DRIFT OR DYNAMICS: RESIDUAL CORRECTION OF LONG-HORIZON NEURAL PDE ROLLOUTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learned simulators offer a way to make expensive physical simulation much faster, with potential applications across science and engineering, from fluid and material dynamics to weather and climate modeling. But using a learned model to predict far into the future remains difficult: each prediction becomes the input to the next, so small errors can accumulate until the simulation no longer resembles the true system. Existing approaches improve long-horizon behavior by training models to make longer sequences of predictions, repeatedly refining their outputs, or adding mechanisms that pull predictions back toward physically or statistically plausible states. These strategies can extend the useful prediction horizon, but they leave an important question unanswered: what should we do when a prediction moves away from the states seen during training? Pulling it back seems natural—but a physical system may be genuinely evolving into a state the model has never seen before. We introduce a geometry-aware correction that pulls back only the part of a prediction outside the principal directions of the training data, leaving the in-subspace component untouched. In controlled experiments, this targeted correction reduces long-horizon error by up to 90%. Importantly, we find that the geometry of the correction matters, and when it removes meaningful dynamics, error can instead increase by up to $130\times$. Our results show that long-horizon stability is not simply a matter of keeping predictions close to the training data. The critical question is whether a departure from the data is an error or a part of the dynamics we are trying to predict.

1 INTRODUCTION

Partial differential equations (PDEs) describe fluid flow, the patterning of materials, and the circulation of the atmosphere and ocean. These equations represent the dynamics. They determine the next physical state from the current one. Numerical solution of these equations is the reference method in weather forecasting and engineering design, but the cost of each solve limits how many simulations can be performed. One alternative is to train a neural network on trajectories from that solver and then use the network to predict the system's next state from its current one (Li et al., 2021; Stachenfeld et al., 2022; Kochkov et al., 2021). This learned simulator is trained once. Over many steps it has to keep reproducing the dynamics.

Predicting one step is not enough. The prediction has to stay close to the numerical solution over many steps. To simulate many steps, every later prediction starts from the network's last output rather than from the true state. Because that output is slightly wrong, each new prediction begins in the wrong place, and the simulation drifts away from the numerical solution (Lorenz, 1969; Venkatraman et al., 2015). On chaotic systems the rollout can decohere from the reference within a few steps. Existing approaches solve this with longer training sequences (Brandstetter et al., 2022), repeated refinement of each prediction (Lippe et al., 2023), or mechanisms that relax the state back toward plausible physics (Pedersen et al., 2025). Each extends the useful horizon, but each assumes that a state outside those solver trajectories is itself the failure. What if it is not? A physical system can move into a state those solver trajectories never included, and a correction that moves the prediction back toward those trajectories then erases the dynamics the simulator was meant to reproduce. Distance from the data cannot tell mistake from physics (Sec. 5.1).

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

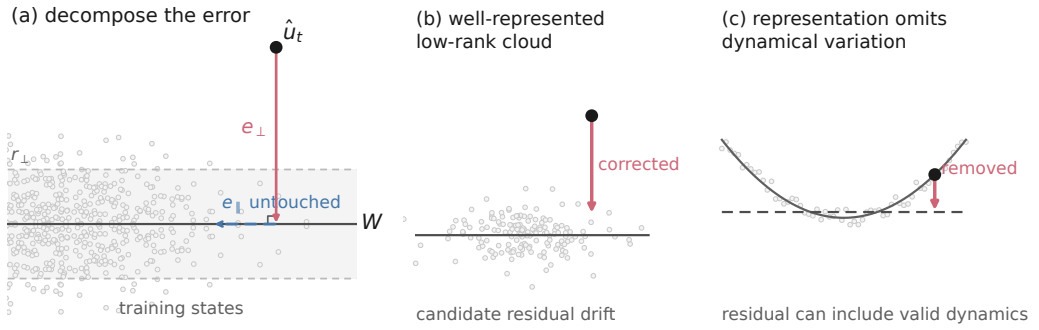


Figure 1: Error decomposition used throughout. A rollout state is split relative to the principal subspace W computed from training states. The residual anchor leaves the in-subspace component untouched and acts only on the residual $r(u)$. Panel (b): when the residual is dominated by rollout drift, contracting it can help. Panel (c) illustrates a case where the residual contains dynamically meaningful structure, as in 2D KS.

We therefore ask a narrower question than how to stabilize a rollout. *Which component of a predicted state should a correction touch, if any?* Our answer is simple. From the states in the solver trajectories, take the directions in which those states vary most. A predicted state then splits into the part along those directions and the part outside them. The outside part is the residual. The correction reduces the residual and leaves the other part alone.

Contributions.

- **A geometric characterization of state-space correction.** Two measurable quantities separate regimes where the correction helps, is inert, or is harmful. One is how concentrated the training cloud's spectrum is, and the other is how much of the rollout error lies off that subspace. A threshold on the pair is a conservative filter for ruling out harm, not a predictor of benefit (Sec. 5.2).
- **A residual contraction with a geometric bound.** A dead-zoned partial contraction on the PCA residual of the training states (Proposition 1), with no additional network evaluations and unchanged in-subspace dynamics. A controlled geometry sweep and a wrong-subspace control isolate that effect (Sec. 5.3).
- **Long-horizon performance.** The 1D Kuramoto–Sivashinsky comparison is Table 2. The Kolmogorov comparison is Table 3, with the per-seed grid in Table 8. The APEBench comparison with published champions is Table 7. The same update composed with a concurrent spectral operator helps some tasks and harms others (Table 4; App. A).

2 Related work

A network can be trained on trajectories from a numerical solver to predict the next state of a physical system. The designs differ in how they represent the dynamics. Some apply the network to spatial patterns of different sizes, which suits the waves and eddies in fluid flow (Li et al., 2021). Some learn the rule that turns one entire field into the next (Lu et al., 2021). Some leave the numerical solver in place and train the network only on the part the solver gets wrong (Kochkov et al., 2021). The same idea has been used to imitate turbulent flow over many steps (Stachenfeld et al., 2022).

The test matters as much as the network. PDEBench and PDEArena score a prediction against a saved dataset (Takamoto et al., 2022; Gupta & Brandstetter, 2023). APEBench instead uses each prediction as the next input and compares the long run with the numerical solver (Koehler et al., 2024). That comparison asks whether the dynamics are still there after many steps. This paper uses that comparison.

The dynamics are not always still there. In a chaotic system a small difference between two states grows (Lorenz, 1969). In a learned model the next input is the network's own output, so that small difference is fed back in (Venkatraman et al., 2015; Talvitie, 2014).

Work on this problem acts in three places. The first is training. The loss can be applied after several predicted steps, so the network is penalized for drift it produced itself (Brandstetter et al., 2022). Training can also replace the true state with the network's own prediction, more often as training goes on (Bengio et al., 2015). A recent method builds extra training targets from the runs in which the network already failed (Zhang et al., 2026a). The loss can also score the spread of predicted states, not only their average. This paper uses the energy score for that purpose (Gneiting & Raftery, 2007).

The second place is the network. Some models predict how a latent state is changing, rather than a single next state, and so do not feed a discrete prediction back in at every step (Zhang et al., 2026b; Hou et al., 2026). Others write the next-state rule in terms of spatial patterns of different sizes, so the dynamics at each size are part of the rule. Structured spectral propagators do this (Lu et al., 2026), and so do spectral generator operators (Li et al., 2026).

The third place is the state after the prediction. A second network can revise that state several times before it is used as the next input (Lippe et al., 2023). The state can also be disturbed with noise and then denoised, which moves it toward states the denoiser treats as plausible (Pedersen et al., 2025). Diffusion models have been used in a similar way to continue a long run (Chen et al., 2024; Ruhe et al., 2024). Each of these steps runs another network. A step that moves the state back toward familiar states can also remove dynamics the first network had already predicted correctly.

This paper keeps the trained network fixed. It finds the directions in which the solver states vary most, and it shrinks only the part of the prediction that lies outside those directions. The correction does not run another network. Learned surfaces have been used to describe a set of plausible states (Park et al., 2019; Gropp et al., 2020). A surface of that kind did not stay accurate at the resolutions studied here, so the directions are computed from the solver states themselves. The experiments compare the correction with APEBench and with long runs from the released spectral generator, including the runs in which the dynamics are preserved and the runs in which they are erased (Secs. 5.5 and 5.6).

3 METHOD

3.1 SETTING AND DRIFT DECOMPOSITION

Let $u_t \in \mathbb{R}^N$ follow a simulator $\mathcal{S}$, $u_{t+1} = \mathcal{S}(u_t)$, and let $\hat{f}_\theta$ be a learned stepper rolled out autoregressively. We study stabilizers applied to each predicted state that reduce long-horizon error without changing the predictor's weights. Fix the training-state mean μ and the principal subspace W computed from training states using Gram PCA on a ≤ 4000-row subsample. We retain enough directions to capture 99.95% of training variance, with rank floor 16 and cap 768; this dimension is d_W. Separately, the participation ratio $r_{\text{eff}} = (\sum_i \lambda_i)^2 / \sum_i \lambda_i^2$ of the same spectrum measures how concentrated the training cloud is: a later threshold on r_{eff} is a statement about spectral concentration; it is distinct from d_W, the number of directions the residual anchor actually uses. Writing $u = \mu + WW^\top(u - \mu) + r(u)$ decomposes any state into within-subspace and residual parts. State error $e_t = \hat{u}_t - u_t$ likewise splits into an in-subspace component and a transverse (off-subspace) component. We use *faithful* operationally to mean that the retained subspace captures the dynamically relevant variation needed for the evaluated rollout horizon. 3D unbalanced advection is a concrete case where it does not (Table 7). Empirical proxies are r_{eff} and the transverse error fraction $D_\perp$ of Eq. 1.

Let $P_\perp = I - WW^\top$. For a trained predictor held fixed,

$$D_\perp = \mathbb{E}\left[\frac{\|P_\perp e_t\|_2^2}{\|e_t\|_2^2}\right]. \tag{1}$$

$D_\perp$=0 means all state error lies in the retained subspace; $D_\perp$=1 means it is entirely transverse. The expectation is the mean per-step ratio over a 100-step rollout on the 30 test-split ICs after training. Those ICs are the full APEBench test set, overlapping GMean@100 and the map's last-15 evaluation

ICs. Because $D_\perp$ is computed on the full test split, the regime analysis is retrospective rather than a held-out deployment test of the filter; we interpret the threshold only as a descriptive rule, not as evidence of prospective generalization. $D_\perp$ does not enter training or residual-anchor calibration and does not select any headline model.

3.2 THE RESIDUAL ANCHOR

Let $\{u_i\}$ be training states with mean μ and principal subspace W (computed directly from the training states, not learned), and let $r(u) = (I - WW^\top)(u - \mu)$, $h(u) = \|r(u)\| - r_\perp$. The dead zone is $r_\perp = q_{90}$ of $\|r\|$ over training states and the firing margin m is q_{95} of $|\,\|r\| - r_\perp|$ on a held-out 10% of training states. The update is applied when $\|r(u)\| > r_\perp + m$:

$$u \;\leftarrow\; u \;-\; \kappa\left(\|r(u)\| - r_\perp\right)\frac{r(u)}{\|r(u)\|}, \qquad \kappa = 0.5, \tag{2}$$

and the identity otherwise. Because the update is radial in $r(u)$, this is exactly a radial contraction of the residual.

Proposition 1 (Contraction and tube invariance)**.** *Let $\kappa \in (0, 1]$. Whenever the update fires, $\|r\|$ strictly decreases and the excess over the dead zone contracts geometrically: $\|r'\| - r_\perp = (1 - \kappa)\,(\|r\| - r_\perp)$; in particular the update does not increase the residual norm and moves the state by at most its own excess $(\|r\| - r_\perp)$, so in-subspace content is untouched and no step can overshoot the tube.*

Proof. $r' = r - \kappa h\hat{r}$ with $h = \|r\| - r_\perp > m$, so $\|r'\| = \|r - \kappa h r/\|r\|\| = \|r\| - \kappa h$ by homogeneity, and the step length is $\kappa h \le h$. □

This guarantees only geometric invariance of the residual tube, not lower rollout error: a correction can leave the residual norm non-increasing while still deleting dynamically meaningful components (Sec. 5.1). A separately evaluated *capped* variant replaces the per-step contraction by $\min\{\kappa h,\, m\}$ and is hence bounded by the margin (App. C); the proposition covers the uncapped update used in the headline results.

The anchor adds two projections through the stored principal directions; it does not evaluate the learned network. NFE in this paper counts evaluations of *learned* modules per rollout step: 1 for every predictor we train (one sample at inference) and 2 for the Thermalizer analog (predictor plus denoiser). An optional spectral-band correction appears only in appendix probes, is fit on half of the test initial conditions, and is not used in any headline result.

3.3 SETTING AND PREDICTOR

Setting means one task-and-predictor pairing; correction on/off is then evaluated within each setting. The long-horizon predictor is trained with the energy score (Gneiting & Raftery, 2007) (four-member ensemble at train time, one sample at inference). We use one-step training on smooth or laminar dynamics and five-step supervised unrolling on chaotic ones.

4 EXPERIMENTAL PROTOCOL

We pre-specified the tested variants and evaluation protocols and report all tested outcomes; full per-seed results and the protocol are in App. A. Unless otherwise stated, reported aggregates are medians with min–max over *all* seeds; no completed seed run was excluded. Bootstrap CIs resample the 17 setting-level units. Kolmogorov reports every 100,000-step seed (n=4). APEBench uses 3 seeds on 1D rows and 10 on 2D/3D, against the benchmark's 50/20. Our APEBench row uses the published champion backbone (Table 13 of Koehler et al. (2024)) and the training rule of Sec. 3, with residual correction only on 1D dispersion.

The rollout metric is the mean normalized RMSE per step aggregated geometrically over 100 steps (GMean@100, APEBench's published Eq. 31): a single ratio over *all* degrees of freedom. Valid time VT_q is the first rollout step at which the spatial correlation with the reference drops below q; spectrum error is the relative L_2 error of log power spectra at a given lead (0 is exact agreement);

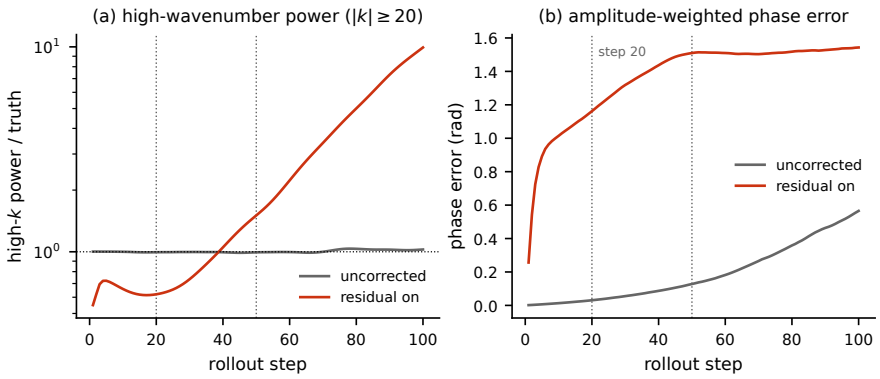


Figure 2: 2D Kuramoto–Sivashinsky diagnostic (mean over the same 3 test ICs). (a) High-wavenumber power relative to the reference. (b) Amplitude- weighted phase error. The anchored rollout develops large phase error by step 20, before the later spectral blowup. Vertical lines mark steps 20 and 50.

d_K is the distance from a rollout state to the training cloud; NFE counts learned-module evaluations per step as defined above.

5 Results

The subsections below separate the cases in which the correction helps from the cases in which it removes dynamics. Section 5.1 is the 2D Kuramoto–Sivashinsky failure. Section 5.2 and Table 1 give the regime rule. Section 5.3 and Fig. 4 test that rule in a controlled geometry. Section 5.4 compares long rollouts with a re-noise/denoise baseline in Tables 2 and 3. Section 5.5 and Table 4 apply the correction to an independently trained operator. Section 5.6 and Table 7 compare with published champions.

5.1 The correction can help—or catastrophically hurt

On 2D Kuramoto–Sivashinsky the residual anchor increases GMean@100 relative to the same predictor with the correction off (Table 1). At step 20 the corrected rollout has suppressed high-wavenumber power ($|k| \geq 20$) to 0.54–0.77× the truth (uncorrected: 0.99–1.00×), while the amplitude-weighted phase error jumps from 0.03–0.04 rad to 1.14–1.18 rad (mean over three of the test ICs; Fig. 2). High-k power then reaches 6–13× truth at step 100. The correction removes high-wavenumber content that the uncorrected predictor tracks close to the reference. The results below ask when the residual anchor is plausible to apply: transverse drift can help, in-subspace drift is inert, and valid residual dynamics are harmful (Table 1).

5.2 The geometric regime

Fig. 3 measures two quantities over 221 seed/predictor observations nested in 17 settings (our predictors plus the authors' champion predictors at 20 seeds on five 2D tasks). Effective rank is the participation ratio of each task's training cloud; $D_\perp$ is Eq. 1. We treat settings—not individual seeds—as the primary inferential units. In the retrospective analysis, a setting is on-rule when its median effective rank is ≤ 10 and its median $D_\perp$ is ≥ 0.2; we call this on/off rule the *filter*. The same qualitative pattern appears on the authors' champion predictors, indicating that it is not specific to our predictor family. Table 1 summarizes the resulting regimes.

2D KS illustrates why $D_\perp$ alone is not sufficient: its high $D_\perp$ (0.81) reflects large residual content, but that content is dynamically meaningful (Fig. 1c) rather than drift, so the filter uses both quantities jointly, and neither alone predicts benefit.

Table 1: Four regimes of residual correction. Pixel-latent models instantiate the in-subspace row.

Regime	What residual contains	Outcome	Evidence
Transverse rollout drift	predominantly rollout error	**helps**	1D KS, 1D dispersion, controlled sweep (up to 90% GMean@100 reduction)
In-subspace drift	error invisible to the anchor	**inert**	controlled $\alpha_\perp$=0; pixel latent (fires on 45% of steps, still diverges)
Valid residual dynamics	signal treated as error	**harms**	2D KS (up to 130× GMean@100; Sec. 5.1)
Misaligned subspace	wrong correction direction	inert or harmful	control (Sec. 5.3)

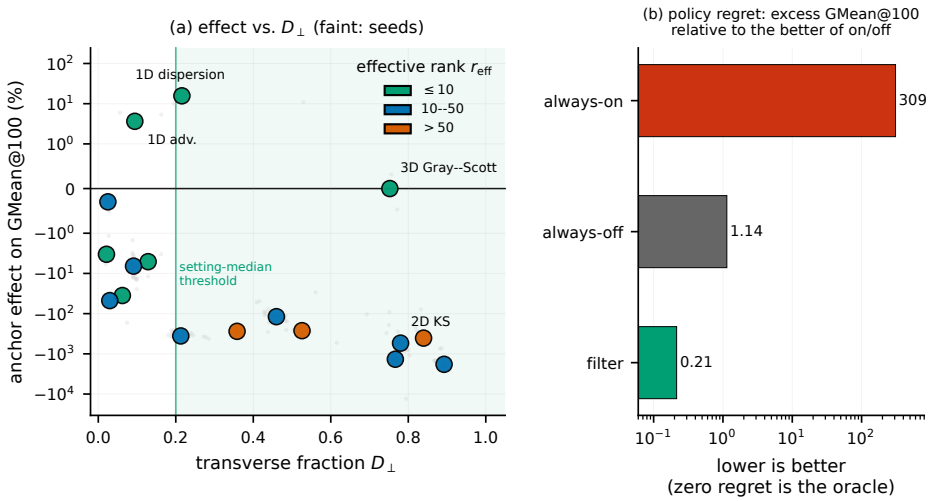


Figure 3: Per-seed correction effect versus $D_\perp$ (faint points) with setting-level medians overlaid and colored by effective rank r_{eff} (legend). The vertical line is the filter's $D_\perp = 0.2$ threshold, applied to setting medians (large outlined markers), not to individual seeds. Faint small points: seeds. Panel (b): policy regret R of Eq. 3 (log scale; zero regret is the oracle and is not a plotted bar).

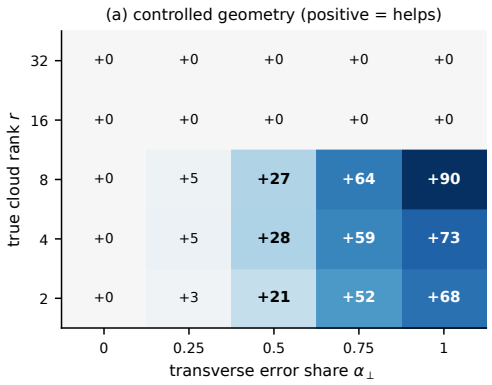
Because positive settings are rare, we report policy regret rather than classification accuracy. Let Δ be a setting's median percent effect of enabling the correction (positive = helps). Identify $L_{\text{off}} = 0$ and $L_{\text{on}} = -\Delta$ in units of percent of the uncorrected GMean@100. Policy regret on that setting is

$$R(\pi) = L(\pi) - \min\big(L_{\text{on}},\, L_{\text{off}}\big). \tag{3}$$

Thus $R(\text{on}) = \max(-\Delta, 0)$ and $R(\text{off}) = \max(\Delta, 0)$; the filter pays $R(\text{on})$ on selected settings and $R(\text{off})$ otherwise. Mean R over the 17 settings is 309 for always-on (median 47; bootstrap 95% CI $[104, 571]$), 1.14 for always-off (median 0; $[0, 3.2]$), and 0.21 for the filter (median 0; $[0, 0.64]$). The filter–always-off difference is 0.92 ($[0, 2.8]$, includes zero; bootstrap over the 17 setting-level units). The geometry is useful primarily for rejecting corrections that are harmful in our measurements; with 17 setting-level units it does not establish a gain over always-off. It selects 2 of 17 settings. 1D advection is a missed positive (all three seeds positive, $D_\perp$ 0.08–0.16); 3D Gray–Scott is on-rule yet seed-bimodal. Refitting the thresholds under leave-one-family-out cross-validation across the eight PDE families fails to recover 1D dispersion when that family is held out. The two quantities describe an applicability *regime*; the threshold is a post-hoc conservative filter, not a transferable classifier.

5.3 Controlled geometry and the data subspace

A controlled sweep with prescribed error geometry (true cloud rank r; transverse share $\alpha_\perp = \|e_\perp\|^2/\|e\|^2$ of $e = e_\parallel + e_\perp$; correction subspace capped at the top eight directions) isolates the



(b) wrong subspace (κ=0.5), median %

Task	PCA	random	rand. dirs
1D dispersion	+15.6	+0.0	−2697
3D Gray–Scott	+0.0	+0.0	+26.9
2D KS	−745	+0.0	−490
2D Burgers	−1358	+0.0	−1705

Figure 4: (a) Controlled environment: each cell is the median percentage change in GMean@100 relative to the unanchored control (positive = the anchor helps) versus true cloud rank r and prescribed transverse error share $\alpha_\perp$. (b) Wrong-subspace control: median percent effect of the residual anchor. PCA; a same-rank random subspace with its own calibration; random directions with the PCA dead zone.

Table 2: 1D Kuramoto–Sivashinsky, 3-seed means (no seed dropped). Columns: Predictor, Training, Anchor (Eq. 2). VT = first lead at which spatial correlation drops below 0.9/0.5; spec = relative L_2 error of log power spectra (0 is exact agreement); d_K = median cloud distance; NFE = learned-module evaluations per step (residual anchor not counted).

Predictor	Training	Anchor	$VT_{90}\uparrow$	$VT_{50}\uparrow$	spec@300↓	spec@600↓	d_K@300↓	NFE↓
Deterministic	1-step	—	189	368	1.050	1.208	36.9	1.00
Energy score	1-step	—	198	361	1.134	1.653	40.7	1.00
Deterministic	5-step unroll	—	205	408	1.041	1.176	36.1	1.00
Energy score	5-step unroll	—	210	414	1.038	1.151	36.6	1.00
Thermalizer (1D port)	**—**	**—**	**185**	**304**	**1.147**	**2.586**	**60.6**	**2.00**
Energy score	1-step	yes	200	377	1.020	1.064	38.1	1.00
Energy score	**5-step unroll**	**yes**	**212**	**421**	**1.006**	**0.995**	**36.1**	**1.00**

mechanism in the controlled environment (Fig. 4a): holding total error magnitude fixed while varying $\alpha_\perp$, contraction helps when error is forced into the corrected subspace and not when it is forced into the untouched subspace. Benefit grows with $\alpha_\perp$ at low rank and is negligible for in-subspace error (Fig. 4a). At high true rank the update is essentially *inert* in this construction. The dead zone almost never fires, so the change is not harmful. Fig. 4 isolates the same distinction seen in Fig. 2 (Sec. 5.1). The effect depends on alignment with the data subspace (Fig. 4b and Table 10): a same-rank random subspace is inert under its own calibration, while random directions with the PCA dead zone are harmful. The improvement is not caused by contraction alone; it depends on the actual geometry of the training distribution. Robustness in κ is in App. C.

5.4 Long-horizon stabilization

Table 2 reports the 3-seed 1D Kuramoto–Sivashinsky comparison. *5-step* is supervised unrolling of length 5; *Anchor* marks the residual anchor of Eq. 2. Thermalizer is a 1D port of the released re-noise/denoise mechanism (the authors' code is 2D-only); NFE follows the convention of Sec. 3.2 (1 per predictor, 2 for Thermalizer's predictor-plus-denoiser pair).

The energy-score 5-step predictor with residual anchor beats the Thermalizer port on every reported metric, at half the learned-module evaluations (Table 2). **Unrolling accounts for most of the valid-time gain. The residual anchor adds little valid time and improves spectral fidelity (Table 2).** Fig. 5(a) shows that the chaotic band structure remains visually coherent for 100 steps (Table 2 evaluates to 600); Fig. 5(b) shows the corresponding 2D fields against the published champion weights.

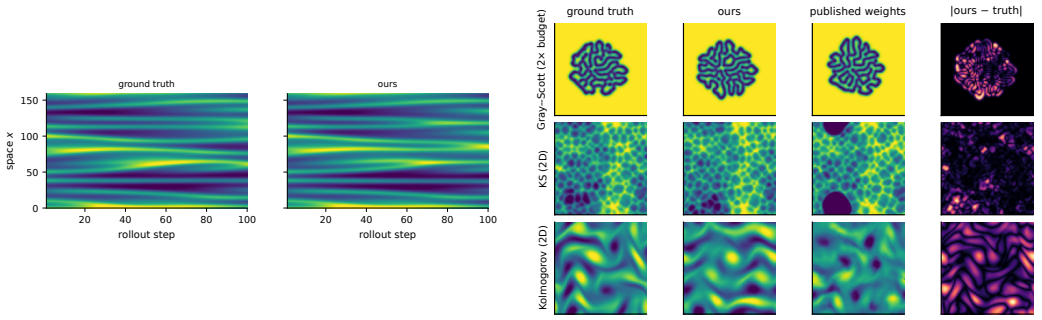


Figure 5: (a) Kuramoto–Sivashinsky (APEBench 1D task): space–time diagrams of the reference continuation (left) and our energy-score, 5-step, residual-anchored rollout (right), one test initial condition; 100 steps shown (Table 2 evaluates to 600). (b) Step-100 rollouts (one IC). Rows: 2D Gray–Scott (2× budget), 2D KS, Kolmogorov. Columns: reference, ours, published weights, absolute error. Row fields share the reference 1st–99th percentile; the error column is independent.

Table 3 reports the 100,000-step Kolmogorov comparison, including the Thermalizer authors' released implementation (emulator, thermalizer, and thermalization algorithm, retrained per seed here).

Table 3: 2D Kolmogorov flow, 100,000-step rollouts, 40 trajectories per seed. Each cell is median escape step / escaped trajectories out of 40 (never = median never escapes). Seed 1 escaped counts were not logged. Thermalizer n=3 (s3 not run). spec@600 is mean [min–max] (Thermalizer: median).

Method	s0	s1	s2	s3	spec@600
Det., 4-step unroll	3,358 / 40	never / —	never / 1	8,512.5 / 40	0.0067 [0.0051–0.0087]
Energy score, 1-step	never / 19	never / —	never / 16	never / 9	0.0094 [0.0069–0.0133]
Energy score, 1-step + residual	never / 4	never / —	never / 0	never / 9	0.0094 [0.0069–0.0133]
Energy score, 5-step + residual	never / 1	never / —	4,959 / 40	never / 7	0.0167 [0.0046–0.0524]
Thermalizer (released code)	2,694 / 33	2,855 / 22	1,582 / 34	—	0.459 [0.457–0.518]

The 1-step energy-score predictor has no median escape through 100,000 steps on any seed (Table 3). Adding the residual anchor lowers the escaped share on some seeds and leaves others unchanged. The 5-step residual-anchored predictor still escapes on every trajectory of one seed, and that is the seed on which the correction fires most often (Table 8). Longer-horizon training does not make the correction automatically safer, and heavy firing did not prevent the escape. The released Thermalizer median-escapes on every seed and remains spectrally behind the non-collapsing 5-step residual seeds (Table 3). Per-seed numbers, including the unlogged seed-1 escape counter, are in Table 8.

5.5 The correction is complementary to operator design

Table 4 runs SGNO's (Li et al., 2026) released code on identical task data with its canonical configurations and the published APEBench Eq. 31 metric. SGNO is stronger on linear/dispersive rows; on the nonlinear 2D tasks in the table, our GMean is lower (Table 4). Composition with SGNO is not uniformly safe. The same update helps some of the shared tasks and substantially harms others (Table 4). 2D Gray–Scott is helped despite a high effective rank. Full comparison: Table 6.

5.6 Benchmark breadth

Fig. 6 compares our run on each published champion backbone (Sec. 4) with the published champion median (Table 13 of Koehler et al. (2024)). The win, tie, and loss tally, including the disclosed Gray–Scott budget and the pre-specified 2D KS schedule, is in Fig. 6 and Table 7.

Table 4: SGNO composition on identical task data. ΔGMean: percent change in SGNO GMean@100 after the residual (positive = improvement). †2× budget in the Ours column.

Task	SGNO	Ours	SGNO+residual	ΔGMean (%)
1D dispersion	0.0008	0.0305	0.0005	+36%
1D advection	0.0122	0.0753	0.0118	+3%
2D Gray–Scott†	0.2487	0.0527	0.2007	+19%
2D KS	0.7624	0.1688	1.1146	-46%
2D Burgers	0.2457	0.0539	0.7903	-222%

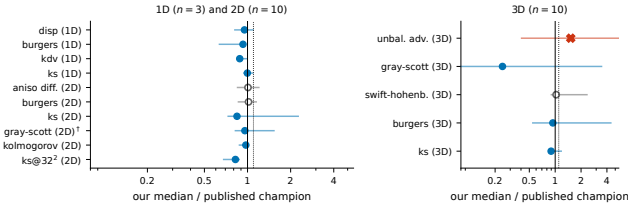


Figure 6: APEBench breadth: our median divided by the published champion median. Horizontal lines: min–max over seeds (n=3 on 1D, n=10 on 2D/3D). Filled: wins; open: ties (1.1× band); crosses: losses. †2× published training budget.

6 Limitations

2D Gray–Scott needed a doubled budget. CEM planning through correction-carrying models fails for both corrected variants (App. D: $+25.5$/$+25.8$ vs. -1.4 in return), plausibly reflecting the update's discontinuity at the dead-zone boundary. Because $D_\perp$ uses the evaluation test split, the regime analysis is retrospective; the threshold neither transfers to held-out PDE families nor significantly beats always-off.

7 Conclusion

The introduction asked which component of a predicted state a correction should touch, if any. The answer is the residual. That is the part of the predicted state outside the directions in which the solver trajectories vary most. The correction should shrink that residual only when it is rollout drift. When the residual is dynamics the predictor was already tracking, the correction should touch nothing.

2D Kuramoto–Sivashinsky is the harmful case. Contracting the residual removes high-wavenumber structure the predictor was tracking, and the long-horizon error grows (Sec. 5.1, Fig. 2). The training states sometimes lie along a few directions, and the error sometimes lies mostly outside those directions. The correction is turned on only when both are true. That happens for 2 of 17 settings, and the difference from leaving it off is not distinguishable from zero (Sec. 5.2, Table 1, Fig. 3). When the error is placed outside those directions, shrinking it helps. When the error is placed along them, shrinking the outside part does nothing. Wrong directions do nothing or make the error worse (Sec. 5.3, Fig. 4). On 1D Kuramoto–Sivashinsky, most of the longer useful horizon comes from training on several steps at once. The correction adds little to that horizon. It does bring the spectrum closer to the numerical solution than a step that adds noise and then removes it (Table 2). On Kolmogorov flow, the rollout still diverges on the seed where the correction acts on the most steps (Tables 3 and 8). Applied after a spectral network trained separately, the same correction lowers the error on some tasks and raises it on others (Table 4). Against the best published network on each task, some of our runs are more accurate, some match, and some are worse (Fig. 6, Table 7).

A gate that can be computed from the training states alone is still open, because the filter here uses the evaluation split. Composing the residual correction with training-time losses and with refinement steps is also open. Those steps act on other parts of the error. Nonlinear subspaces learned from the states did not stay accurate at the resolutions studied here. A later stabilizer still has to choose which directions of a predicted state to touch.

REPRODUCIBILITY STATEMENT

The pre-registered protocol and the full hyperparameter inventory (Table 12) are included in this paper. Per-seed results and qualitative rollouts are in Appendix A; the metric cross-validation is in Appendix B; compute and timing details are in Appendix D; and the verification inventory is in Appendix F.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing and editing the manuscript, and to create or edit software code. We have not used generative AI tools to generate synthetic data sets, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, or interpret results; the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to draft parts of the paper, summarize or analyze existing literature, suggest a structure for the paper, propose a title, format references, and edit the paper to improve readability. We have reviewed all AI-assisted work: LLM-generated and LLM-edited code was verified against the repository's checks and smoke tests, and all claims and numbers in the manuscript were checked against the reported artifacts by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural PDE solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning (ICML)*, 2020.

Jayesh K. Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *Transactions on Machine Learning Research (TMLR)*, 2023. arXiv:2209.15616.

Xianglong Hou, Xinquan Huang, and Paris Perdikaris. CFO: Learning continuous-time PDE dynamics via flow-matched neural operators. In *International Conference on Learning Representations (ICLR)*, 2026.

Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. In *Proceedings of the National Academy of Sciences*, volume 118, 2021. arXiv:2102.01010.

Felix Koehler, Simon Niedermayr, Rüdiger Westermann, and Nils Thuerey. APEBench: A benchmark for autoregressive neural emulators of PDEs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2024. arXiv:2411.00180.

Jiayi Li, Penghao Jiang, Hira Saleem, Zhaonan Wang, Piotr Koniusz, and Flora D. Salim. SGNO: Spectral generator neural operators for stable long horizon PDE rollouts. *arXiv preprint arXiv:2602.18801*, 2026.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2010.08895.

Phillip Lippe, Bastiaan S. Veeling, Paris Perdikaris, Richard E. Turner, and Johannes Brandstetter. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. arXiv:2308.05732.

Edward N. Lorenz. The predictability of a flow which possesses many scales of motion. *Tellus*, 21 (3):289–307, 1969.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021. arXiv:1910.03193.

Xiaoxiao Lu, Ye Yuan, and Jiahao Shi. Stable long-horizon PDE forecasting via latent structured spectral propagators. *arXiv preprint arXiv:2605.10154*, 2026.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Christian Pedersen, Laure Zanna, and Joan Bruna. Thermalizer: Stable autoregressive neural emulation of spatiotemporal chaos. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2503.18731.

David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. In *International Conference on Machine Learning (ICML)*, 2024.

Kimberly Stachenfeld, Drummond Buschman Field, Ian Grooms, Noah D. Brenowitz, Michael Pritchard, and Stephan Hoyer. Learned coarse models for efficient turbulence simulation. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2112.15275.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBENCH: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks*, 2022. arXiv:2210.07182.

Erik Talvitie. Model regularization for stable sample rollouts. In *Uncertainty in Artificial Intelligence (UAI)*, 2014.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *AAAI Conference on Artificial Intelligence*, 2015.

Jiaquan Zhang, Shuxu Chen, Haifan Meng, Yi Lu, Zhihan Lyu, Fan Mo, Wei Dong, Yang Yang, and Chaoning Zhang. HERO: History-enriched rollout training for long-horizon autoregressive neural operators. *arXiv preprint arXiv:2607.29135*, 2026a.

Jiaquan Zhang, Caiyan Qin, Haoyu Bian, Libin Cai, Yi Lu, Chaoning Zhang, Wei Dong, Yuanfang Guo, Yang Yang, and Heng Tao Shen. Autoregression-free neural operators for time-dependent PDEs. *arXiv preprint arXiv:2605.25413*, 2026b.

Appendix A contains per-seed benchmark results and qualitative rollouts; Appendix B documents the APEBench metric implementation; Appendix C contains residual-update controls and rejected variants; Appendix D gives compute and controlled-environment details; Appendix E lists the calibrations and hyperparameters.

A PER-SEED RESULTS AND QUALITATIVE ROLLOUTS

Table 5 reports every 1D/2D seed of the APEBench breadth table (no seed dropped); 3D seeds appear as min–max in Table 7. Each breadth row uses the published champion backbone and a pre-specified training rule (one-step on smooth tasks, five-step unrolling on chaotic ones). Residual correction is on only for 1D dispersion: residual-on median 0.0305 vs. published 0.032 (win); residual-off 0.0369 is a $1.15\times$ loss. The other fourteen published rows are uncorrected. Gray–Scott at $2\times$ the published step budget was registered after a $1.15\times$ loss at $1\times$; 2D KS at 32^2 ran both registered schedules, and the pre-specified five-step schedule is the reported win. Under the pre-specified arena rule and published step budget those two rows are: Gray–Scott one-step at $1\times$, median 0.0635 $[0.0484, 0.1135]$ vs. published 0.055 ($1.15\times$, lose); 2D KS at 32^2 with five-step unrolling, median 0.2239 (the reported win; the one-step alternative was 0.3572, a $1.31\times$ loss, and is not the arena-rule row). The pre-specified published-budget tally is therefore $13/15$; the reported tally is $14/15$ after the disclosed Gray–Scott budget change (the KS 32^2 arena-rule row is already the reported win). A $2\times$-budget rescue of 3D unbalanced advection still lost; the table keeps the $1\times$ loss. Table 8 is the per-seed Kolmogorov 100k grid. Table 7 is the full breadth table. Table 6 is the full SGNO comparison. Figure 8 shows the full qualitative gallery; Fig. 9 the 3D slices; Fig. 7 per-step nRMSE.

Table 5: Per-seed GMean@100 for Table 7. — = not run.

Task	pub. arch	pub.	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	median
1d_diff_disp	Conv	0.032	0.0258	0.0305	0.0355	—	—	—	—	—	—	—	0.0305
1d_diff_burgers	Res	0.013	0.0121	0.0082	0.0124	—	—	—	—	—	—	—	0.0121
1d_diff_kdv	Dil	0.077	0.0679	0.0775	0.0657	—	—	—	—	—	—	—	0.0679
1d_diff_ks	Dil	0.157	0.1466	0.1731	0.1565	—	—	—	—	—	—	—	0.1565
2d_phy_aniso_diff	Conv	0.016	0.0187	0.0154	0.0135	0.0177	0.0163	0.0159	0.0194	0.0168	0.0149	0.0145	0.0161
2d_diff_burgers	Res	0.053	0.0539	0.0466	0.0566	0.0452	0.0557	0.0617	0.0556	0.0526	0.0539	0.0500	0.0539
2d_diff_ks	Res	0.200	0.1448	0.1588	0.3228	0.1562	0.4571	0.4071	0.1668	0.1694	0.3281	0.1683	0.1688
2d_phy_gs_theta	Conv	0.055	0.0530	0.0447	0.0462	0.0578	0.0851	0.0472	0.0482	0.0555	0.0524	0.0544	0.0527
2d_phy_kolm_flow	FNO	0.689	0.6926	0.6708	0.6778	0.6663	0.5976	0.6786	0.6655	0.6704	0.6816	0.6341	0.6706
2d_diff_ks32	FNO	0.272	0.2239	0.1831	0.2411	—	—	—	—	—	—	—	0.2239

Table 6: SGNO composition on identical task data (released code, canonical configurations, APEBench Eq. 31). "ours" is the reported breadth row of Table 7; ratio < 1 means SGNO better. Residual = whether that breadth row applies the residual update. †Gray–Scott uses the doubled budget.

Task	ours (median)	SGNO	ratio	residual
1D adv.	0.0753	0.0122	0.16×	residual on
1D diff.	0.0135	0.0024	0.18×	uncorrected
1D disp.	0.0305	0.0008	0.02×	residual on
1D burgers	0.0121	0.0132	1.09×	uncorrected
1D kdv	0.0679	0.0127	0.19×	uncorrected
1D ks	0.1565	0.0294	0.19×	uncorrected
2D aniso	0.0161	0.0143	0.89×	uncorrected
2D burgers	0.0539	0.2457	4.56×	uncorrected
2D ks	0.1688	0.7624	4.52×	uncorrected
2D gs†	0.0527	0.2487	4.72×	uncorrected
2D kolm	0.6706	0.5782	0.86×	uncorrected
2D ks32	0.2239	0.1367	0.61×	uncorrected
3D unbal	0.2033	0.0220	0.11×	uncorrected
3D gs	0.0127	0.0217	1.71×	uncorrected
3D sh	0.0503	0.1225	2.43×	uncorrected
3D burgers	0.1371	0.1447	1.06×	uncorrected
3D ks	0.3308	1.5706	4.75×	uncorrected

Table 7: APEBench breadth, GMean@100 (Eq. 31; lower better). Published champions from Koehler et al. (2024) Table 13 (their medians over 50 seeds in 1D, 20 in 2D/3D). Ours: median [min–max] of our run on that champion backbone (3 seeds in 1D, 10 otherwise); protocol in Sec. 4 and App. A. tie = within $1.1\times$. Below the midrule: the five published 3D tasks, each on a $32\times32\times32$ grid (APEBench 3D resolution). †2D Gray–Scott trains with $2\times$ the published step budget (disclosed).

Task	pub. arch	pub.	ours	[min,max]	seeds	verdict
1d_diff_disp	Conv	0.032	0.0305 [0.026,0.035]		3	WIN (0.95x)
1d_diff_burgers	Res	0.013	0.0121 [0.008,0.012]		3	WIN (0.93x)
1d_diff_kdv	Dil	0.077	0.0679 [0.066,0.078]		3	WIN (0.88x)
1d_diff_ks	Dil	0.157	0.1565 [0.147,0.173]		3	WIN (1.00x)
2d_phy_aniso_diff	Conv	0.016	0.0161 [0.013,0.019]		10	tie (1.01x)
2d_diff_burgers	Res	0.053	0.0539 [0.045,0.062]		10	tie (1.02x)
2d_diff_ks	Res	0.200	0.1688 [0.145,0.457]		10	WIN (0.84x)
2d_phy_gs_theta†	Conv	0.055	0.0527 [0.045,0.085]		10	WIN (0.96x)
2d_phy_kolm_flow	FNO	0.689	0.6706 [0.598,0.693]		10	WIN (0.97x)
2d_diff_ks32	FNO	0.272	0.2239 [0.183,0.241]		3	WIN (0.82x)
3d_phy_unbal_adv	Res	0.134	0.2033 [0.053,1.000]		10	lose (1.52x)
3d_phy_gs_theta	Conv	0.052	0.0127 [0.000,0.184]		10	WIN (0.24x)
3d_phy_sh	Res	0.049	0.0503 [0.043,0.117]		10	tie (1.03x)
3d_diff_burgers	Res	0.146	0.1371 [0.079,0.661]		10	WIN (0.94x)
3d_diff_ks	UNet	0.369	0.3308 [0.306,0.443]		10	WIN (0.90x)

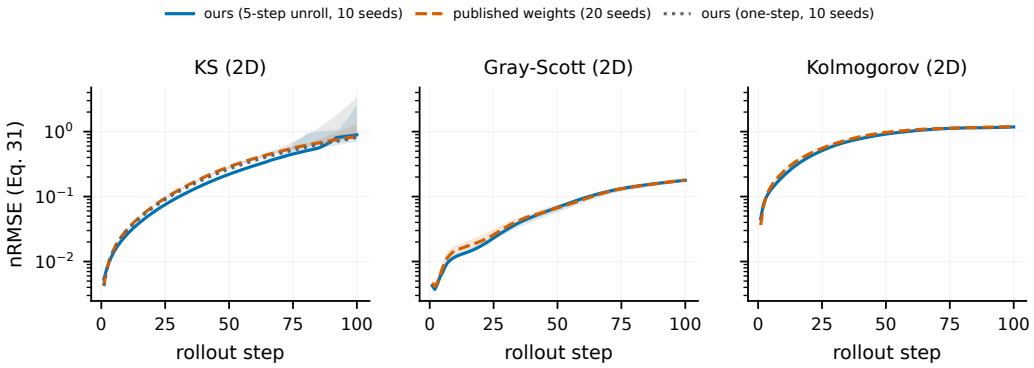


Figure 7: Per-step nRMSE (median over seeds, shaded: interquartile range) for the 2D breadth tasks: our recipes versus the published weights evaluated in the same harness. Lower is better; log scale.

B Reproducing APEBench's published metric

During reproduction we found that the released APEBench implementation differs from the published Eq. 31 on two-channel fields. Recomputing Eq. 31 from the authors' released weights reproduces their published values; the released channel-summed implementation differs by up to $7\times$. We use the published equation for all reported comparisons.

C Anchor controls and rejected components

The regimes of Table 1 are the conceptual taxonomy; pixel-latent models instantiate the in-subspace row.

Fig. 12 reports contraction-strength robustness. Across $\kappa \in \{0.1, 0.25, 0.5, 0.75, 1.0\}$ the 1D dispersion median effect is positive at every κ and monotone ($+11.0\%$ to $+16.5\%$), while the 2D harms persist at every κ (-493% to -856% on KS; -1127% to -1388% on Burgers): the sign of the effect is not tuned, and $\kappa = 0.5$ is an interior default. 3D Gray–Scott is seed-bimodal throughout.

Table 8: Kolmogorov flow, per-seed 100,000-step grid. NFE is learned-module evaluations (residual update not counted). Fire is the fraction of executed (trajectory, step) pairs on which the residual update (or thermalization) fired; — = no residual update; rates below 10^{-3} are in scientific notation so they are not rounded to 0.000. On 1-step residual seeds 0 and 2 the update is a rare event (1.2×10^{-4} and 4.5×10^{-4} of $40\times100{,}000$ steps). Escaped counts for seeds 2–3 are from first-escape log lines. The seed-1 trajectory-level escape counter was not logged; median escape and spec@600 for that seed were. "Released emulator (thermalization off)" is the same released emulator with the thermalization step disabled. The 1D Thermalizer *port* in Table 2 always evaluates a denoiser (NFE 2); the released 2D code thermalizes rarely (NFE ≈ 1).

Method	seed	median escape	escaped	spec@600	NFE	fire
Det., 4-step unroll	s0	3,358	40/40	0.0087	1.00	—
Det., 4-step unroll	s1	never	—	0.0060	1.00	—
Det., 4-step unroll	s2	never	1/40	0.0071	1.00	—
Det., 4-step unroll	s3	8,512.5	40/40	0.0051	1.00	—
Energy score, 1-step	s0	never	19/40	0.0069	1.00	—
Energy score, 1-step	s1	never	—	0.0070	1.00	—
Energy score, 1-step	s2	never	16/40	0.0105	1.00	—
Energy score, 1-step	s3	never	9/40	0.0133	1.00	—
Energy score, 1-step + residual	s0	never	4/40	0.0069	1.00	1.2×10^{-4}
Energy score, 1-step + residual	s1	never	—	0.0070	1.00	0.035
Energy score, 1-step + residual	s2	never	0/40	0.0105	1.00	4.5×10^{-4}
Energy score, 1-step + residual	s3	never	9/40	0.0133	1.00	0.112
Energy score, 5-step + residual	s0	never	1/40	0.0046	1.00	0.014
Energy score, 5-step + residual	s1	never	—	0.0050	1.00	5.5×10^{-5}
Energy score, 5-step + residual	s2	4,959	40/40	0.0524	1.00	0.696
Energy score, 5-step + residual	s3	never	7/40	0.0047	1.00	0.082
Thermalizer (released code)	s0	2,694	33/40	0.4574	1.00	0.014
Released emulator (thermalization off)	s0	2,563	30/40	0.4575	1.00	0
Thermalizer (released code)	s1	2,855	22/40	0.5183	1.00	3.0×10^{-4}
Released emulator (thermalization off)	s1	never	18/40	2.2363	1.00	0
Thermalizer (released code)	s2	1,582	34/40	0.4588	1.01	0.009
Released emulator (thermalization off)	s2	1,873	27/40	0.4587	1.00	0

Table 9: Metric deviation, measured on the paper's own published weights (seed 0): Eq. 31 reproduces the published number; the released channel-summed `nRMSE` does not. Identical rollouts and data.

Task	published	Eq. 31 (ours)	channel-sum (0.1.x)
2d_phy_gs_theta (2 ch.)	0.0659	0.0659	0.4648
2d_diff_burgers (2 ch.)	0.0535	0.0541	0.1086
2d_phy_aniso_diff (1 ch.)	0.0162	0.0163	0.0163

We report the components that failed; each rejected component was tested against a pre-specified failure hypothesis: a learned SDF anchor (supervision noise dominates at field scale); a latent anchor (inert where drift is in-subspace); an always-on projector (removes real high-wavenumber content); margin widening (a $4\times$ probe changes nothing: the 2D harm is structural); a correction cap (halves the harm, kept as the capped variant); and a symmetric amplifying band corrector (destabilizes chaotic tasks). The optional per-band correction for periodic domains fits the predictor's per-band log-energy drift bias on a calibration split (half the test initial conditions) and is scored on the disjoint other half; no headline result uses it.

D Cost accounting and controlled-environment details

APEBench rows use the benchmark's champion backbones (32k/61k/200k parameters; their Table 5); our long-horizon settings use 646k-parameter 1D and 52M-parameter 2D predictors. The residual update adds two projections through the stored principal directions and does not evaluate the learned network; the energy-score ensemble is training-only, so NFE/step is 1 vs. 2 for renoise/denoise. Measured wall-clock per 40-trajectory rollout step (one H200 shard, Kolmogorov

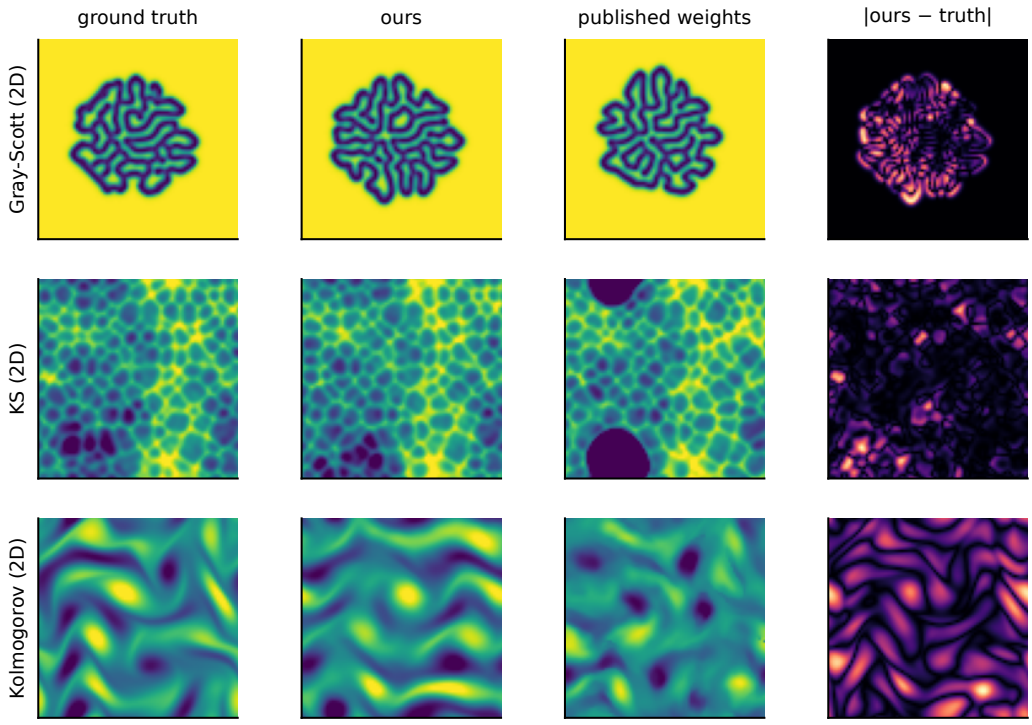


Figure 8: Qualitative rollouts at step 100 (one test initial condition, first channel where multi-channel). Rows: Gray–Scott, KS, Kolmogorov (2D). Color scales are per-row, shared between panels.

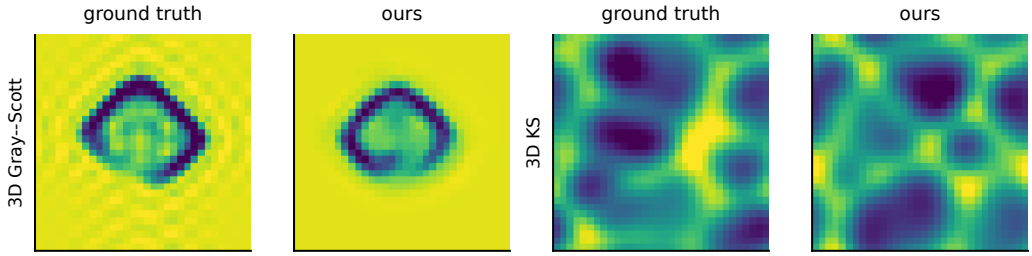


Figure 9: 3D rollouts, mid-plane slices: 3D Gray–Scott at step 5 and 3D KS at step 100 (reference and ours). The 3D Gray–Scott task is under-resolved at 32^3 (the benchmark's own note): its reference relaxes to the uniform state within $\sim$20 steps, so late snapshots are featureless for every model and we show the transient.

setting): 22.5 ms with the residual update versus 22.5 ms for re-noise/denoise—the halved NFE does not yield faster wall-clock in this batched implementation, and we report both quantities.

Analytic environment: energy distance at step 300 is -0.001 for the corrected model versus 0.016 for the uncorrected predictor; drift d_K 0.077 vs. 0.208; fire rate 12%. Learned-tube environment: surface fit $\mathrm{rmse}/r_\perp = 0.14$; contraction slope 0.41 at $4r_\perp$ excursions. Control environment: anchored drift d_K 0.071 vs. the reference's 0.072; planning through correction-carrying models fails for both corrected variants ($+25.5$/$+25.8$ vs. -1.4 in return).

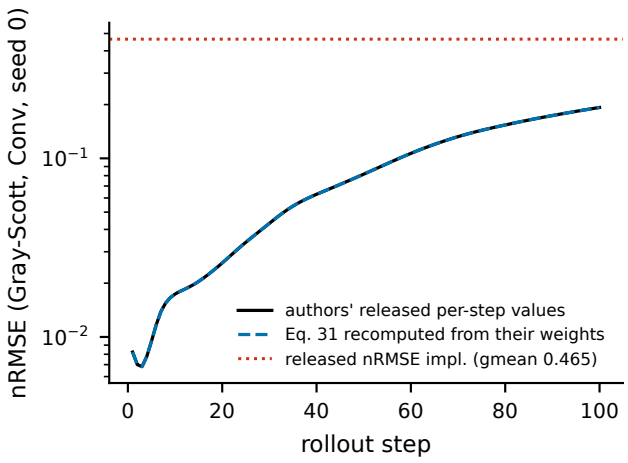


Figure 10: Metric cross-validation on the authors' released Gray–Scott weights (seed 0). Black: the per-step values in the released pipeline artifacts; blue: our recomputation under Eq. 31; red: the geometric mean of the released channel-summed implementation.

Table 10: Wrong-subspace control (same medians as Fig. 4b). A same-rank random subspace is inert; random directions with the PCA dead zone destroy the rollout.

Task	PCA	random	rand. dirs
1D dispersion	$+15.6$	$+0.0$	-2697
3D Gray–Scott	$+0.0$	$+0.0$	$+26.9$
2D KS	-745	$+0.0$	-490
2D Burgers	-1358	$+0.0$	-1705

E Implementation and hyperparameters

Table 11 distinguishes the four calibrations. Table 12 lists the settings needed to reproduce the pipeline.

Table 11: Four calibrations. W, μ and $(r_\perp, m)$ use training states. $D_\perp$ uses the full 30-IC test set after training, overlapping GMean@100 and the map's last-15 evaluation ICs, so the regime analysis is retrospective. The band correction is an appendix-only probe.

Quantity	Data	When	Used for
W, μ, d_W	training states	after training	residual geometry
$r_\perp, m$	training + held-out 10% of training	after training	update firing
$D_\perp$	100-step, 30 test ICs (overlap GMean)	after training	retrospective applicability analysis only
band correction	half of test ICs	after training	appendix-only probe

F Verification inventory

The verification suite comprises twelve checks—smoothing bound, mean collapse, escape-versus-projection, projector circle, telescoping value, gradient explosion, composition, solid-core comparability, accelerator consistency, the analytic anchor gate, viability on learned manifolds, and the theory chain—each with pre-registered pass thresholds, plus a smoke suite that exercises the full pipeline at toy scale. Each of these checks passes its pre-registered threshold.

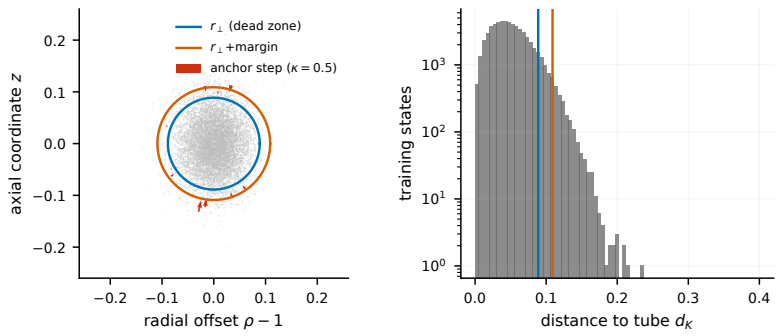


Figure 11: Analytic-environment residual geometry. Left: training states in the residual plane with the dead zone and margin; arrows show the contraction of Eq. 2. Right: distribution of distance-to-tube d_K over the training cloud.

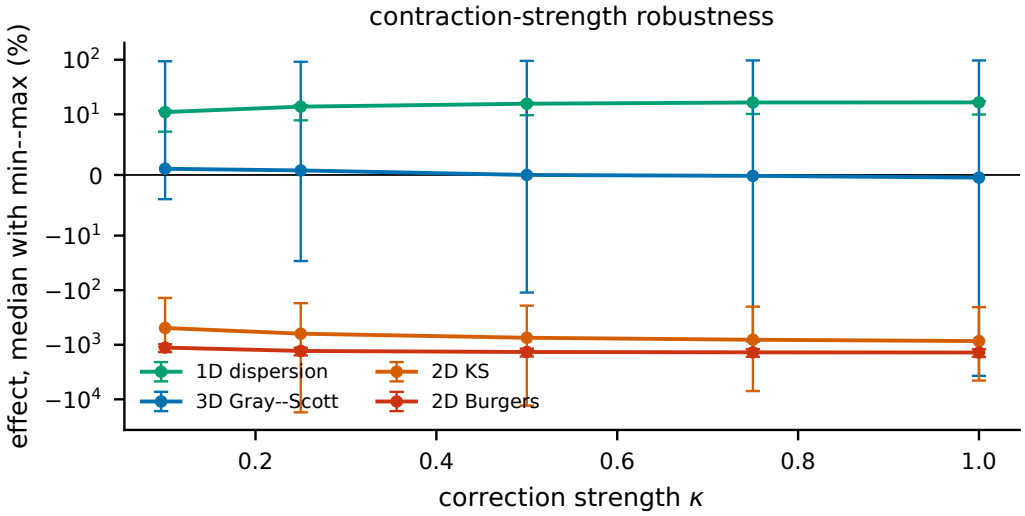


Figure 12: Contraction-strength robustness: median effect per task, with min–max whiskers. The 1D dispersion median effect is positive at every κ; the 2D harms persist at every κ. Wrong-subspace control is Fig. 4b.

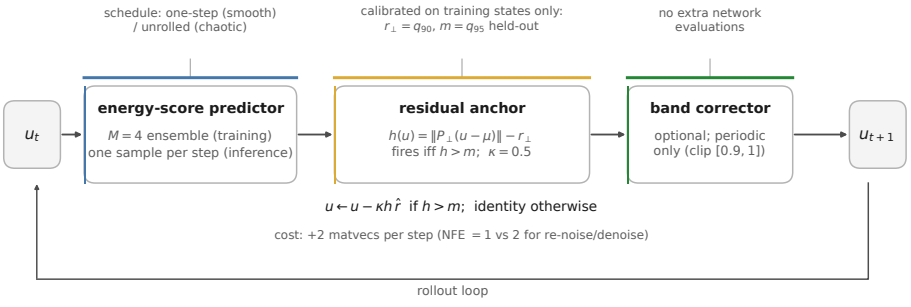


Figure 13: The full pipeline. Each rollout step applies the learned predictor and the dead-zoned residual update (Eq. 2). An optional spectral-band correction for periodic domains is appendix-only and is not used in any headline result. Only the predictor contributes network function evaluations.

Table 12: Hyperparameter inventory for the pipeline (values as used in the reported runs).

Component	Setting	Value
PCA tube	retained energy	99.95% of the training-cloud energy (rank cap 768, floor 16; Gram PCA on a ≤ 4000-row subsample)
PCA tube	dead zone $r_\perp$	q_{90} of the training residual norms $\|r(u)\|$
PCA tube	fire margin m	q_{95} of the held-out-10% residual spread $\big\| \|r\| - r_\perp \big|$
Anchor	contraction	$\kappa = 0.5$ (robust over 0.1–1.0, App. C); identity inside the tube; exact residual direction $r/\|r\|$
Anchor	cost	$+2$ matrix–vector products per step; NFE $= 1$ (predictor) vs. 2 (re-noise/denoise analog); residual update is not a network eval
Energy score	ensemble	$M = 4$ members; training-time only, one sample per inference step
Energy score	optimizer	AdamW, lr 3×10^{-4}, weight decay 10^{-5}, batch 1024, 30 epochs (mechanism suite)
APEBench	trainer	their released defaults: Adam, 10^4 steps, 2×10^3-step warmup, cosine decay to 0, peak lr 10^{-3}; one-step or five-step supervised unrolling; champion backbone per task
APEBench	evaluation	Eq. 31 nRMSE, GMean@100; 3 seeds (1D) / 10 (2D, 3D) against the published 50/20
Long-horizon	1D Kuramoto–Sivashinsky	64 test trajectories to horizon 600; 646k-parameter 1D predictor
Long-horizon	2D Kolmogorov flow	40 trajectories to $100{,}000$ steps; 52M-parameter 2D predictor
Schedules	predictor training	one-step supervision on smooth tasks, supervised unrolling on chaotic ones
Hardware	compute	H200 GPU shards (2 at a time); JAX/equinox for APEBench, PyTorch for the mechanism suite