-ready submission. The implementation and reproducibility scripts are planned for public release.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, crowdsourcing, or personally identifiable data, so IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method, experiments, and scientific contributions do not use LLMs as an important, original, or non-standard component. Any writing or formatting assistance, if used, does not affect the methodology, results, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.