Controlling Transient Amplification Improves Long-horizon Rollouts

Adeel Pervez
Institute of Science and Technology Austria
Adeel.Pervez@ist.ac.at

Francesco Locatello
Institute of Science and Technology Austria
Francesco.Locatello@ist.ac.at

Abstract

Autoregressive neural simulators now match classical solvers on short-horizon prediction of physical systems, yet their accuracy degrades rapidly when rolled out over long horizons. In this work, we identify *transient amplification* of perturbations around rollout trajectories as a structural mechanism driving rollout error. Using a linearization analysis we show that when the Jacobians along an autoregressive trajectory are non-normal and non-commuting, the model amplifies errors transiently, resulting in model rollout drift even when the overall system is asymptotically stable. Building on the analysis, we propose *commutativity regularization*: a combination of two penalties designed to reduce the normality defect of individual Jacobians and the commutator norm of Jacobians across steps. The penalties are estimated with Jacobian-vector products and have no inference-time cost. We show a propagator bound that quantifies rollout error under approximate commutativity and normality. We evaluate UNet and FNO variants with commutativity regularization on 1D and 2D spatio-temporal data in synthetic and real settings, showing successful long-horizon rollouts over thousands of steps. Further, we show that the method improves FourCastNet climate forecasts on ERA5 without using any new data. The gain is most pronounced out-of-distribution: trained on trajectories of a few hundred steps, regularized models remain in-distribution for thousands of rollout steps on initial conditions where baselines diverge.

1 Introduction

Neural networks that simulate physical systems one step at a time have become remarkably capable. Models such as FourCastNet [20], GraphCast [12], Pangu-Weather [3], Aurora [4], NeuralGCM [9] and the diffusion-based GenCast [22] now match or surpass operational numerical weather prediction at lead times of five to seven days. Similar progress has been made across fluid dynamics [27], climate emulation [9], and molecular simulation [2] using autoregressive architectures such as Fourier Neural Operators, U-Nets and graph neural networks [13, 25]. On the other hand, the same models, when run iteratively for many steps, degrade far faster than their single-step accuracy would suggest: predictions that closely match the ground truth at one step may diverge or drift within tens of steps on complex systems [14]. Bridging this gap between single-step and long-rollout performance is the central open problem for neural simulators of physical dynamics.

The standard explanation of this phenomenon is *distributional shift*. During training the model sees clean ground-truth states as inputs; at rollout time it receives its own imperfect predictions, which progressively drift away from the training distribution. Recent methods address this by exposing the model to its own errors during training [14, 6] or by applying iterative corrections at inference time [10], requiring multi-step training or expensive inference. These approaches are often effective, but they treat rollout distribution shift as a data or inference problem and do not identify the structural mechanisms underlying error growth.

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

In this paper, we identify one such mechanism in *transient amplification* of rollout error. For a model linearized around its predicted trajectory, the prediction error at each step evolves under a sequence of local linear Jacobian maps. When such a map is *normal* [1] it acts cleanly on errors, stretching them along orthogonal directions so that each direction behaves in isolation. *Non-normal* maps have *non-orthogonal* directions of amplification which interact to produce error growth. For asymptotically stable maps these excursions of growth, called *transients*, eventually die away. Prior work has frequently targeted *stability* as a desirable property that guarantees asymptotic decay of error. Rollouts, however, are evaluated at tens to hundreds of steps, precisely the medium-horizon regime where transients dominate: stable maps can still produce large transient amplification at the horizons that matter in practice.

Normality of individual Jacobians, however, is only part of the story. Nonlinear models have *state-dependent* Jacobians for which the directions of error amplification may rotate with time. When these directions are misaligned across steps, error components can amplify even when no individual step does so on its own. Aligning successive directions can mitigate this amplification; and a sufficient condition for alignment is that Jacobians across steps be *commutative*. Commuting normal maps share common orthogonal eigenbases, which suppresses transient amplification at every horizon. Non-commuting, non-normal Jacobians violate both conditions simultaneously, and unregularized networks trained on non-normal dynamics tend to learn Jacobians with this combination of properties.

To control transient amplification directly, we propose *commutativity regularization*, inclusive of two auxiliary training losses applied over latent space operators. A *commutator penalty* regulates commutativity of consecutive Jacobians, targeting the cross-step compounding of transients; a *normality penalty* discourages individual Jacobians from generating large transients in the first place. The two are complementary: penalizing non-commutativity alone still permits large within-step transients, and penalizing non-normality alone leaves cross-step compounding uncontrolled. Their combination drives the learned dynamics toward a regime in which errors decay at the rate set by the Jacobian spectra with controlled intermediate amplification. At inference, the rollout procedure is unchanged and there is no additional cost.

We validate the method on the following settings: the 1D KdV equation, the 2D Barotropic Vorticity Equation, sea surface temperature data, and fine-tuning of the pretrained FourCastNet global weather model on ERA5 atmospheric data. In each setting, commutativity regularization improves long-horizon rollout accuracy without modifying the inference procedure.

Contributions. We make the following contributions.

(i) We identify non-normal, non-commuting latent Jacobians as a structural driver of transient amplification in autoregressive neural operators, one not addressed by stability.

(ii) We propose commutativity regularization: a commutator penalty together with a normality penalty, both estimated via Jacobian-vector products with zero inference-time overhead.

(iii) We show a propagator bound under approximate commutativity and normality that replaces the per-step spectral norm with the shared spectral radius as the effective decay rate, a strict improvement whenever the Jacobians are non-normal.

(iv) We validate the method on settings spanning chaotic, geophysical, integrable, and synthetic dynamical systems as well as large-scale atmospheric forecasting, obtaining consistent long-horizon accuracy gains over thousands of steps, including in an OOD setting.

2 Autoregressive Neural Operators and Transient Error

Let $F_\theta : \mathcal{X} \to \mathcal{X}$ be an operator trained to advance a dynamical state by one timestep. At inference the model is applied *autoregressively*: given an initial condition $x_0 \in \mathcal{X}$, the rollout is

$$\hat{x}_0 = x_0, \qquad \hat{x}_{t+1} = F_\theta(\hat{x}_t), \quad t = 0, 1, \ldots, T-1. \tag{1}$$

The model is typically trained on a single-step mean-squared loss $\mathcal{L}_{\text{pred}} = \mathbb{E}_t\big[\|F_\theta(x_t) - x_{t+1}\|^2\big]$, which does not expose it to its own prediction errors during training.

[1] A linear operator or matrix A is *normal* if $AA^\top = A^\top A$. Normal matrices have a full set of *orthogonal* eigenvectors.

Error propagation. Let $\varepsilon_t = x_t - \hat{x}_t$ denote the prediction error at step t. Linearising F_θ around the predicted trajectory gives

$$\varepsilon_{t+1} \approx J_t\, \varepsilon_t, \qquad J_t \;=\; \frac{\partial F_\theta}{\partial x}\Big|_{\hat{x}_t}, \tag{2}$$

so that after T steps $\varepsilon_T \approx \Phi_T\, \varepsilon_0$, where the *propagator* is the ordered Jacobian product

$$\Phi_T \;=\; J_{T-1}\, J_{T-2} \cdots J_0. \tag{3}$$

The rollout error is therefore bounded by $\|\varepsilon_T\|_2 \le \|\Phi_T\|_2 \|\varepsilon_0\|_2$. Controlling the spectral norm of individual Jacobians, $\|J_t\|_2 \le \alpha < 1$, yields $\|\Phi_T\|_2 \le \alpha^T$ by submultiplicativity. However, this estimate conceals a key problem: for *non-normal* Jacobians, the intermediate propagator norm $\sup_{t \le T} \|\Phi_t\|_2$ can greatly exceed $\|\Phi_T\|_2$ due to ordering effects between non-commuting factors. This intermediate growth can occur even when each individual factor is contractive and the system is asymptotically stable [31]. This is the transient amplification mechanism we target.

Highlight

Suppressing transient amplification of error across a rollout improves long-horizon accuracy in dynamical models with no inference-time overhead. *Commutativity regularization* achieves this by encouraging normality and commutativity of Jacobians around rollout trajectories.

2.1 Non-Normal Matrices and Transient Growth

A matrix A is *normal* if $A^\top A = A A^\top$. For normal matrices the eigenvectors are orthonormal, the spectral radius equals the spectral norm, $\rho(A) = \|A\|_2$, and there is no transient growth: $\|A^t\|_2 = \rho(A)^t$ for all $t \ge 0$.

For a non-normal matrix, by contrast, it may be that $\rho(A) \ll \|A\|_2$ and $\|A^t\|_2$ can vastly exceed $\rho(A)^t$ for intermediate t (transient growth), even though asymptotically $\lim_{t\to\infty} \|A^t\|_2^{1/t} = \rho(A)$ [8]. The same phenomenon arises in products of *distinct* matrices. For $A_0, \ldots, A_{T-1}$ each satisfying $\|A_t\|_2 \le \alpha$, submultiplicativity gives

$$\|A_{T-1} \cdots A_0\|_2 \;\le\; \alpha^T. \tag{4}$$

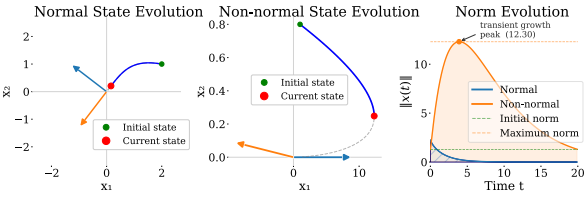


Figure 1: Normal vs. non-normal transient growth on 2×2 stable systems, *Left:* normal system, eigenvectors are orthogonal, *Centre:* non-normal system, eigenvectors are nearly parallel, *Right:* norm $\|x(t)\|$ showing transient growth.

When the matrices are non-commuting, and are not simultaneously diagonalizable, the bound can be highly non-tight, and the product can exhibit significant transient amplification relative to its asymptotic behavior, even when all $\rho(A_t) < 1$.

Illustration. Figure 1 contrasts the two behaviours on 2×2 toy systems. Both matrices have all eigenvalues with negative real part, so both systems are asymptotically stable. The *normal* system has orthogonal eigenvectors: the state decays smoothly along the eigendirections and the norm $\|x(t)\|$ decreases monotonically. The *non-normal* system has nearly parallel eigenvectors: the initial state contains large, nearly-cancelling projections onto them. During the transient phase these components briefly reinforce one another and the norm *grows*, reaching a peak well above the initial value, before eventually decaying. This transient amplification is invisible to a purely eigenvalue-based stability analysis [31], yet it is the mechanism that compounds across time steps in autoregressive rollout for neural operators trained on spatio-temporal data.

2.2 Temporal Commutativity as the LTV Analogue of Normality

The system $x_{t+1} = J_t\, x_t$ is a linear time-varying (LTV) system [26]. For linear time-invariant (LTI) systems (static $J_t = A$), normality ensures $\|A^T\|_2 = \rho(A)^T$, providing tight control of long-horizon amplification. For LTV systems the direct analogue is *temporal commutativity* in addition to normality:

if $[J_{t_1}, J_{t_2}] \equiv J_{t_1} J_{t_2} - J_{t_2} J_{t_1} = 0$ for all t_1, t_2, then the matrices share a common eigenbasis [8] and the propagator satisfies

$$\Phi_T = J_{T-1} \cdots J_0 = U \, \Lambda_{T-1} \cdots \Lambda_0 \, U^\top, \tag{5}$$

where U is the shared eigenbasis. Consequently $\|\Phi_T\|_2 \leq \rho^T$, where $\rho = \max_{i,t} |\lambda_i(J_t)| \leq \alpha$. This recovers a tight LTI-like bound, with the spectral radius ρ (which may be strictly less than the spectral norm α for non-normal matrices) controlling growth rather than the norm. The correspondence between LTI normality and LTV temporal commutativity is shown in Table 1.

Table 1: LTI normality concepts and LTV analogues.

LTI (static A)	LTV (time-varying $\{J_t\}$)
$A^\top A = A A^\top$	$[J_{t_1}, J_{t_2}] = 0 \; \forall \, t_1, t_2$ and $J_t^\top J_t = J_t J_t^\top \; \forall t$
Orthonormal eigenvectors	Shared orthonormal eigenbasis
$\|A^T\|_2 = \rho(A)^T$	$\|\Phi_T\|_2 \leq \rho^T \leq \alpha^T$
No transient growth	No ordering-induced transients

2.3 Error Bound Under Approximate Commutativity and Normality

We now state the key theoretical result motivating our regularizer. The ideal case is when all Jacobians are simultaneously, orthogonally diagonalizable. The following theorem characterises how far the propagator deviates from this ideal when the two conditions, all-pairs commutativity and per-step normality, hold approximately. The proof is deferred to Appendix A.

Theorem 2.1 (Propagator bound under approximate commutativity and normality)**.** *Let* $J_0, J_1, \ldots, J_{T-1} \in \mathbb{R}^{n \times n}$ *satisfy*

(i) $\|J_t\|_2 \leq \alpha < 1$ (contraction) *for all* t*,*

(ii) $\rho(J_t) \leq \rho \leq \alpha$ (spectral radius bound) *for all* t*,*

(iii) $\|[J_s, J_t]\|_F \leq \varepsilon$ *for all* $0 \leq s, t \leq T-1$ (all-pairs approximate commutativity)*, and*

(iv) $\|J_t^\top J_t - J_t J_t^\top\|_F \leq \eta$ (approximate normality) *for all* t*.*

Then the propagator satisfies $\|\Phi_T\|_2 \leq \rho^T + 2T\alpha^{T-1} \, \delta(\varepsilon, \eta)$*, where* $\delta(\varepsilon, \eta) \to 0$ *as* $\varepsilon, \eta \to 0$*.*

The assumption of all-pairs commutativity can be weakened to commutativity of adjacent Jacobians under the additional assumption of a *uniform spectral gap* for the intermediate matrices $J_1, \ldots, J_{T-2}$. See Appendix A.1 for a statement and proof.

2.4 Commutativity and Normality Regularization

Theorem 2.1 provides two auxiliary losses that penalize the mechanisms driving the growth of $\|\Phi_t\|_2$.

Latent-space Jacobians. For a given hidden representation z_t of input x_t for a neural model define the *latent advance map* $G_\theta : \mathbf{z}_t \mapsto \mathbf{z}_{t+1}$ as the bottleneck-to-bottleneck dynamics, and let $\mathbf{J}_t = \partial G_\theta / \partial \mathbf{z}\big|_{\mathbf{z}_t}$ denote its Jacobian. For a UNet we use the bottleneck feature, for FNO we use the middle layer features (Figure 2). Regularizing latent map Jacobians prevents the model from introducing further non-normality in representation space. For large models (such as FourCastNet) we regularize a selected block as shown in Figure 11.

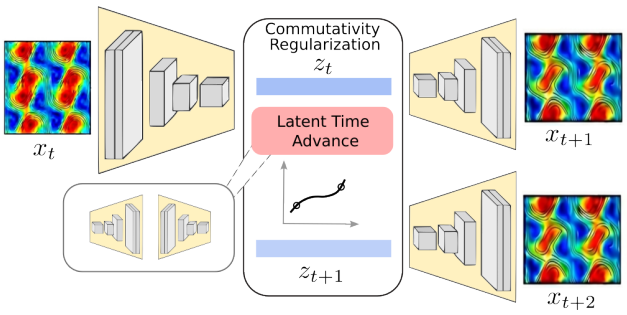


Figure 2: Latent advance architecture used by commutativity regularization. Other configurations possible (See Appendix B.

Commutator penalty. We estimate the squared Frobenius norm of the commutator $[\mathbf{J}_{t+1}, \mathbf{J}_t]$ using a single random probe $\mathbf{v} \sim \mathcal{N}(0, I)$ per minibatch (Hutchinson estimator):

$$\mathcal{L}_{\text{comm}} = \mathbb{E}_{\mathbf{v}} \left[\left\| \mathbf{J}_{t+1} \, \mathbf{J}_t \, \mathbf{v} - \mathbf{J}_t \, \mathbf{J}_{t+1} \, \mathbf{v} \right\|^2 \right]. \tag{6}$$

Normality penalty. To suppress within-step transient amplification we additionally penalise the normality defect of each $\mathbf{J}_t$:

$$\mathcal{L}_{\text{norm}} \;=\; \mathbb{E}_{\mathbf{v}}\Big[\big\|\mathbf{J}_t^\top \mathbf{J}_t\,\mathbf{v} \;-\; \mathbf{J}_t\,\mathbf{J}_t^\top\,\mathbf{v}\big\|^2\Big]. \tag{7}$$

This penalises $\|J_t^\top J_t - J_t J_t^\top\|_F^2$ in expectation. Together with $\mathcal{L}_{\text{comm}}$, it drives $\mathbf{J}_t$ toward the ideal case of Theorem 2.1: contractive, normal, and temporally commutative.

Training objective. The full training loss is

$$\mathcal{L} \;=\; \mathcal{L}_{\text{pred}} \;+\; \lambda_{\text{c}}\,\mathcal{L}_{\text{comm}} \;+\; \lambda_{\text{n}}\,\mathcal{L}_{\text{norm}}, \tag{8}$$

where $\mathcal{L}_{\text{pred}}$ is the standard single-step MSE loss.

2.5 Efficient JVP-Based Implementation

Matrix-free computation. All four JVPs required for $\mathcal{L}_{\text{comm}}$ and the two JVPs plus one VJP required for $\mathcal{L}_{\text{norm}}$ are computed using `torch.func.jvp` and `torch.func.vjp` (PyTorch 2.0+) [19]. No explicit $n \times n$ Jacobian matrix is stored at any point. The sequence of operations for $\mathcal{L}_{\text{comm}}$ is:

$$\mathbf{u} = \mathbf{J}_{t+1}\,\mathbf{v}, \qquad \mathbf{p} = \mathbf{J}_t\,\mathbf{u} = \mathbf{J}_t\mathbf{J}_{t+1}\,\mathbf{v}, \tag{9}$$

$$\mathbf{w} = \mathbf{J}_t\,\mathbf{v}, \qquad \mathbf{q} = \mathbf{J}_{t+1}\,\mathbf{w} = \mathbf{J}_{t+1}\mathbf{J}_t\,\mathbf{v}, \tag{10}$$

after which $\mathcal{L}_{\text{comm}} = \|\mathbf{p} - \mathbf{q}\|^2$. Each `jvp` call is a single forward pass over the latent-advance sub-network, applied to the flattened latent. For $\mathcal{L}_{\text{norm}}$:

$$\mathbf{J}_t\,\mathbf{v} \;\xrightarrow{\texttt{vjp}}\; \mathbf{J}_t^\top(\mathbf{J}_t\,\mathbf{v}), \qquad \mathbf{J}_t^\top\,\mathbf{v} \;\xrightarrow{\texttt{jvp}}\; \mathbf{J}_t\big(\mathbf{J}_t^\top\,\mathbf{v}\big), \tag{11}$$

yielding $(\mathbf{J}_t^\top\mathbf{J}_t - \mathbf{J}_t\mathbf{J}_t^\top)\mathbf{v}$ as required by (7).

3 Related Work

Long-horizon error in autoregressive neural simulators. The dominant framing of rollout drift is *distributional shift*: a model trained on ground-truth states must, at inference, consume its own noisy predictions. PDE-Refiner [14] addresses this by training a denoising network that iteratively refines each rollout step, and Spectral-Refiner [6] fine-tunes an FNO to correct mis-resolved Fourier modes; conditional diffusion approaches [10] go further, running many denoising iterations per step in exchange for trajectory-level error correction. These methods target a symptom of the problem rather than its structural mechanism and impose substantial inference-time cost. McCabe et al. [16] employ the method of [18] and regularize the spectral norm of each weight matrix, estimated using power iteration , focusing on stability rather than transient amplification which is the goal of this paper.

Linear-operator views and non-normal dynamics. Koopman approaches [17, 33, 1] model the latent dynamics with a single *linear* operator K, for which all Jacobians along a trajectory commute trivially. Commutativity regularisation can be read as a relaxation of the Koopman view: it asks only that the time-varying latent Jacobians *share* an orthogonal eigenbasis with state-dependent eigenvalues, recovering the favourable spectral-radius propagator bound while permitting genuinely nonlinear dynamics. The phenomenon that non-normal operators can amplify perturbations even when individually contractive originates in hydrodynamic stability theory and pseudospectra [32, 28, 31, 11]; we transfer this lens from physical operators to the learned Jacobians of neural simulators and turn it into a training signal.

Multi-step and unrolled training. A natural alternative to single-step training is to optimise the rollout itself [15]. The pushforward trick [5] backpropagates through one rollout step while detaching the gradient at an earlier prediction. Such schemes mitigate covariate shift and bring training closer to the deployment regime, but their gradient signal degrades rapidly with the unroll length: Scieur et al. [29] show that differentiating through long unrolls suffers from a "curse of unrolling" in which the gradient becomes either vanishing or ill-conditioned, limiting the practical horizon over which they can shape the operator. Commutativity regularisation acts on the same composition of

Jacobians but estimates their structural defect locally, with two-step probes, side-stepping long-unroll backpropagation entirely. The two ideas are orthogonal and can be combined: a short pushforward window controls covariate shift while comm-reg controls the per-step structural amplification.

Training-time noise injection. An alternative stabilisation line injects perturbations into rollout inputs during training, so that the model learns to recover from its own errors. Originally introduced for graph-network particle simulators [27] and mesh-based fluid simulators [21], the technique has since been applied to turbulence emulators [30] to stabilise rollouts well beyond the training horizon. Like multi-step training, these schemes target the input distribution rather than the structural Jacobian properties that drive transient amplification, and are orthogonal to commutativity regularisation.

4 Experiments

We evaluate commutativity regularisation across four spatio-temporal forecasting settings spanning 1D and 2D, integrable and chaotic, and synthetic and real data: the 1D KdV equation, the 2D barotropic vorticity equation, fine-tuning of the pretrained FourCastNet weather model on ERA5, and an OISST sea surface temperature benchmark. In each setting we compare a baseline trained with the standard prediction loss against an otherwise-identical model trained with the same objective plus the commutator and normality penalties of Eq. (8); the rollout procedure is unchanged at inference, and we evaluate on horizons of hundreds to thousands of steps, well beyond the training window. Across architectures (U-Net, FNO, U-FNO, and FourCastNet's AFNO backbone) the regularisation consistently extends the horizon over which the learned operator stays close to ground-truth trajectories, with the largest gains where transient amplification is most pronounced: deep rollouts and out-of-distribution initial conditions.

4.1 Korteweg–de Vries equation

The KdV equation is a clean probe for our central claim. It is integrable, so any long-horizon divergence between two rollouts on the same initial condition is attributable to the learned operator rather than to chaotic loss of predictability. We train every model on a short window of *single*-soliton dynamics (200 rollout steps, 10 s of physical time) and then evaluate the autoregressive rollout out to $25\times$ that horizon, both *in-distribution* and on an *out-of-distribution* test set whose initial conditions contain up to three interacting solitons, a regime never seen during training.

We consider four backbones: a 1D UNet, FNO [13], U-FNO, PDE-Refiner [14] as a representative inference-time correction baseline. Every regime is trained with a one-step MSE loss on the same 256-trajectory single-soliton dataset. Results are shown in Figure 6 and appendix Tables 3, 4, and 5.

In-distribution: commutativity reg. wins at long horizons. At very short leads, the unregularised UNet baseline is *slightly more accurate* than UNet+CR : the regulariser pays a small single-step cost. This ordering reverses already inside the training window and the gap then widens monotonically. By step 1000 UNet+CR shows a $5\times$ improvement over the baseline. PDE-Refiner is the strongest unregularised baseline at the shorter horizons, it is more accurate than UNet+CR up to $\sim$1000 steps, paying $5\times$ inference-time cost, but it eventually accumulates error and at step 5000, it is $5\times$ worse than the training-time regulariser despite its much inference cost.

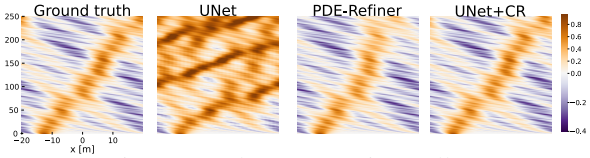


Figure 3: KdV UNet variant rollout

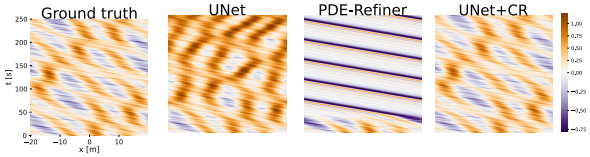


Figure 4: KdV UNet variant OOD rollout

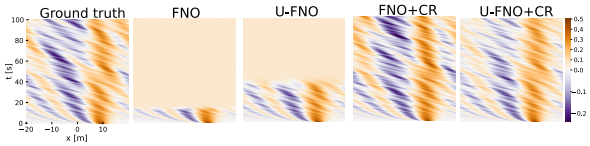


Figure 5: KdV FNO variant Rollouts

Out-of-distribution: the gap grows on multi-soliton ICs. On the multi-soliton OOD split the unregularised UNet and PDE-Refiner both experience severe degradation, failing to maintain rollout coherence. UNet+CR is

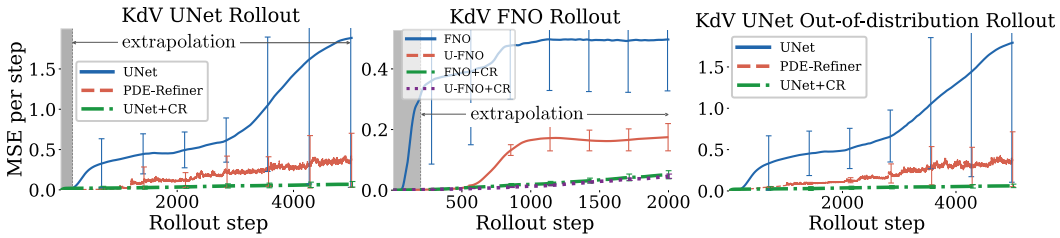


Figure 6: KdV rollout nMSE vs. time, averaged over 50 held-out test trajectories. The dashed line marks the training horizon ($t = 10$ s, 200 steps); everything to its right is extrapolation in rollout length. onward. (Cf. Tables 3, 4, 5)

essentially unaffected by the distribution shift. The regularised model is the only one that stays on a coherent multi-soliton trajectory at the longest horizons. The qualitative space-time visualization is in Figures 3 and 4.

Spectral backbones: comm. reg. recovers an unstable rollout. On the same data, the FNO and U-FNO backbones exhibit a stronger version of the same effect (Figure 6, middle; detailed results in Table 5). The unregularised FNO has an excellent one-step MSE on the validation set, $\sim 4\times$ better per step than its regularised counterpart) yet its autoregressive rollout becomes unstable around step 100 and loses coherence already by the end of the training window. The U-FNO baseline holds longer, until step 500, before collapsing. Both commutativity-regularised variants stay bounded throughout the 2000-step evaluation window, comparable to the regularised UNet at the same horizon and is reached on a backbone that diverges entirely without the penalty. On the spectral architectures the regulariser thus does more than improve a stable baseline: it converts a divergent autoregressive rollout into a bounded one, exactly the qualitative behaviour the propagator analysis of Section 2 predicts when non-normal Jacobian products are left unconstrained.

Summary. Across all four backbones, commutativity regularisation trades a small amount of one-step accuracy for a long-horizon rollout that is bounded and, beyond the training window, between $5\times$ and several orders of magnitude more accurate than its unregularised counterpart. It matches PDE-Refiner at intermediate horizons without the latter's $5\times$ inference-time overhead, and outperforms it past $\sim$2000 steps. The OOD experiment shows that the gain is robust to a regime shift in the initial conditions, suggesting that the regulariser shapes a structural property of the learned operator rather than memorising a training distribution. Per-trajectory space–time snapshots, the full per-horizon nMSE numbers, and the architectural and training details are in Appendix D.

4.2 Barotropic Vorticity Equation

We evaluate on a chaotic 2D fluid PDE: the barotropic vorticity equation (BVE). The system is dissipative and chaotic: small perturbations amplify exponentially in time, so any improvement must model intrinsic divergence.

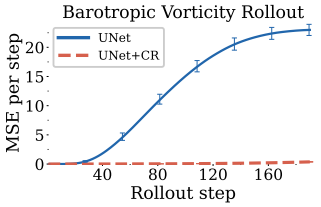


Figure 7: BVE rollout RMSE on the held-out test set.

Setup. The backbone is a 2D UNet with circular padding, and an $8\times8\times256$ bottleneck on which the regulariser acts. Both regimes are trained with one-step MSE on 200-frame trajectories. Further details are deferred to Appendix E.

Result: the baseline destabilises inside the training window. Figure 7 and Table 8 report the comparison. The two regimes are essentially tied at one step but they quickly separate fast. By t=1 s (20 steps) the baseline is already $5\times$ worse than the regularised model. RMSE=1, error variance equal to signal variance, and from $t\approx$3 s onward it saturates at RMSE$\approx$3–6, where the rollout has decorrelated from the ground truth. The regularised model stays below RMSE=1 throughout the training horizon. At the end of the 10 s window the regularised RMSE is $9\times$ smaller than the baseline; in nMSE terms (the operationally relevant unit on this normalised field) the gap is $82\times$.

The qualitative picture is in the snapshot grid in Figure 8 and Appendix E.6: the baseline visibly amplifies vorticity to several times the natural scale of the flow well before the end of the training horizon, while the regularised rollout preserves coherent vortex structures and the correct amplitude band throughout.

Summary. On a 2D chaotic PDE the unregularised UNet rollout becomes unstable inside the training horizon. The same training-time regulariser that controls long-horizon drift on KdV turns this instability into a bounded, on-attractor rollout, with a $9\times$ RMSE improvement at the end of the $10\,\mathrm{s}$ horizon and a $\sim 80\times$ improvement in nMSE. Together with the KdV results, this validates the effectiveness the regulariser across 1D and 2D and integrable vs. chaotic. In both regimes, single-step MSE and rollout error rank the regimes in opposite orders, and a Jacobian-product-aware regulariser recovers the long-horizon behaviour that the standard one-step training objective leaves unconstrained. Per-step RMSE numbers, full data generation, architecture and hyperparameters, and more visualizations are in Appendix E.

4.3 FourCastNet on ERA5

We test commutativity regularisation on a large pretrained autoregressive weather model: FourCastNet v1 [20], an AFNO that maps the 20-channel ERA5 [7, 23] state at 720×1440 ($0.25°$) to the state $6\,\mathrm{h}$ later. The question is whether the regulariser improves long-horizon rollout when FCN is fine-tuned on a small, in-distribution slice of its own training data, so that any gain has to come from the structural effect of the regulariser on the learned Jacobians. The main finding is that plain supervised fine-tuning on this slice actively *worsens* the long-horizon near-surface temperature forecast, while fine-tuning with commutativity regularisation on the same data turns this regression into a substantial improvement. Results are shown in Figure 9 and Tables 10 and 11.

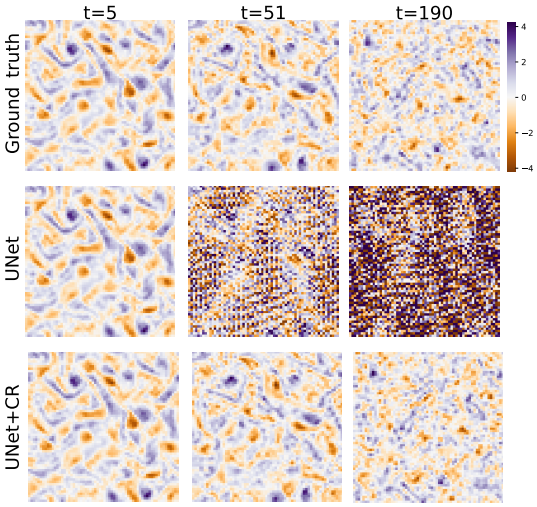


Figure 8: Barotropic vorticity unregularized (middle) and regularized (bottom) UNet rollout

Setup. Starting from the pretrained FCNv1 [2] backbone, we fine-tune three years (2013-2015) inside FCN's own 1979–2015 training range and compare against (i) the frozen pretrained model and (ii) a plain supervised fine-tune on 2013–2015 with the same optimiser and schedule but no regularization. The regularization is applied on the deep AFNO blocks: the first 8 blocks are run once and detached, and the penalties are evaluated on the last two blocks. In this case, we apply the regularizer to layer normalized blocks. We evaluate 40-step (10-day) autoregressive rollouts on ERA5 2018 at 41 initial conditions, following [20], and report latitude-weighted RMSE in physical units on the variables z_{500} and $t_{2\mathrm{m}}$. A second held-out evaluation on ERA5 2019 shows the same improvement for the three-year finetuning. Architecture, hyperparameters, visualizations and other details are in Appendix F.

Plain finetuning worsens $t_{2\mathrm{m}}$, the regulariser recovers it. Table 10 reports the $t_{2\mathrm{m}}$ rollout RMSE on ERA5 2018. Through day 5 the three regimes are tied within $\sim 5\%$. From day 7 onwards they split sharply: the plain finetune *regresses* relative to the frozen model , while the regularised finetune on the same data improves on the frozen model. The 2019 replication (Appendix F.4) shows the same pattern. On the upper-air channels z_{500}, t_{850}, u_{850} (all four in Appendix F.4) the regularised and frozen models are within ~ 1–3% at every lead, while the plain finetune again drifts at the longest lead (e.g. t_{850} day 10: $4.26\,\mathrm{K}$ frozen, $6.33\,\mathrm{K}$ plain finetune, $4.14\,\mathrm{K}$ regularised). The regulariser thus matches the frozen model on the variables FCN already gets right and substantially improves on the variable it most struggles with.

[2] https://github.com/NVlabs/FourCastNet

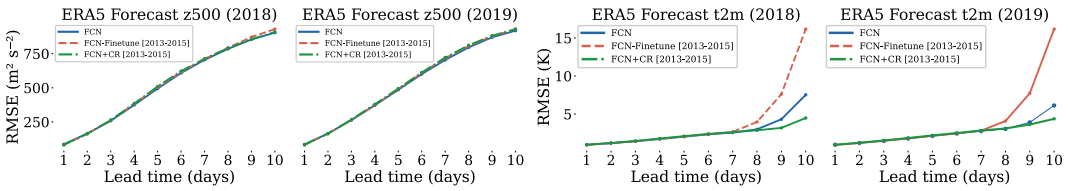


Figure 9: ERA5 rollout RMSE vs. lead time on $t_{2\mathrm{m}}$ and z_{500} for held-out years 2018 and 2019 together with finetuning without regularization. Finetuning-only worsens long-horizon $t_{2\mathrm{m}}$; the same data with commutativity regularisation pulls well below the frozen FCN baseline.

Summary. On a 74 M-parameter pretrained weather model, fine-tuning on a small in-distribution slice of its own training years with the same training-time regulariser used for the synthetic PDEs flips a catastrophic $3\times$ degradation of $t_{2\mathrm{m}}$ at day 10 into a 41% improvement (and a comparable improvement on the held-out 2019 year), without exposure to new data and without inference-time cost. Together with the KdV and BVE results, this brings the validation from 1D integrable and 2D chaotic synthetic settings up to a high-resolution real-world weather model, on the variable for which the iterated rollout most visibly fails. The full per-channel, per-lead, multi-year tables, the regulariser implementation details and the architectural hyperparameters are in Appendix F.

4.4 Sea Surface Temperature Forecast

We train and compared a regularized model from scratch on a real, observation-derived field: weekly NOAA OISST sea-surface temperature on a global 1° grid [24]. A 2D UNet (k_{in}=1, depth 4, base width 64) is trained with one-step MSE on the 1200-week training split, with and without the latent commutativity / normality penalties of Section 1 on its bottleneck. We initialise from the first frame of the 377-week test split and roll autoregressively for 376 weeks (≈ 7.2 years). Setup, architecture and hyperparameters are in Appendix G.

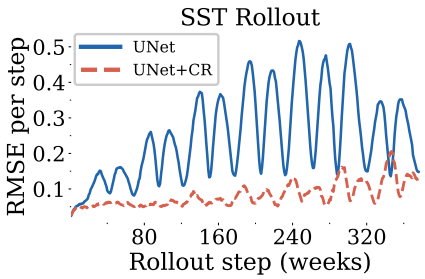


Figure 10: SST rollout RMSE (normalised units) versus lead time,

The two models are tied at one-week lead, i.e. the regulariser does not hurt short-lead accuracy. They diverge from $\sim$ 3 months onward: the baseline drifts in phase relative to the annual cycle and amplifies anomalies, while the regularised rollout tracks the seasonal cycle out to the end of the $\sim$ 7-year window. Averaged over the full 376-step rollout the regularised model is $3.0\times$ more accurate. Together with the BVE result, this shows that the same regulariser controls two qualitatively different long-horizon failure modes, chaotic energy saturation on BVE, and slow phase drift on a forced near-periodic real-world field.

5 Conclusion

We identified *transient amplification* of perturbations along rollout trajectories, driven by non-normal and non-commuting latent Jacobians, as a structural mechanism for long-horizon error growth in autoregressive neural simulators and responsible from distributional shift. On the basis of this analysis we introduced *commutativity regularisation*: a pair of JVP-based penalties on per-step Jacobian normality and across-step commutativity, supported by a propagator bound that replaces the spectral-norm α^T rate by the spectral-radius ρ^T rate as the two penalties go to zero. Across KdV, the barotropic vorticity equation, sea surface temperature data, and fine-tuning of the pretrained FourCastNet weather model on ERA5, the method consistently improves long-horizon accuracy at zero inference-time cost, and pushes regularised models well past the rollout horizons at which baselines diverge.

Limitations and outlook. The propagator bound is linearisation-based and assumes that the local Jacobians are representative of the rollout error map; far from the training distribution, where the linearisation itself is suspect, our guarantees become only suggestive. The penalties are estimated stochastically with single-probe JVPs, and per-step normality together with adjacent-step commuta-

tivity is a tractable surrogate for the all-pairs joint diagonalisability condition that the theory asks for. The method also targets the *structural* component of error growth and is therefore complementary to, rather than a replacement for, refinement and diffusion methods that address distributional shift; we expect the strongest results from combining the two. Finally, the regulariser is most natural for architectures with an explicit latent bottleneck through which temporal dynamics must pass; choosing the regularised subspace for purely spectral or attention-based simulators, and scaling the method to global weather and climate models at higher resolution, are the directions we view as most pressing.

Acknowledgements

AP. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 101034413. *FL.* This research was funded in whole or in part by the Austrian Science Fund (FWF) [10.55776/COE12].

References

[1] Omri Azencot, N. Benjamin Erichson, Vanessa Lin, and Michael W. Mahoney. Forecasting sequential data using consistent Koopman autoencoders. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 475–485, 2020. URL https://proceedings.mlr.press/v119/azencot20a.html.

[2] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13: 2453, 2022. doi: 10.1038/s41467-022-29939-5.

[3] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotian Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, 619:533–538, 2023. doi: 10.1038/s41586-023-06185-3.

[4] Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan Weyn, Haiyu Dong, Anna Vaughan, Jayesh K. Gupta, Kit Tambiratnam, Alex Archibald, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris. Aurora: A foundation model of the atmosphere. *arXiv preprint arXiv:2405.13063*, 2024.

[5] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU.

[6] Shuhao Cao, Francesco Brarda, Rui Peng Li, and Yuanzhe Xi. Spectral-refiner: Accurate fine-tuning of spatiotemporal fourier neural operator for turbulent flows. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=MKP1g8wU0P.

[7] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia de Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. doi: https://doi.org/10.1002/qj.3803. URL https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3803.

[8] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.

[9] Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, Sam Hatfield, Peter Battaglia, Alvaro Sanchez-Gonzalez, Matthew Willson, Michael P. Brenner, and Stephan Hoyer. Neural general circulation models for weather and climate. *Nature*, 632:1060–1066, 2024.

[10] Georg Kohl, Li-Wei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. *Neural Networks*, 199:108641, 2026. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2026.108641. URL https://www.sciencedirect.com/science/article/pii/S0893608026001036.

[11] Heinz-Otto Kreiss. Über die Stäbilitätsdefinition für Differenzengleichungen die partielle Differentialgleichungen approximieren. *BIT Numerical Mathematics*, 2(3):153–181, 1962. doi: 10.1007/BF01957346.

[12] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weiwen Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336.

[13] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.08895.

[14] Phillip Lippe, Bastiaan S. Veeling, Paris Perdikaris, Richard E Turner, and Johannes Brandstetter. PDE-refiner: Achieving accurate long rollouts with neural PDE solvers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Qv6468llWS.

[15] Bjoern List, Li-Wei Chen, Kartik Bali, and Nils Thuerey. Differentiability in unrolled training of neural physics simulators on transient dynamics. *Computer Methods in Applied Mechanics and Engineering*, 433:117441, January 2025. ISSN 0045-7825. doi: 10.1016/j.cma.2024.117441. URL https://www.sciencedirect.com/science/article/pii/S0045782524006960.

[16] Michael McCabe, Peter Harrington, Shashank Subramanian, and Jed Brown. Towards stability of autoregressive neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=RFfUUtKYOG.

[17] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, 2005. doi: 10.1007/s11071-005-2824-x.

[18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[20] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Anima Anandkumar. FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022. URL https://arxiv.org/abs/2202.11214.

[21] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[22] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. Probabilistic weather forecasting with machine learning. *Nature*, 637:84–90, 2025.

[23] Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallegue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. Weatherbench 2: A benchmark for the next generation of data-driven global weather models, 2023.

[24] Richard W. Reynolds, Nick A. Rayner, Thomas M. Smith, Diane C. Stokes, and Wanqiu Wang. An improved in situ and satellite SST analysis for climate. *Journal of Climate*, 15(13): 1609–1625, 2002. doi: 10.1175/1520-0442(2002)015<1609:AIISAS>2.0.CO;2.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015. doi: 10.1007/978-3-319-24574-4_28.

[26] Wilson John Rugh. *Nonlinear system theory*. Johns Hopkins University Press Baltimore, 1981.

[27] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning (ICML)*, 2020.

[28] Peter J. Schmid. Nonmodal stability theory. *Annual Review of Fluid Mechanics*, 39:129–162, 2007. doi: 10.1146/annurev.fluid.38.050304.092139.

[29] Damien Scieur, Gauthier Gidel, Quentin Bertrand, and Fabian Pedregosa. The curse of unrolling: Rate of differentiating through optimization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=b57KM4ydqpp`.

[30] Kimberly Stachenfeld, Drummond B. Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation. In *International Conference on Learning Representations (ICLR)*, 2022.

[31] Lloyd N. Trefethen and Mark Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, Princeton, NJ, 2005.

[32] Lloyd N. Trefethen, Anne E. Trefethen, Satish C. Reddy, and Tobin A. Driscoll. Hydrodynamic stability without eigenvalues. *Science*, 261(5121):578–584, 1993. doi: 10.1126/science.261.5121.578.

[33] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015. doi: 10.1007/s00332-015-9258-5.

A Proof of theorem 2.1

We first record the exact result when the Jacobians commute and are individually normal.

Proposition A.1 (Exact commuting case)**.** *Let $J_0, \dots, J_{T-1} \in \mathbb{R}^{n\times n}$ be simultaneously diagonalisable as $J_t = U\Lambda_t U^\top$ for a common orthogonal matrix U and diagonal $\Lambda_t = \mathrm{diag}(\lambda_{1,t}, \dots, \lambda_{n,t})$. If $\rho = \max_{i,t} |\lambda_{i,t}|$, then*

$$\|\Phi_T\|_2 \;\le\; \rho^T. \tag{12}$$

Proof. $\Phi_T = U\,\Lambda_{T-1}\cdots\Lambda_0\,U^\top$. Since U is orthogonal, $\|\Phi_T\|_2 = \|\Lambda_{T-1}\cdots\Lambda_0\|_2$. The product of diagonal matrices is diagonal with (i,i) entry $\prod_{t=0}^{T-1}\lambda_{i,t}$, so $\|\Lambda_{T-1}\cdots\Lambda_0\|_2 = \max_i |\prod_t \lambda_{i,t}| \le \rho^T$. □

Remark A.2. *The improvement over the crude submultiplicativity bound α^T (where $\alpha = \max_t \|J_t\|_2 \ge \rho(J_t) \ge \rho$) is real whenever the Jacobians are* non-normal*: a non-normal matrix satisfies $\rho(J) < \|J\|_2$, so $\rho < \alpha$ and $\rho^T \ll \alpha^T$ for large T. Proposition A.1 therefore shows that commutativity (plus normality) lets the propagator grow at the* spectral-radius *rate rather than the* spectral-norm *rate.*

Theorem 2.1 (Propagator bound under approximate commutativity and normality)**.** *Let $J_0, J_1, \dots, J_{T-1} \in \mathbb{R}^{n\times n}$ satisfy*

- *(i)* $\|J_t\|_2 \le \alpha < 1$ (contraction) *for all t,*
- *(ii)* $\rho(J_t) \le \rho \le \alpha$ (spectral radius bound) *for all t,*
- *(iii)* $\|[J_s, J_t]\|_F \le \varepsilon$ *for all* $0 \le s, t \le T-1$ (all-pairs approximate commutativity)*, and*
- *(iv)* $\|J_t^\top J_t - J_t J_t^\top\|_F \le \eta$ (approximate normality) *for all t.*

Then the propagator satisfies $\|\Phi_T\|_2 \;\le\; \rho^T \;+\; 2T\alpha^{T-1}\,\delta(\varepsilon,\eta)$, where $\delta(\varepsilon,\eta) \to 0$ as $\varepsilon, \eta \to 0$.

Proof. When $\varepsilon = \eta = 0$, each J_t is normal and all pairs commute. Commuting normal matrices are simultaneously diagonalizable by a common orthogonal matrix [8], giving $\|\Phi_T\|_2 \le \rho^T$ by Proposition A.1.

For $\varepsilon, \eta > 0$, the joint conditions (iii)–(iv) imply that $\{J_t\}$ lies within distance $\delta(\varepsilon,\eta)$ of the closed set of simultaneously orthogonally diagonalizable families, i.e. there exist $J'_t = U\Lambda_t U^\top$ (orthogonal U, diagonal Λ_t with $|\lambda_{i,t}| \le \rho$) satisfying $\|J_t - J'_t\|_2 \le \delta(\varepsilon,\eta)$.

Write $J_t = J'_t + E_t$ with $\|E_t\|_2 \le \delta \equiv \delta(\varepsilon,\eta)$. Expanding the product to first order in $\{E_t\}$:

$$\Phi_T \;=\; \prod_{t=0}^{T-1}(J'_t + E_t) \;=\; \underbrace{\prod_{t=0}^{T-1} J'_t}_{=:\,\Phi'_T} \;+\; \sum_{k=0}^{T-1} \underbrace{J'_{T-1}\cdots J'_{k+1}}_{L_k}\, E_k\, \underbrace{J_{k-1}\cdots J_0}_{R_k} \;+\; O(\delta^2).$$

Since $J'_t = U\Lambda_t U^\top$, we have $\|\Phi'_T\|_2 = \|\Lambda_{T-1}\cdots\Lambda_0\|_2 \le (\rho+\delta)^T \le \rho^T + T\rho^{T-1}\delta \le \rho^T + T\alpha^{T-1}\delta$. Each cross-term satisfies $\|L_k E_k R_k\|_2 \le \alpha^{T-1}\delta$. Summing the T cross-terms and absorbing the $O(\delta^2)$ remainder into the leading terms for small δ:

$$\|\Phi_T\|_2 \;\le\; \rho^T + T\alpha^{T-1}\delta + T\alpha^{T-1}\delta$$

To show that $\delta(\varepsilon,\eta) \to 0$ we work with fixed n, T, α, ρ. Define a metric on T-tuples by

$$d_\infty((A_t),(B_t)) := \max_{0\le t<T} \|A_t - B_t\|_2.$$

Let

$$\mathcal{K} := (J_0, \dots, J_{T-1}) \in (\mathbb{C}^{n\times n})^T : \|J_t\|_2 \le \alpha,\ \rho(J_t) \le \rho \text{ for all } t.$$

This set is compact because it is closed and bounded in a finite-dimensional space. Define the exact set

$$\mathcal{S} := \{(U\Lambda_0 U^*, \dots, U\Lambda_{T-1}U^*) : U \in \mathsf{U}(n),\ \Lambda_t \text{ diagonal},\ \max_i |\lambda_{t,i}| \le \rho\}.$$

The set $\mathcal{S}$ is compact, hence closed.

Now define δ in terms of the sets

$$\delta(\varepsilon,\eta) := \sup d_\infty((J_t),\mathcal{S}) : (J_t) \in \mathcal{K}, \max_{s<t} \|J_sJ_t - J_tJ_s\|_2 \le \varepsilon, \max_t \|J_tJ_t^* - J_t^*J_t\|_2 \le \eta.$$

Then $\delta(\varepsilon,\eta) \to 0$ as $\varepsilon,\eta \to 0$.

Assume the contrary. Then there exist $\gamma > 0$, sequences $\varepsilon_m, \eta_m \to 0$, and tuples $(J_t^{(m)}) \in \mathcal{K}$ satisfying the two approximate conditions but

$$d_\infty((J_t^{(m)}),\mathcal{S}) \ge \gamma.$$

By compactness of $\mathcal{K}$, there is a convergent subsequence,

$$(J_0^{(m)},\dots,J_{T-1}^{(m)}) \to (J_0,\dots,J_{T-1}) \in \mathcal{K}.$$

The commutator and normality maps are continuous, so the limit tuple satisfies

$$J_sJ_t = J_tJ_s, \qquad J_tJ_t^* = J_t^*J_t.$$

Thus the J_t's are pairwise commuting normal matrices, hence simultaneously unitarily diagonalizable. Since also $\rho(J_t) \le \rho$, the limit tuple lies in $\mathcal{S}$. Therefore

$$d_\infty((J_t^{(m)}),\mathcal{S}) \to 0,$$

contradicting the lower bound γ. Hence $\delta(\varepsilon,\eta) \to 0$, as required and the bound follows. □

Remark A.3 (Why both conditions are needed)**.** *Condition (iii) alone (commutativity, without normality) yields commuting approximants J_t' that share an eigenbasis, but that basis need not be orthogonal, so the $\|\Phi_t'\|_2$ bound would require a conditioning factor for the eigenbasis. Condition (iv) alone (normality, without commutativity) ensures each J_t individually has $\|J_t^k\|_2 = \rho(J_t)^k$, but products of distinct normal matrices need not satisfy $\|\Phi_T\|_2 \le \rho^T$. The two conditions are complementary: together they drive the sequence toward a simultaneously orthogonally diagonalizable family.*

Remark A.4 (Spectral norm alone is insufficient)**.** *Condition (i) without (iii)–(iv) does not prevent transient amplification. There exist sequences $\{J_t\}$ with $\|J_t\|_2 = \alpha < 1$ yet $\|\Phi_T\|_2 \gg \alpha^T$ for T in the transient regime, due to non-commuting, non-normal interactions.*

A.1 Weakening All-pairs Commutativity to Adjacent Commutativity

We can weaken all-pairs commutativity to adjacent commutativity under the additional assumption of a *uniform spectral gap* for the intermediate matrices $J_1,\dots,J_{T-2}$.

We assume there is $\gamma > 0$ such that for every intermediate $t = 1,\dots,T-2$, the eigenvalues of J_t are simple and satisfy

$$\min_{i\ne j} |\lambda_i(J_t) - \lambda_j(J_t)| \ge \gamma.$$

Then adjacent commutativity can propagate along the chain.

In the exact case with a *simple* spectrum

$$[J_t, J_{t+1}] = 0 \;\forall t \quad \text{and} \quad J_1,\dots,J_{T-2} \text{ simple}$$

implies simultaneous diagonalizability by induction.

A.1.1 Approximate theorem with adjacent commutativity

We only need to show that $\delta_(\varepsilon, \eta; \gamma) \to 0$ as $\varepsilon, \eta \to 0$. The proof uses the same compactness contradiction argument as in the proof of 2.1.

Lemma A.5. *Fix $n, T, \alpha, \rho, \gamma > 0$. Let $\mathcal{K}_\gamma$ be the compact set of tuples $(J_0, \ldots, J_{T-1})$ satisfying the norm and spectral-radius bounds, and such that each intermediate J_t, $1 \le t \le T-2$, has eigenvalue gap at least γ. Then there exists*

$$\delta(\varepsilon, \eta; \gamma) \to 0 \qquad \textit{as} \qquad \varepsilon, \eta \to 0$$

such that if

$$\max_{0 \le t < T-1} \|J_t J_{t+1} - J_{t+1} J_t\|_2 \le \varepsilon$$

and

$$\max_t \|J_t J_t^* - J_t^* J_t\|_2 \le \eta,$$

then the tuple (J_t) lies within distance $\delta(\varepsilon, \eta; \gamma)$ of the set of simultaneously unitarily diagonalizable tuples.

Suppose not. Then there exist $\varepsilon_m, \eta_m \to 0$ and tuples $(J_t^{(m)}) \in \mathcal{K}_\gamma$ satisfying adjacent commutativity and approximate normality, but staying a fixed distance away from the simultaneously diagonalizable set. By compactness, there is a convergent subsequence,

$$(J_0^{(m)}, \ldots, J_{T-1}^{(m)}) \to (J_0, \ldots, J_{T-1}).$$

The limit is exactly normal and exactly adjacent-commuting. The eigenvalue gap condition passes to the limit, so $J_1, \ldots, J_{T-2}$ have simple spectra and are simultaneously diagonalizable. This leads to a contradiction to positive distance showing $\delta(\varepsilon, \eta; \gamma) \to 0$.

B Regularizer configuration

For very large models where the regularizing the latent advance in infeasible (such as FourCastNet), we instead regularize a chosen block in the network. A schematic of this is show in Figure 11.

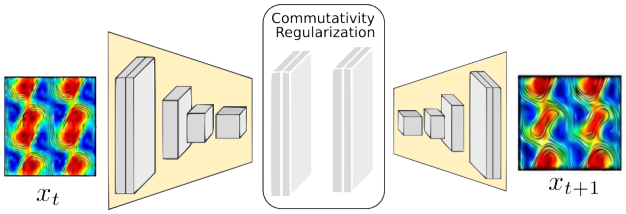


Figure 11: Latent block commutativity regularization. Use only for the FourCastNet experiment in this paper.

C Reducing computational overhead

Three practical approximations reduce cost with negligible quality loss. (a) *Temporal subsampling*: the regularization terms are evaluated only every $K \in \{1, 5, 10\}$ minibatches depending on the experiment, since Jacobian structure changes slowly relative to individual gradient steps. (b) Jacobian vector products are chosen on a smaller sample (about half) of each minibatch, and (c) *Single shared probe*: one probe vector $\mathbf{v}$ is drawn per minibatch and reused for all examples, rather than drawing per-example vectors. With these approximations the additional training wall-clock cost is approximately 20–30% over the baseline. The inference cost is *exactly zero*: the regularizer affects only the learned weights θ, not the rollout procedure.

D KdV experiment: details

This appendix expands the KdV experiment of Section 4.1: data and OOD construction, the four backbone architectures, training and regulariser hyperparameters, and the full per-horizon nMSE numbers underlying Figure 6 and Tables 3, 4 and 5.

D.1 Data generation and OOD construction

Equation and solver. We solve the 1D Korteweg–de Vries equation

$$u_t + u\,u_x + u_{xxx} \;=\; 0, \qquad x \in [-20, 20], \qquad u(\cdot, t) \text{ periodic}, \tag{13}$$

on a uniform grid of $N = 256$ points. Spatial derivatives are taken pseudospectrally, $u_x = \mathcal{F}^{-1}[ik\hat{u}]$ and $u_{xxx} = \mathcal{F}^{-1}[(ik)^3\hat{u}]$, and the resulting ODE system is integrated with SciPy's adaptive RK45 (`solve_ivp`, `rtol` $= 10^{-6}$). Snapshots are stored on a uniform time grid of $\Delta t = 0.05\,\mathrm{s}$.

Single-soliton training set (in-distribution). Initial conditions are random single solitons,

$$u_0(x) \;=\; A\,\mathrm{sech}^2\big((x - x_0)/w\big), \qquad A \sim \mathcal{U}[0.5, 2.0], \quad w \sim \mathcal{U}[0.5, 2.0], \quad x_0 \sim \mathcal{U}[-15, 15]. \tag{14}$$

We generate 256 training and 120 validation trajectories of 200 steps each (10 s of physical time) for fitting all four backbones, and an additional 50 trajectories of 5000 steps for the long-horizon in-distribution evaluation.

Multi-soliton OOD set. The OOD test set is generated by the same solver but with initial conditions consisting of a uniformly random number of solitons in $\{1, 2, 3\}$, each with independently sampled amplitude, width and centre from the training distribution. The 50 OOD trajectories are 5000 steps long. Models trained only on single solitons are thus evaluated on collisions and overtakings they have never seen during training.

D.2 Architectures

All four backbones consume a single channel ($u(x, t_k)$) and produce a single channel ($u(x, t_{k+1})$) on the same 256-point periodic grid.

1D UNet. A circular-padded 1D UNet, $\sim$1.39 M parameters): base channel width 32, channel multipliers $(1, 2, 4, 8)$, depth 4, 3×1 kernels, GELU activations, no normalisation, residual two-conv blocks at every encoder/decoder level, strided 2×1 convolutions for down-sampling and transposed convolutions for up-sampling. Skip connections at every encoder level. The bottleneck has shape $(256, 16) = 4096$ dimensions and is the latent state on which the regulariser acts; the latent advance operator is $F(z, \text{skips}) = \text{encode}\big(\text{decode}(z, \text{skips})\big)$, so the regularised Jacobian naturally couples the bottleneck channels with the skip-feature channels.

Fourier Neural Operator (FNO). A four-block FNO, with 64 retained Fourier modes per layer, channel width 128, GELU activations, and a final two-layer pointwise projection. The first two blocks define the encoder ($z = \text{encode}(u)$ of shape $(64, 256) = 16384$ dims, no skip connections); the last two define the decoder. The latent advance operator differentiated by the regulariser is $F(z) = \text{encode}(\text{decode}(z))$.

U-FNO. The U-FNO backbone replaces the last two FNO blocks with U-FNO blocks that sum a spectral convolution, a small two-level 1D UNet on the same feature map, and a 1×1 pointwise convolution before activation. Channel width 128, 64 retained Fourier modes, GELU activations. The regulariser acts on the encoder/decoder split in the same way as for the plain FNO.

PDE-Refiner. We use the PDE-Refiner formulation [14] with a single denoising network conditioned on the previous state, the current (noisy) prediction, and a step index $k \in \{0, \ldots, M\}$. The backbone is identical to the UNet above; we use $M = 4$ refinement iterations per rollout step and the geometric noise schedule of [14] with $\sigma_{\min} = 10^{-7}$. PDE-Refiner therefore costs $M+1 = 5\times$ backbone evaluations per rollout step at inference; commutativity regularisation does not modify the rollout procedure and adds no inference-time cost.

D.3 Training and regulariser hyperparameters

Table 2: KdV training hyperparameters, identical between baseline and regularised regimes except for the loss term. PDE-Refiner uses the same optimiser settings as the UNet and adds the denoising loss schedule of [14].

Optimiser	AdamW
Peak learning rate	3×10^{-4}
Weight decay	10^{-5}
Schedule	cosine annealing to 10^{-7}
Epochs	500 (UNet, FNO, U-FNO) / 500 (PDE-Refiner)
Batch size	256
Training trajectory length	200 steps (10 s)
Loss (one-step)	$\|F_\theta(u_t) - u_{t+1}\|^2$ (MSE)
Regulariser	latent-space comm. + normality
λ_{c}	10^{-4}
λ_{n}	10^{-4}
JVP frequency	every 10th minibatch
Probe vector v	i.i.d. Gaussian, fresh per evaluation
Adjacent pair	$(z_t, z_{t+1}{=}G_\theta(z_t))$

Regulariser construction (same on every backbone). At a regulariser-active minibatch we draw a fresh sample u_t, encode it to obtain z_t, then decode-then-encode the pair (z_t, skips_t) to obtain the model's own one-step image $z_{t+1} = G_\theta(z_t)$ (and the corresponding skip activations for the UNet). The commutator and normality penalties are computed as

$$\mathcal{L}_{\mathrm{comm}} \;=\; \|[J_t, J_{t+1}]\, v\|^2, \qquad \mathcal{L}_{\mathrm{norm}} \;=\; \|(J_t^\top J_t - J_t J_t^\top)\, v\|^2,$$

on the latent state space, with $J_\bullet = \partial F/\partial z$ evaluated at the appropriate latent state, each Jacobian–vector product computed by a single `torch.func.jvp` call (and one `vjp` for the normality term). Subsampling the regulariser to every tenth minibatch keeps the training-time overhead within $\sim$10–20% of the baseline per epoch on the same hardware.

D.4 Full per-horizon results: UNet variants

Table 3: KdV final-step nMSE for the UNet variants on the single-soliton in-distribution test set (50 trajectories, mean over 3 seeds). Training horizon is 200 steps. Lower is better.

Step	50	100	200	500	1000	2000	3000	5000
UNet	2.9×10^{-4}	2.1×10^{-3}	2.1×10^{-2}	2.6×10^{-1}	3.8×10^{-1}	4.7×10^{-1}	6.7×10^{-1}	1.9
PDE-Refiner	2.2×10^{-4}	6.1×10^{-4}	1.5×10^{-3}	4.2×10^{-3}	1.2×10^{-2}	1.1×10^{-1}	2.1×10^{-1}	3.7×10^{-1}
UNet + comm. reg.	1.3×10^{-2}	1.8×10^{-2}	1.9×10^{-2}	2.0×10^{-2}	2.3×10^{-2}	3.5×10^{-2}	5.1×10^{-2}	6.9×10^{-2}

Table 4: KdV final-step nMSE for the UNet variants on the multi-soliton out-of-distribution test set (50 trajectories with up to three solitons, mean over 3 seeds). Training horizon is 200 steps. Lower is better.

Step	50	100	200	500	1000	2000	3000	5000
UNet	2.0×10^{-4}	2.2×10^{-3}	2.1×10^{-2}	2.6×10^{-1}	4.0×10^{-1}	4.9×10^{-1}	7.3×10^{-1}	1.8
PDE-Refiner	3.8×10^{-3}	5.8×10^{-3}	1.1×10^{-2}	3.4×10^{-2}	6.6×10^{-2}	1.1×10^{-1}	2.0×10^{-1}	3.8×10^{-1}
UNet + comm. reg.	1.6×10^{-2}	2.1×10^{-2}	2.1×10^{-2}	2.3×10^{-2}	2.5×10^{-2}	3.6×10^{-2}	4.7×10^{-2}	6.1×10^{-2}

The UNet baseline is the more accurate model at the very first rollout steps (≤ 50) but is overtaken by both alternatives by step $\sim$100 and is more than an order of magnitude worse than UNet+CR already inside the training window (step 200). PDE-Refiner is the strongest *unregularised* model up to step $\sim$1000, paying $5\times$ inference-time cost; from step $\sim$2000 onwards UNet+CR overtakes it on both the in-distribution and the out-of-distribution split. The relative gain is largest in the $25\times$-extrapolation regime, where the unregularised UNet rollout has lost coherence and PDE-Refiner has accumulated enough error to also degrade.

D.5 Full per-horizon results: spectral backbones

Table 5: KdV final-step nMSE for the spectral backbones on the in-distribution single-soliton test set (120 trajectories, mean over seeds: FNO and FNO+CR three seeds, U-FNO+CR three seeds, U-FNO baseline two seeds). Training horizon is 200 steps; the rightmost four columns are extrapolation in rollout length. Lower is better.

Step	50	100	200	500	1000	1500	2000
FNO	6.3×10^{-4}	7.6×10^{-2}	3.1×10^{-1}	3.9×10^{-1}	4.9×10^{-1}	5.0×10^{-1}	5.0×10^{-1}
U-FNO	2.1×10^{-4}	6.1×10^{-4}	1.5×10^{-3}	7.2×10^{-3}	1.7×10^{-1}	1.6×10^{-1}	1.7×10^{-1}
FNO + comm. reg.	2.1×10^{-4}	6.1×10^{-4}	1.5×10^{-3}	7.3×10^{-3}	1.9×10^{-2}	3.4×10^{-2}	5.1×10^{-2}
U-FNO + comm. reg.	2.1×10^{-4}	6.2×10^{-4}	1.5×10^{-3}	4.4×10^{-3}	1.3×10^{-2}	2.9×10^{-2}	4.6×10^{-2}

Why the unregularised FNO collapses. At the end of training the baseline FNO has a one-step validation MSE of $\sim1.5\times10^{-7}$ and the regularised FNO $\sim6.9\times10^{-7}$, i.e. the baseline is $\sim4\times$ *better* per step. Yet the baseline's autoregressive rollout amplifies error super-exponentially and reaches nMSE ~0.3 by step 200, while FNO+CR maintains nMSE $\sim1.5\times10^{-3}$ at the same horizon. The U-FNO baseline is more stable than the plain FNO baseline because the embedded multi-scale UNet path injects local mixing that breaks the otherwise diagonal Fourier-block structure, but it still collapses around step ~1000. In both cases the regulariser converts a divergent rollout into a bounded one, and the per-step accuracy difference between regularised and unregularised is in the third or fourth significant digit.

D.6 In-distribution Per-trajectory snapshots

See Figure 12.

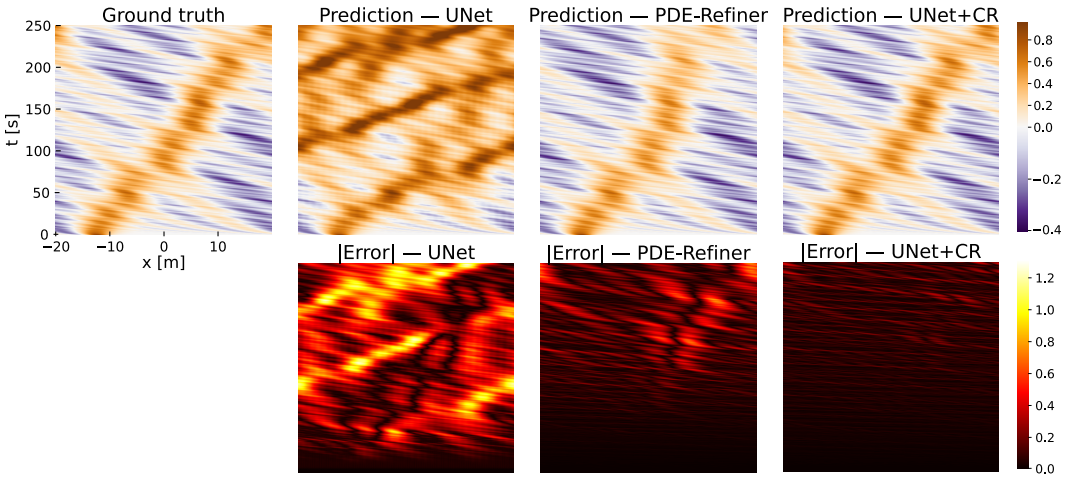


Figure 12: KdV space–time plots of $u(x,t)$ for representative in-distribution test trajectories: ground truth, baseline rollout, and commutativity-regularised rollout, run for the full 5000 steps (UNet variants) from a single initial condition. Color scale is symmetric and shared per trajectory.

D.7 Out-of-distribution Per-trajectory snapshots

See Figure 13.

E BVE experiment: details

This appendix expands the BVE experiment of Section 4.2: data generation, the 2D UNet architecture, training and regulariser hyperparameters, evaluation, the full per-horizon RMSE numbers underlying Figure 7 and Table 8, and per-trajectory snapshots.

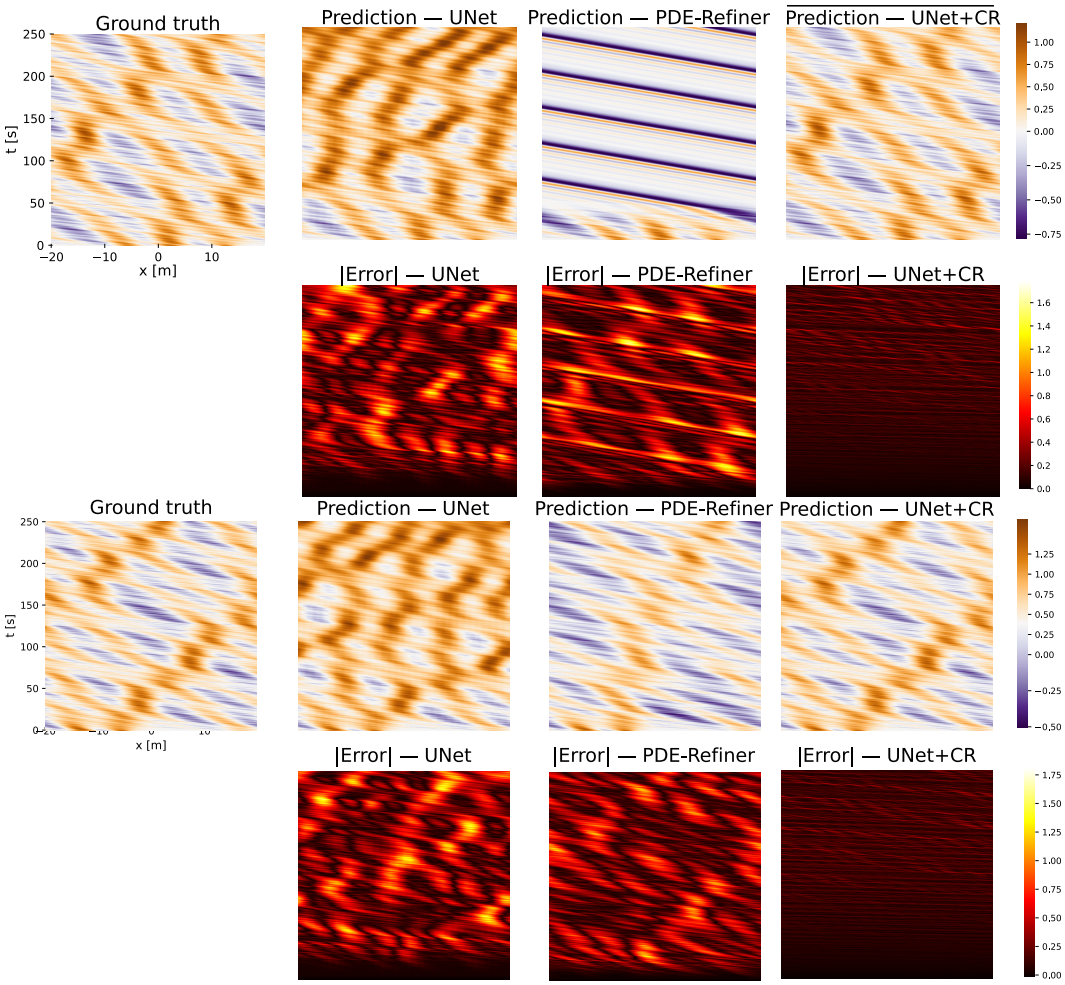


Figure 13: KdV space–time plots of $u(x, t)$ for representative OOD test trajectories: ground truth, baseline rollout, and commutativity-regularised rollout, run for the full 5000 steps (UNet variants).

E.1 Data generation

Equation and solver. We integrate the 2D barotropic vorticity equation

$$\partial_t \zeta + J(\psi,\, \zeta + \beta y) \;=\; -\nu(-\nabla^2)^2\zeta - r\zeta, \qquad \zeta = \nabla^2\psi, \tag{15}$$

on the doubly-periodic domain $[0, 2\pi]^2$ with $N{=}64$ grid points per side. The solver is pseudospectral in space with the $2/3$-rule dealiasing mask and RK4 in time. The Jacobian $J(\psi, q) = \partial_x\psi\,\partial_y q - \partial_y\psi\,\partial_x q$ is evaluated in real space; all linear operations (Laplacian inversion, derivatives, hyperviscosity) are evaluated in spectral space. Physical and numerical parameters are summarised in Table 6.

Initial conditions. Initial conditions are random Gaussian fields with prescribed energy spectrum

$$E(k) \;\propto\; k^4 \exp\!\big(-2\,(k/k_{\text{peak}})^2\big), \qquad k_{\text{peak}} = 6, \tag{16}$$

sampled in spectral space with i.i.d. uniform phases on $[0, 2\pi)$, then rescaled in real space so that the RMS vorticity is 1.5. Each trajectory is integrated through a 2 s spin-up that is discarded, after which 200 snapshots are recorded at $\Delta t_{\text{out}}{=}0.05$ s, giving a 10 s recording window.

Splits and normalisation. The dataset contains 300 trajectories, split deterministically into 240 / 30 / 30 for train / val / test. Vorticity fields are standardised to zero mean and unit standard deviation using the train-split statistics before being passed through the network; predictions are evaluated in

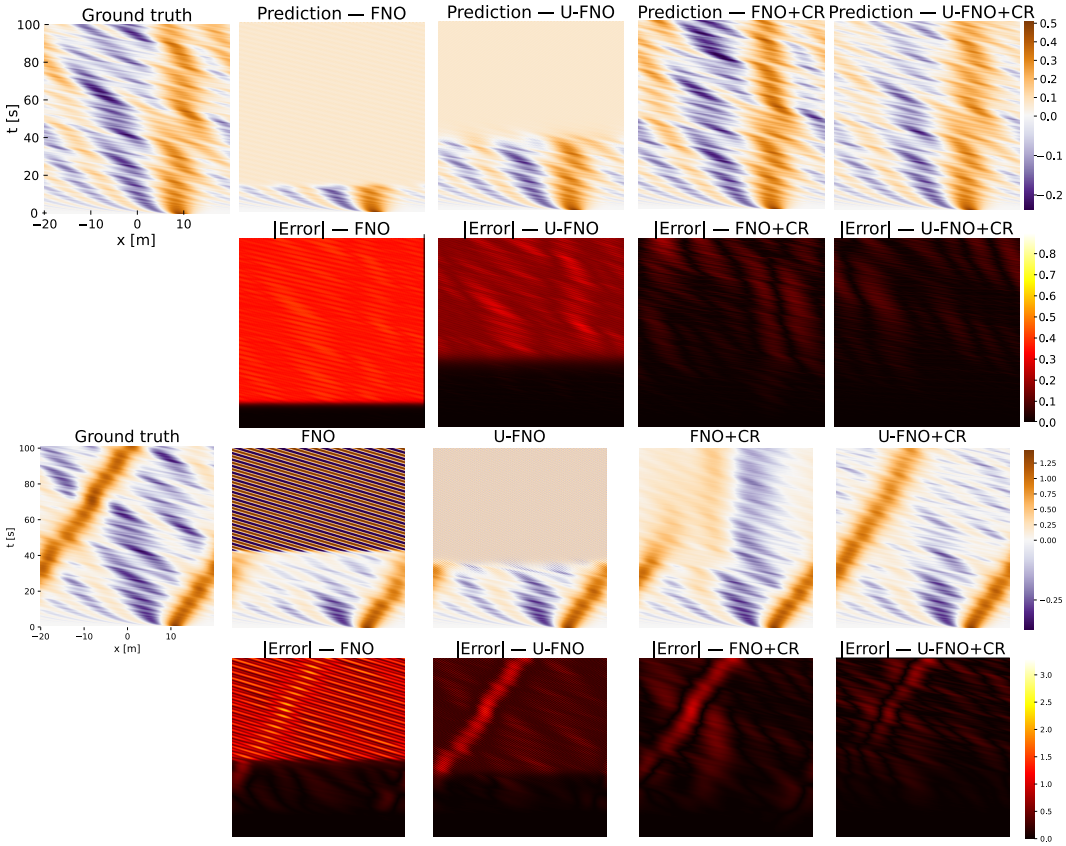


Figure 14: KdV space–time plots of $u(x, t)$ for representative test trajectories: ground truth, baseline rollout, and commutativity-regularised rollout, run for the full 2000 steps (FNO variants).

Table 6: BVE solver parameters.

Domain	$[0, 2\pi]^2$, doubly periodic
Grid	$N \times N = 64 \times 64$
β (planetary vorticity gradient)	1.0
Hyperviscosity	$\nu = 10^{-8}$, biharmonic order $p = 2$
Ekman drag r	10^{-2}
Time step Δt	5×10^{-4}
Save stride	every 100 steps; $\Delta t_{\text{out}} = 0.05$ s
Spin-up	2.0 s (discarded)
Recording window	10 s (200 snapshots/trajectory)
Dealiasing	2/3 rule
Time integrator	RK4

this normalised space. All RMSE numbers in the main text and this appendix are in these normalised vorticity units.

E.2 Architecture

The 2D UNet advances the field one step in time, $\zeta_t \mapsto \zeta_{t+1}$, with the following layout:

- **Encoder.** Three residual conv blocks with channel widths $(64, 128, 256)$. Each block is two 3×3 circular convolutions wrapped in `Conv` $\rightarrow$ `GroupNorm`$(\min(8, C))$ $\rightarrow$ `GELU`, with a 1×1 skip when the channel count changes. Each level is followed by 2×2 average pooling, producing skip activations at resolutions $64, 32, 16$.

- **Bottleneck.** A residual conv block $\mathrm{ConvBlock}(256 \to 256)$ at the $8 \times 8 \times 256 = 16384$-dimensional latent state.
- **Decoder.** Two transposed 2×2 convolutions interleaved with residual conv blocks consuming the corresponding encoder skip activations at resolutions 16 and 32. A bilinear interpolation guards against odd-dimension rounding before concatenation.
- **Head.** A 1×1 convolution to a single output channel.

The latent state used by the regulariser is the output of `encode(ζ)` after all encoder downsamplings and the bottleneck residual block, and the latent advance operator is $F(z, \mathrm{skips}) = \mathrm{encode}\big(\mathrm{decode}(z, \mathrm{skips})\big)$, exactly as in the KdV UNet of Section 4.1. The Jacobian penalised by the regulariser is therefore the Jacobian of this encode$\circ$decode map at the bottleneck, naturally coupling bottleneck channels with the skip-feature channels.

E.3 Training and regulariser hyperparameters

Table 7: BVE training hyperparameters, identical between baseline and regularised regimes except for the loss term.

Optimiser	AdamW
Peak learning rate	10^{-4}
Weight decay	10^{-5}
Schedule	cosine annealing to 10^{-7}
Epochs	500
Batch size	128
Training trajectory length	200 frames (10 s)
Loss (one-step)	$\|F_\theta(\zeta_t) - \zeta_{t+1}\|^2$ (MSE)
Checkpoint selection	best validation MSE
Regulariser	latent-space comm. + normality
λ_c	10^{-7}
λ_n	10^{-7}
JVP frequency	every 15th minibatch
Regulariser sub-batch	first 25 samples of the minibatch
Regulariser pair	in-trajectory adjacent step $(\zeta_{t'}, \zeta_{t'+1})$, t' uniform per sample
Probe vector v	i.i.d. Gaussian, fresh per evaluation

Construction of the in-trajectory pair. Each minibatch sample carries, in addition to the supervised pair (ζ_t, ζ_{t+1}), an independently drawn second adjacent pair $(\zeta_{t'}, \zeta_{t'+1})$ from the *same* trajectory with t' uniform over the recorded frames. The regulariser encodes both endpoints of this partner pair, $z_{t'} = \mathrm{encode}(\zeta_{t'})$ and $z_{t'+1} = \mathrm{encode}(\zeta_{t'+1})$, and computes

$$\mathcal{L}_\mathrm{c} \;=\; \big\|[J_{t'},\, J_{t'+1}]\, v\big\|^2, \qquad \mathcal{L}_\mathrm{n} \;=\; \big\|(J_{t'}^\top J_{t'} - J_{t'} J_{t'}^\top)\, v\big\|^2, \tag{17}$$

on the latent state space, with $J_\bullet = \partial F/\partial z\,|_{(z_\bullet, \mathrm{skips}_\bullet)}$ and v a fresh Gaussian probe vector. Each Jacobian–vector product is computed by a single `torch.func.jvp` call (and one `vjp` for the normality term). The total loss is $\mathcal{L} = \mathrm{MSE} + \lambda_\mathrm{c}\mathcal{L}_\mathrm{c} + \lambda_\mathrm{n}\mathcal{L}_\mathrm{n}$ with $\lambda_\mathrm{c} = \lambda_\mathrm{n} = 10^{-7}$. The regulariser is applied on the first 25 samples of every 15th minibatch, which keeps the training-time overhead small relative to the baseline per epoch while still providing a fresh in-trajectory pair on each application.

E.4 Evaluation Metric

The reported metric is the per-step spatial RMSE on the normalised vorticity field,

$$\mathrm{RMSE}(t) \;=\; \sqrt{\tfrac{1}{N^2}\sum_{i,j}(\hat{\zeta}_{t,ij} - \zeta_{t,ij})^2},$$

averaged over the 30 test trajectories per seed.

E.5 Full per-horizon results

Table 8 reports the per-step RMSE at every 20 rollout steps on the same 30 test trajectories, together with the across-seed standard deviation. Two features are visible across the entire range. First, the baseline crosses RMSE 1 around step 33 ($t \approx 1.65$ s, well within the training horizon) and saturates near RMSE $\approx$ 5–6 from step $\sim$140 onward; the regularised model stays bounded throughout. Second, the across-seed dispersion (std at step 199) is 1.90 for the baseline ($\sigma/\mu \approx 0.32$) and 0.009 for the regularised model ($\sigma/\mu \approx 0.013$); the regulariser not only improves the mean rollout but also collapses the seed spread, suggesting that the unconstrained Jacobian product dominates the seed-to-seed variability of the baseline rollout while the regularised rollout is essentially a structural property of the data that all seeds converge to.

Table 8: BVE per-step RMSE on the 30-trajectory test split at every 20 rollout steps (mean over 3 training seeds, normalised vorticity units). $\Delta t_{\text{out}} = 0.05$ s.

Step	1	20	40	60	80	100	120	140	160	199
Time (s)	0.05	1	2	3	4	5	6	7	8	9.95
Baseline (mean)	0.0040	0.272	1.582	2.810	3.620	4.266	4.890	5.391	5.721	6.007
Baseline (std)	0.0001	0.027	0.242	0.381	0.332	0.282	0.695	1.187	1.540	1.896
Comm. reg. (mean)	0.0028	0.0512	0.0962	0.140	0.186	0.239	0.302	0.377	0.465	0.665
Comm. reg. (std)	0.00003	0.0007	0.0014	0.002	0.003	0.004	0.005	0.006	0.007	0.009
Ratio (mean)	1.4	5.3	16.4	20.1	19.5	17.8	16.2	14.3	12.3	9.0

E.6 Per-trajectory snapshots

See Figure 15

F FCN/ERA5 experiment: details

This appendix expands the FCN experiment of Section 4.3: data and preprocessing, the training/eval protocol, the regulariser implementation in the AFNO latent stack, and the full per-channel, per-lead RMSE numbers underlying Figure 9 in Tables 10 and 11.

F.1 Data and preprocessing

Source. ERA5 reanalysis on FCN's native 20-channel state at 721×1440 ($0.25°$, $\Delta t = 6$ h): 10 m winds (u_{10}, v_{10}), 2 m temperature $t_{2\mathrm{m}}$, surface and mean-sea-level pressures $(\mathrm{sp}, \mathrm{msl})$, T at 850 hPa, winds at 1000/850/500 hPa, geopotential at 1000/850/500/50 hPa, relative humidity at 500/850 hPa, T at 500 hPa, and total column water vapour tcwv. We use FCN's per-variable normalisation statistics throughout, so the network is run in its native input space; metric values are reported in physical units after the inverse normalisation.

Years. Pretraining of FCN v1 covers 1979–2015, so all fine-tuning years used here lie inside its training range and expose the model to no new physical signal:

- FCN+CR [2013–15]: fine-tuned on $\{2013, 2014, 2015\}$.
- Plain-finetune baseline: fine-tuned on $\{2013, 2014, 2015\}$.

Validation uses 2017 during fine-tuning. The reported test years are 2018 (Section 4.3) and 2019 (held-out replication, Appendix F.4); both are genuinely outside FCN v1's training range.

Evaluation initial conditions. For each test year we sample 41 initial conditions at a fixed stride (every 9 h, stride=36 on the 6 h steps) and run an autoregressive 40-step rollout (10 days) per IC. The reported metric is the mean over ICs of the latitude-weighted RMSE,

$$\mathrm{RMSE}(t) \;=\; \sqrt{\tfrac{1}{HW}\sum_{i,j} w_i \left(\hat{X}_{t,ij} - X_{t,ij}\right)^2}, \qquad w_i = \cos(\phi_i),$$

on each of the channels z_{500}, t_{850}, u_{850}, $t_{2\mathrm{m}}$. Numbers are reported in physical units after inverse-normalising with the FCN per-channel statistics.

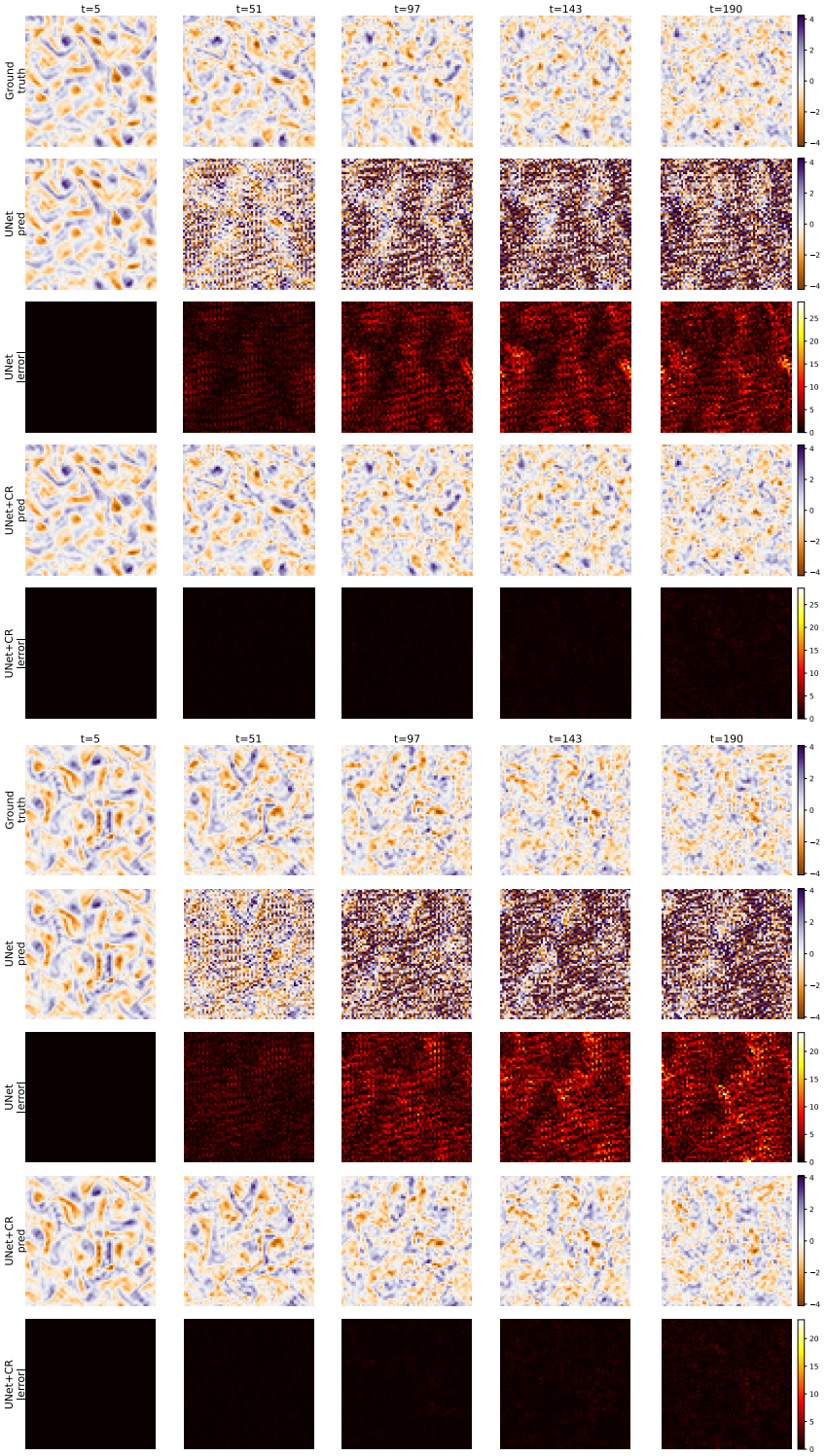


Figure 15: Vorticity snapshots $\zeta(x, y, t)$ for representative test trajectories: ground truth, baseline rollout, and commutativity-regularised rollout, all run for the full 199 rollout steps ($\sim 10\,\mathrm{s}$) from a single initial condition. Color scale is symmetric and shared between truth and predictions per trajectory; absolute error uses a separate scale. The baseline visibly amplifies vorticity to several times the natural scale of the flow well before the end of the training horizon, while the regularised model preserves coherent structures and the correct amplitude band over the full rollout.

F.2 Architecture and where the regulariser acts

The backbone is the released FCN v1 AFNO with 74.7 M total parameters. Fine-tuning unfreezes the 34.5 M parameters of the deep AFNO blocks and the head; the patch-embed and the first several blocks are kept frozen and run once per minibatch with their output detached.

Latent space operator. The latent state on which the regulariser acts is the activation *after* the first $K{=}10$ AFNO blocks; the regulariser flows JVP / VJP only through the remaining (deep) blocks, so the Jacobian–vector products it touches have memory cost $\propto (\text{depth} - K)$ rather than full depth. This is necessary at $720{\times}1440$ resolution: fully differentiating the network would otherwise blow GPU memory at batch size ≥ 1, even at low precision.

F.3 Training and regulariser hyperparameters

Table 9: FCN fine-tuning hyperparameters (identical across the two fine-tune regimes except for the loss term).

Backbone	FCN v1 AFNO, pretrained [20]
Optimiser	AdamW
Peak learning rate	5×10^{-6}
Weight decay	0
Schedule	cosine annealing
Epochs	50
Batch size	2 (single-GPU, A100/H100 80 GB)
Loss (one-step)	latitude-weighted MSE between $\hat{X}_{t+1}$ and ERA5
Regulariser	latent comm. + normality
λ_{c}	10^{-5}
λ_{n}	10^{-5}
JVP frequency	every minibatch (`comm_freq`=1)
Skip blocks	first 10 AFNO blocks detached (`comm_skip_blocks`=10)
Comm. pair	adjacent-step pair (X_t, X_{t+1}) (`comm_window`=1)
Probe vector v	i.i.d. Rademacher, fresh per call

Construction of the JVP. At each (regulariser-active) minibatch we draw the supervised pair (X_t, X_{t+1}) and compute, in the post-skip latent space,

$$\mathcal{L}_{\mathrm{c}} \;=\; \big\|[J(z_t),\, J(z_{t+1})]\, v\big\|^2, \qquad \mathcal{L}_{\mathrm{n}} \;=\; \big\|J(z_t)^\top J(z_t)\, v - J(z_t) J(z_t)^\top\, v\big\|^2,$$

where J is the Jacobian of the deep-block stack at the indicated latent state, v is a fresh Rademacher probe, and the JVP/VJP are evaluated on the first two samples of the minibatch (memory bound). Three JVPs and one VJP are needed per call; $J(z_t)\, v$ is reused between the commutator and normality terms. The total fine-tune loss is $\mathcal{L} = \mathcal{L}_{\mathrm{MSE}} + \lambda_{\mathrm{c}}\, \mathcal{L}_{\mathrm{c}} + \lambda_{\mathrm{n}}\, \mathcal{L}_{\mathrm{n}}$ with $\lambda_{\mathrm{c}} = \lambda_{\mathrm{n}} = 10^{-5}$.

F.4 Full per-channel, per-lead RMSE numbers

Table 10 (test year 2018) and Table 11 (held-out year 2019) give the latitude-weighted RMSE in physical units at every day from 1 to 10 on all four WB2 headline channels.

F.5 Qualitative forecasts

See Figure 16 for ERA5 t2m forecasts.

G SST: details

This appendix expands the SST experiment summarised in Section 4.4: data and preprocessing, the full UNet architecture, training hyperparameters, the regulariser implementation, and the per-step rollout numbers and qualitative snapshots underlying Figure 10.

Table 10: ERA5 2018 rollout RMSE (latitude-weighted, physical units) at daily lead times for the four variables, averaged over 41 initial conditions (9 h stride). **Bold** marks the best entry per column. The FCN+CR entries here use the [2013–15] fine-tune.

Channel	Method	1 d	2 d	3 d	4 d	5 d	6 d	7 d	8 d	9 d	10 d
z_{500} [m^2/s^2]	Frozen	86.9	165.0	258.3	374.5	494.5	608.8	**704.4**	785.0	**854.3**	**903.7**
	Finetune	**81.7**	**162.7**	260.8	381.1	503.0	615.5	710.9	792.9	871.7	930.0
	FCN+CR [2013-2015]	82.7	164.2	264.0	385.0	510.3	622.6	712.5	**790.0**	855.3	908.4
t_{850} [K]	Frozen	0.86	**1.19**	**1.59**	**2.06**	**2.52**	**2.98**	**3.36**	**3.61**	3.88	4.26
	Finetune	**0.84**	**1.19**	1.62	2.10	2.57	3.02	3.38	3.67	4.10	5.18
	FCN+CR [2013-2015]	0.86	1.22	1.65	2.13	2.60	3.07	3.42	3.68	**3.91**	**4.14**
u_{850} [m/s]	Frozen	**1.60**	**2.43**	**3.29**	**4.15**	**4.93**	**5.60**	**6.06**	**6.46**	**6.80**	**7.01**
	Finetune	1.61	2.45	3.34	4.23	5.01	5.67	6.15	6.53	6.88	7.20
	FCN+CR [2013-2015]	1.62	2.48	3.39	4.28	5.05	5.73	6.21	6.56	6.90	7.08
t_{2m} [K]	Frozen	0.97	1.18	**1.42**	**1.72**	**2.03**	**2.34**	2.56	3.00	4.32	7.53
	Finetune	**0.95**	**1.18**	1.43	1.74	2.05	2.36	2.66	3.97	7.62	16.16
	FCN+CR [2013-2015]	0.97	1.21	1.47	1.79	2.10	2.39	**2.62**	**2.87**	**3.19**	**4.48**

Table 11: ERA5 2019 replication: rollout RMSE (latitude-weighted, physical units) at daily lead times. **Bold** marks the best entry per column.

Channel	Method	1 d	2 d	3 d	4 d	5 d	6 d	7 d	8 d	9 d	10 d
z_{500} [m^2/s^2]	Frozen	87.9	**166.1**	**263.2**	**369.9**	**481.9**	**595.3**	**697.7**	**788.2**	**861.0**	**908.9**
	Finetune [2013–15]	**84.1**	164.9	263.4	371.2	486.9	602.8	710.5	797.0	868.2	920.1
	FCN+CR [2013–15]	84.7	166.2	267.0	377.7	491.3	607.0	714.6	803.6	872.6	920.0
t_{850} [K]	Frozen	**0.87**	**1.20**	**1.62**	**2.05**	**2.49**	**2.91**	**3.28**	**3.65**	**3.93**	**4.23**
	Finetune [2013–15]	0.86	1.21	1.63	2.07	2.52	2.94	3.32	3.67	4.10	5.02
	FCN+CR [2013–15]	**0.87**	1.23	1.66	2.11	2.57	2.99	3.37	3.73	4.00	4.24
u_{850} [m/s]	Frozen	**1.62**	**2.43**	**3.35**	**4.15**	**4.90**	**5.55**	**6.03**	**6.46**	**6.78**	**6.95**
	Finetune [2013–15]	1.63	2.45	3.38	4.19	4.95	5.62	6.16	6.53	6.84	7.10
	FCN+CR [2013–15]	1.64	2.48	3.41	4.23	4.98	5.65	6.21	6.62	6.91	7.12
t_{2m} [K]	Frozen	0.97	**1.18**	**1.42**	**1.69**	**1.99**	**2.28**	**2.55**	**2.81**	3.58	5.54
	Finetune [2013–15]	**0.95**	1.18	1.43	1.71	2.01	2.28	2.61	3.73	6.96	14.43
	FCN+CR [2013–15]	0.97	1.20	1.47	1.75	2.06	2.34	2.63	2.87	**3.33**	**3.99**

G.1 Data and preprocessing

Source. NOAA Optimum Interpolation Sea-Surface Temperature, weekly-mean product (`sst.wkmean.1990-present`) [24], covering the global ocean on a 1° (180×360) grid at weekly cadence. The version we use contains 1727 weeks beginning in 1990. Land pixels are filled with 0 in the raw NetCDF and are masked only in the visualisations (Appendix G.6); the network sees them as zero pixels.

Padding. The network requires spatial dimensions divisible by $2^4 = 16$. We pad each frame from 180×360 to 192×384 using boundary-aware padding: 6 rows of *reflect* padding at each pole (no real wrap-around at the poles), and 12 columns of *wrap* (periodic) padding at the 0°/360° longitude seam. This avoids the mid-Pacific padding artefacts that a default zero or reflect padding would introduce and matches the natural periodicity of the longitude axis.

Splits and normalisation. We split chronologically into 1200 / 150 / 377 weeks of train / val / test. The test split therefore covers ≈ 7.24 years and includes ~ 7 full annual cycles. Frames are standardised to zero mean and unit standard deviation using train-set statistics; all RMSE numbers in Section 4.4 and below are reported in these normalised units.

G.2 Architecture

The backbone is a 2D UNet with circular-aware inputs (the wrap padding above provides the longitude periodicity) operating on the full 192×384 field.

Encoder/decoder structure. Depth $D = 4$, base channel width $C_0 = 64$, so the encoder produces channel widths $(64, 128, 256, 512)$ at successive levels. Each encoder block is a residual two-conv

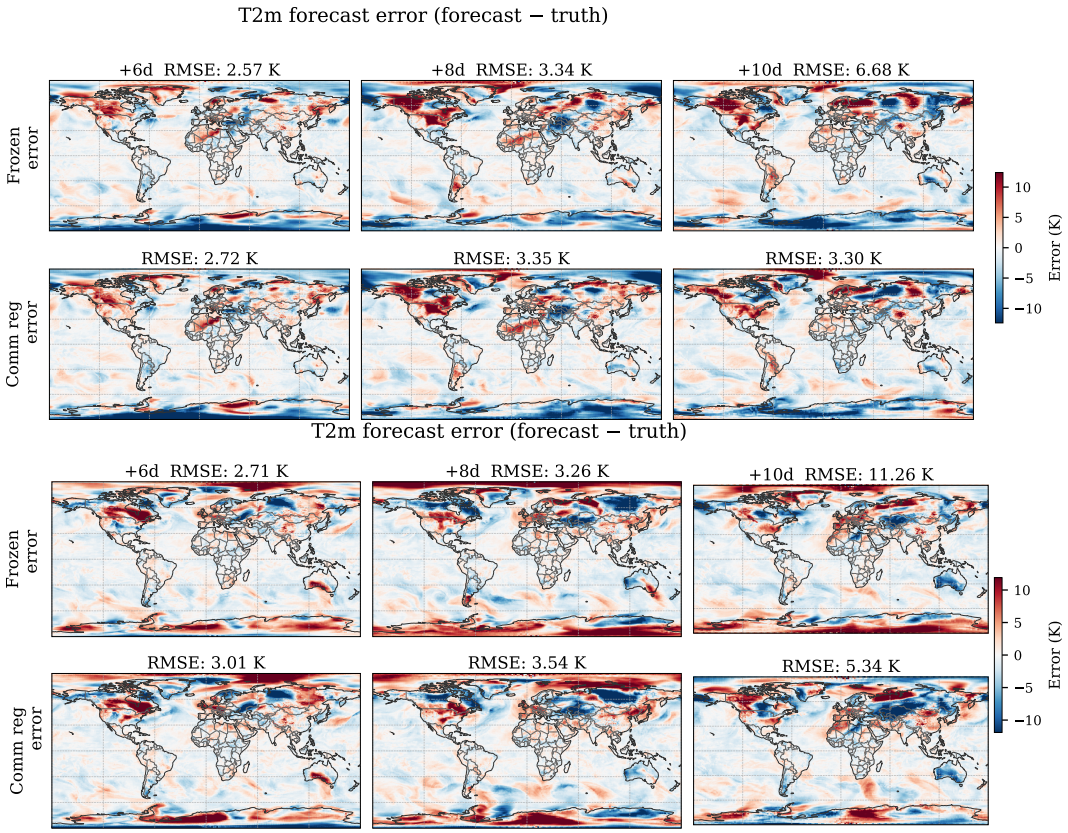


Figure 16: Qualitative day-6, 8 and 10 rollout error snapshots of $t_{2\mathrm{m}}$ on a single ERA5 2018 initial condition for 2m temperature.

block $\texttt{Conv3}\times3 \rightarrow \texttt{GroupNorm}(\min(8, C)) \rightarrow \texttt{GELU}$ applied twice with a 1×1 skip when the channel count changes, followed by 2×2 average pooling. The decoder mirrors the encoder with 2×2 transposed convolutions and a residual block taking the concatenated skip features. The bottleneck spatial size is therefore $192/16 \times 384/16 = 12\times24$ at 512 channels, the latent state on which the regulariser acts. A final 1×1 convolution maps back to one output channel (the next-week SST). The model has $\sim$17 M parameters.

Inputs and outputs. $k_{\text{in}} = 1$: the network sees one previous week and predicts the next. We did not find a benefit from longer context windows on this data.

G.3 Training and regulariser

Optimiser and schedule. AdamW with learning rate 10^{-4}, weight decay 10^{-5}, batch size 4, 200 epochs. The learning rate follows a cosine annealing schedule with $\eta_{\min} = 10^{-7}$. Loss is one-step MSE on the next-week SST: the model never sees a multi-step rollout during training.

Commutativity regulariser. The regulariser is identical in form to the BVE one (Appendix E.3) but applied to the 12×24 bottleneck of the SST UNet. At minibatch b with input x and an in-trajectory random partner $\tilde{x}$, the latent advance operator $F(z, s) = \mathrm{encode}(\mathrm{decode}(z, s))$ is differentiated through `torch.func.jvp` on the bottleneck features and skip stack, and the squared norms

$$\mathcal{L}_{\mathrm{c}}(x, \tilde{x}) = \left\| [J_F(z(x)),\, J_F(z(\tilde{x}))]\, v \right\|^2, \tag{18}$$

$$\mathcal{L}_{\mathrm{n}}(x) = \left\| J_F(z(x))^\top J_F(z(x))\, v - J_F(z(x)) J_F(z(x))^\top v \right\|^2, \tag{19}$$

are estimated on a single random Gaussian probe v. The total loss is $\mathcal{L} = \mathrm{MSE} + \lambda_{\mathrm{c}}\mathcal{L}_{\mathrm{c}} + \lambda_{\mathrm{n}}\mathcal{L}_{\mathrm{n}}$ with $\lambda_{\mathrm{c}} = \lambda_{\mathrm{n}} = 10^{-7}$, evaluated every fifth minibatch and on the first 5 samples of that minibatch (the exact JVP at full batch and full 192×384 resolution is GPU-memory bound).

Table 12: SST rollout spatial RMSE on the normalised test field at selected rollout steps (single rollout, 376 weeks). Lower is better; **bold** marks the better entry per column. The two final columns are summary statistics over the entire 376-step rollout.

Method	1	13	26	52	104	156	260	376	mean	max
Lead	1 wk	3 mo	6 mo	1 yr	2 yr	3 yr	5 yr	7.2 yr	all	all
Baseline	**0.025**	0.059	0.120	0.160	0.250	0.277	0.193	0.148	0.248	0.516
Commutativity reg.	**0.025**	**0.046**	**0.050**	**0.062**	**0.055**	**0.066**	**0.097**	**0.114**	**0.082**	**0.162**

G.4 Evaluation protocol

We evaluate a single autoregressive rollout starting from the first frame of the test split and continuing for $T_{\text{eval}} = 376$ weeks (the test length minus the input context). At each step the network produces the next normalised SST field, which is then fed back as the next input. The reported metric is the per-step spatial RMSE

$$\text{RMSE}(t) \;=\; \sqrt{\tfrac{1}{HW}\sum_{i,j}\big(\hat{T}_{t,ij} - T_{t,ij}\big)^2},$$

on the normalised field, computed over the entire padded 192×384 grid (including the masked land pixels, whose target value is 0 in the dataset). Reporting on the ocean-pixels-only mask gives values that are uniformly $\sim 5\%$ larger but does not change the relative comparison.

G.5 Per-step RMSE numbers

Table 12 reports per-step RMSE at a denser set of lead times than Section 4.4. The qualitative behaviour, baseline error oscillating with the annual cycle and growing in envelope, regularised error phase-locked and bounded, is visible at every horizon beyond $\sim$3 months.

G.6 Qualitative snapshots

Visualizations of sea surface temperature predictions and errors are shown in Figure 17.

H Compute Requirements

All experiments in the paper were run on a single GPU on an internal cluster. The PDE experiments were run on a L40S GPU with 48GB memory. The FourCastNet finetuning was done on an H100 GPU for up to 2 days. The PDE experiments take $\sim$ 1 to 5 hours at most per run for 1D and 2D respectively. There is no distributed training involved.

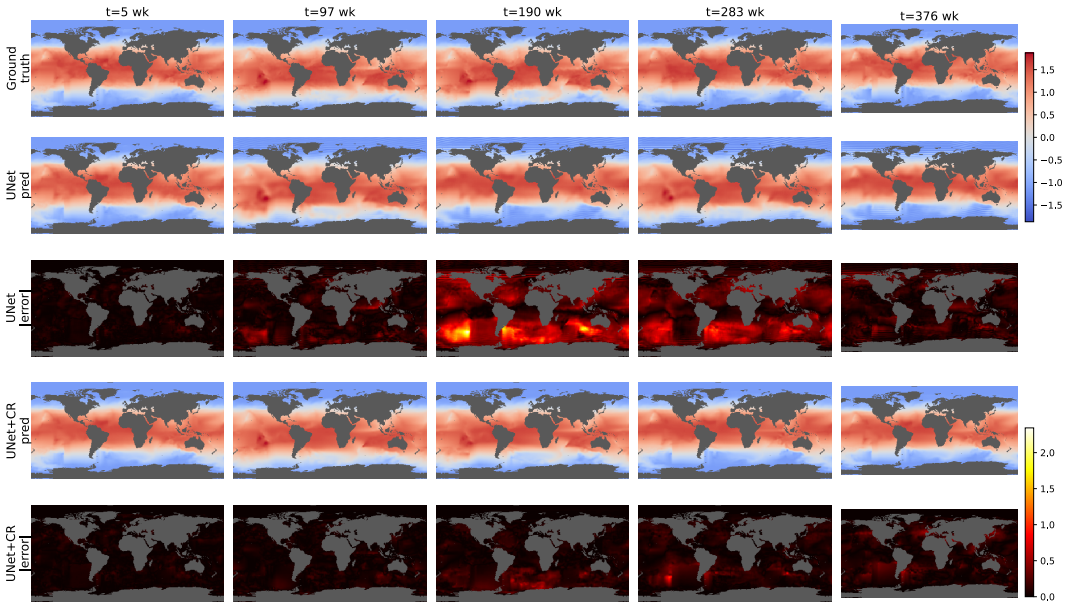


Figure 17: Spatial snapshots of the SST autoregressive rollout at selected lead times for the baseline and the commutativity-regularised UNet, with ground-truth SST in the top row and absolute error in the lower rows. Land is overlaid in grey. The baseline progressively blurs mid-latitude anomalies and drifts in the seasonal phase; the regularised rollout retains the large-scale structure of the true field through the full 7.2-year window.

I Other Experiments

I.1 Loss Ablation Experiments

We compare long horizon performance of commutativity regularization with normality, normality-only regularization and commutativity-only regularization on the KdV dataset with the UNet and FNO model. We compare MSE for up to 10,0000 steps for UNet and 5000 steps for FNO. We retrained all models from scratch. Results are in Tables 13 and 14.

We find that commutativity-only regularization (i.e., without normality) diverges on long rollouts on UNet. Normality-only regularization remains stable throught the horizon for UNet. On FNO both settle into a steady state with giving large errors. Taken together they improve both models on long rollouts.

Table 13: Loss ablation on KdV UNet

Rollout Step	500	1000	2000	4000	5000
Comm. Only	2.97	inf	inf	inf	inf
Normality Only	6e-3	2e-2	6e-2	8e-2	2e-1
Comm. Reg.	7e-3	1e-2	3e-2	5e-2	9e-2

Table 14: Loss ablation on KdV FNO

Rollout Step	500	1000	2000	4000	5000
Comm. Only	1e-02	4e-02	0.11	0.11	0.10
Normality Only	1e-02	7e-02	0.22	0.39	0.37
Comm. Reg.	5e-03	1e-02	3e-2	6e-2	7e-2

I.2 Non-adjacent Commutators

We experiment with a variant of the loss when the commutators are computed at non-adjacent steps. Tables 15 and 16 show commutativity defects between non-adjacent commutators for fully trained models for KdV and BVE UNet models for various gaps. The results show that the commutators are reduced, relative to the baseline, even when only adjacent commutators are explicitly optimized.

Table 15: Non-adjacent commutators on the KdV

Gap	1	5	10	15	20	50
Baseline	3.8e-02	5.2e-02	5.7e-02	6.0e-02	6.8e-02	1.0e-01
Regularized	4.4e-05	5.6e-05	5.6e-05	8.1e-05	9.8e-05	2.4e-04

Table 16: Non-adjacent commutators on the BVE

Gap	1	5	10	15	20	50
Baseline	2.4e+01	5.4e+02	1.6e+03	2.5e+03	3.1e+03	4.8e+03
Regularized	4.2e-01	9.8e+00	3.16e+01	5.2e+01	6.6e+01	1.0e+02

I.3 Spectral Baselines

We compare with spectral regularization methods [16], [18] applied to the weight matrix and a variant applied to the operator Jacobian. We trained the baselines and regularized models from scratch on the KdV dataset with the UNet and FNO models and compare long horizon accuracy. The resuls are in Tables 17 and 18.

Table 17: Spectral Baselines on KdV with UNet

Rollout Step	500	1000	2000	3000	5000
Spectral Weight Reg.	1e-1	2e-1	2e-1	2e-1	2e-1
Spectral Jac Reg.	4e-3	inf	inf	inf	inf
Comm. Reg.	4e-3	7e-3	1e-2	2e-2	4e-2

Table 18: Spectral Baselines on KdV with FNO

Rollout Step	500	1000	2000	3000	5000
Spectral Weight Reg.	1e-02	6e-02	1e-01	1e-01	0.12
Spectral Jac Reg.	1e-02	3e-02	6e-02	7e-02	8e-02
Comm. Reg.	5e-03	1e-02	2e-02	3e-2	7e-2

From the results, we observe spectral weight regularization to have poor long horizon performance. Spectral Jacobian regularization diverges on the UNet while it settles in an average state around 1500-2000 steps whereas commutativity regularization outperforms both baselines.

J Licenses for existing assets

This appendix lists the third-party data, pretrained models, and code we use, together with their licenses and original sources. All assets are used under their respective licenses; we credit the original authors in the references and at the points of use in the main text.

J.1 Datasets

ERA5 reanalysis (used in the FourCastNet experiment, Section 4.3). ERA5 hourly reanalysis on pressure and single levels is produced by the Copernicus Climate Change Service (C3S) at ECMWF and distributed through the Copernicus Climate Data Store (`https://cds.climate.copernicus.eu/`). The data are released under the **Licence to Use Copernicus Products** (`https://cds.climate.copernicus.eu/api/v2/terms/static/licence-to-use-copernicus-products.pdf`), which permits free use, reproduction, and distribution with attribution. We acknowledge: *"Neither the European Commission nor ECMWF is responsible for any use that may be made of the Copernicus information or data it contains."* The original reference is Hersbach et al. [7].

NOAA OISST v2 (used in the SST experiment, Section 4.4). The NOAA $1/4^\circ$ Daily / weekly Optimum Interpolation Sea Surface Temperature (OISST) v2 product (`sst.wkmean.1990-present`) [24] is produced by NOAA NCEI and made available through NOAA PSL (`https://www.psl.noaa.gov/data/gridded/data.noaa.oisst.v2.html`). As a work of the U.S. Government, the dataset is in the **public domain** and not subject to copyright protection in the United States; NOAA requests acknowledgement in publications that use the data, which we provide via the citation above.

WeatherBench 2 evaluation conventions (Section 4.3). We follow the headline-variable and latitude-weighted RMSE conventions of WeatherBench 2 [23]. The WeatherBench 2 code and reference data tooling (`https://github.com/google-research/weatherbench2`) are released under the **Apache License 2.0**.

J.2 Pretrained models

FourCastNet v1 (FCNv1) backbone and weights. We fine-tune the publicly released FCNv1 AFNO backbone [20] from `https://github.com/NVlabs/FourCastNet`. The repository (code and reference checkpoints) is released under the **BSD 3-Clause License**, copyright NVIDIA Corporation. We use the weights as a starting point for fine-tuning on a strict subset of FCN's original ERA5 training years (Section 4.3) and respect the license terms in redistribution and attribution.

J.3 Baseline code

Fourier Neural Operator (FNO) baseline. Our FNO implementation follows the reference codebase of Li et al. [13]. We use the `neuraloperator` library (`https://github.com/neuraloperator/neuraloperator`), released under the **MIT License**.

U-FNO baseline. The U-FNO architecture and reference implementation (`https://github.com/gegewen/ufno`) are released under the **Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0)** license. We use the published architecture for non-commercial academic comparison only and do not redistribute modified weights or code.

PDE-Refiner / PDEArena baseline. The PDE-Refiner implementation is part of the PDEArena codebase (`https://github.com/microsoft/pdearena`), released under the **MIT License**, copyright Microsoft Corporation.

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: Regularizer analysis is in section 2.3 and proof in appendix, in-distrubtion and OOD experiments are in Sec 4.1, 4.2, forecastnet finetuning 4.3, SST 4.4.

   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: Limitation are discussed at the end next to the conclusion under the subheading 'Limitations and outlook'.

   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: [Yes]

Justification: Proof of the propagator bound together with assumption is in Appendix A

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: There are details of each experiment given in the appendix which should be enough for reproduction.

   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: [Yes]

Justification: Source code is included with the submission. The source archive includes all main experiments required as proof of concept. The large scale FCN experiment is excluded which requires more setup. Data generation and download scripts are included for each experiment.

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: All this information is present in the appendix separately for each experiment.

   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: For the main experiments, error bars are include. We do not compute error bars for the climate/FCN experiments which are more expensive.

   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: A description of compute requirements is present in the appendix H .

   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: We have reviewed the NeurIPS Code of Ethics and the research presented conforms to it; the work uses publicly released datasets and pretrained models under their respective licenses (Appendix J) and involves no human subjects, sensitive data, or deployed systems.

   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: [N/A]

    Justification: There is no direct or immediate societal impact of this work, being related to a technical issue in neural operators.

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: Safety concerns are not immediately relevant to this work.

    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: Authors of assets are credited. Links are given the sources. Licenses are referenced in a separate section in the appendix J

    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:

    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:

    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:

    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:

    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.