
Plausible States Are Not Plausible Trajectories: Self-Supervised Objectives for Chaotic Dynamics

Ramon Daniel Regueiro-Espino¹ Jorge Mifsut-Benet¹ Nicolas Thome^{1,2,3} Patrick Gallinari^{1,4}

¹Sorbonne Université, CNRS, ISIR, F-75005 Paris, France

²Institut universitaire de France (IUF)

³ILLS, CNRS, Montreal, QC H2S 3H1

⁴Criteo AI Lab, Paris, France

regueiroespino@isir.upmc.fr

Abstract

Forecasting provides a natural self-supervised objective for learning models of dynamical systems. For chaotic systems, however, individual trajectories become unpredictable beyond a finite *predictability horizon*, and plausible states need not form plausible trajectories. Beyond this horizon, the objective should shift from predicting a particular future to generating trajectories that preserve both the statistics of physical states and their temporal evolution. We study this as an objective-design problem using a Jump-and-Complete framework that factorizes trajectory generation into two stochastic tasks: a *Jumper* samples plausible distant endpoints, and a *Completer* generates the paths connecting the observed context to those endpoints. Both are trained with the Energy Score and evaluated with state- and path-space diagnostics. On 2D Kuramoto–Sivashinsky and Kolmogorov flow, we first show that, past the predictability horizon, pointwise errors fail to distinguish models with very different temporal behavior. We then show that two seemingly auxiliary design choices shape the learned trajectory distribution. First, training the Completer on Jumper-sampled rather than exact endpoints changes which temporal statistics of composed rollouts are matched on Kolmogorov flow. Second, sharing one latent draw across a trajectory preserves temporal dependence better than resampling noise at each time, despite an equal or better Energy Score. Thus, for stochastic trajectory learning, the effective objective is determined not only by the loss, but also by what the model is conditioned on during training and how randomness is shared across time.

1 Introduction

Trajectories of physical systems provide their own supervision, making forecasting a natural self-supervised objective for learning dynamical models [Koehler et al., 2024, Takamoto et al., 2022]. Most work evaluates such models through ensemble forecast skill [Price et al., 2025] or rollout stability [Brandstetter et al., 2022, Lippe et al., 2023]. For chaotic systems, however, this raises a fundamental question: *what does the forecasting objective identify once the particular future becomes unpredictable?*

In chaotic PDEs, small differences in initial conditions grow exponentially, causing nearby trajectories to diverge and making a specific future increasingly difficult to predict [Boffetta and Musacchio, 2017]. The characteristic timescale of this divergence is the Lyapunov time T_L ; its definition and estimation are given in Appendix B.5. Beyond this predictability horizon, an imperfect surrogate cannot be expected to track the particular observed future. The objective should therefore shift from reproducing a single realization to generating *dynamically plausible futures*. This requires

two complementary forms of fidelity: generated states should reproduce the relevant state-space statistics, as targeted by invariant-measure matching methods [Pedersen et al., 2025, Schiff et al., 2024], while generated sequences should also reproduce the temporal dependencies of the underlying dynamics. We refer to the combination of these properties as *trajectory fidelity*. These properties are distinct because independent draws from the stationary distribution may produce plausible states while destroying temporal dependence.

This objective also motivates a stochastic formulation. Even when the underlying dynamics are deterministic, beyond the predictability horizon the available context does not identify a practically predictable unique future. Let x_t denote the state at time t , c_t the context available at that time, and H the forecast horizon in time steps. Rather than predicting a single future trajectory, we therefore aim to learn a conditional distribution over future trajectories,

$$p(x_{t+1:t+H} \mid c_t). \quad (1)$$

A deterministic regression model instead returns a single prediction, typically favoring conditional averages under squared-error training, which need not represent a typical or dynamically plausible realization.

We show that conventional pointwise forecasting metrics can fail to capture this objective. Once trajectories have decorrelated, RMSE approaches the error between unrelated realizations [Lorenz, 1969] and may assign similar scores to persistence forecasts, independent samples from the stationary distribution, and models that better reproduce temporal dynamics. Thus, evaluating long-horizon generation requires examining not only the realism of individual states but also the temporal structure of the generated trajectories: *plausible states are not sufficient for plausible trajectories*.

We study long-horizon forecasting as a self-supervised objective-design problem. We factor trajectory generation into two stochastic tasks: a *Jumper* samples a plausible distant endpoint conditioned on the observed context, and a *Completer* generates the trajectory conditioned on both the context and the sampled endpoint. Both are trained with the Energy Score (ES), a strictly proper scoring rule for probabilistic predictions. This construction also makes explicit two usually implicit methodological choices that can affect trajectory fidelity: the distribution of endpoints used to train the Completer, and how stochastic variation is represented and coupled across time along a generated path.

Related work has explored temporal coupling of stochasticity in generative sequence models. Ge et al. [2023] introduce correlated noise priors for video diffusion, while Andrae et al. [2025] correlate noise across forecast lead times to generate temporally consistent weather ensembles. Building on this perspective, we investigate the effect of shared versus independently sampled path latents in chaotic PDE models trained with a joint trajectory ES, examining whether similar training scores can conceal differences in temporal dependence.

Our experiments on two canonical chaotic PDEs, 2D Kuramoto–Sivashinsky and Kolmogorov flow, show that these design choices shape the trajectory distribution learned by the model. In particular, on Kolmogorov flow, training the Completer on Jumper-sampled rather than exact future states changes the statistics of composed rollouts. On both systems, independently resampling noise across time reduces temporal coherence despite an unchanged or better ES. Together, these results show that a loss function alone does not define the trajectory distribution a stochastic model learns: *the conditioning used during training and the way randomness is organized across time are part of the objective itself*.

2 Trajectory generation beyond the predictability horizon

Given a context $c_t = x_{t-L_c+1:t}$ of L_c observed states, our goal is to generate plausible future trajectories $x_{t+1:t+H}$. In our experiments, we use a single context frame ($L_c = 1$), since one full state determines the future evolution. We distinguish three notions of fidelity: *tracking fidelity* measures agreement with the observed future at corresponding times, typically using RMSE; *state-space fidelity* measures how well generated states reproduce the reference distribution, using statistics such as the spatial energy spectrum $E(k)$; and *path-space fidelity* measures whether generated trajectories preserve temporal dependencies, assessed through frame-to-frame correlations and temporal power spectra $P_t(f)$. Beyond the predictability horizon, state- and path-space fidelity provide alignment-free criteria for evaluating generated trajectories. Diagnostic definitions are given in Appendix A.

2.1 Jump-and-Complete

Rather than generating the entire future directly, we decompose its conditional distribution as

$$p(x_{t+1:t+H} \mid c_t) = p(x_{t+H} \mid c_t) p(x_{t+1:t+H-1} \mid c_t, x_{t+H}). \quad (2)$$

This exact factorization separates *where the system may evolve* from *how it gets there*. We realize these two factors with separate models: a *Jumper* generates a plausible future endpoint, and a *Completer* generates a dynamically plausible path connecting the observed context to that endpoint. This construction follows the forecaster–interpolator decomposition of Rühling Cachay et al. [2023], while allowing us to vary the two design choices affecting trajectory fidelity independently.

For a fixed horizon H , the stochastic Jumper and Completer are

$$\hat{x}_{t+H} = J_\theta(c_t, H, z), \quad \hat{x}_{t+1:t+H-1} = C_\phi(c_t, \tilde{x}_{t+H}, H, z'). \quad (3)$$

Here, z and z' are 16-channel Gaussian latent noise fields controlling stochasticity in endpoint and path generation, respectively. The variable $\tilde{x}_{t+H}$ denotes the endpoint used to condition the Completer, which depends on the training strategy. At inference, $\tilde{x}_{t+H} = \hat{x}_{t+H}$, and successive segments are chained autoregressively for long rollouts.

2.2 Training objective and design choices

Both the Jumper and Completer generate ensembles trained with the ES, a strictly proper scoring rule [Gneiting and Raftery, 2007] that balances target fidelity against ensemble diversity and has previously served as a training objective for generative forecasters [Pacchiardi et al., 2024]. For an ensemble $\{\hat{y}^{(m)}\}_{m=1}^M$ and target y , the loss is

$$\mathcal{L}_{\text{ES}} = \frac{1}{M} \sum_{m=1}^M \|\hat{y}^{(m)} - y\|_2 - \frac{1}{2M(M-1)} \sum_{m \neq n} \|\hat{y}^{(m)} - \hat{y}^{(n)}\|_2. \quad (4)$$

For the Jumper, the target is the future endpoint $y = x_{t+H}$. For the Completer, it is the entire intermediate trajectory $y = x_{t+1:t+H-1}$, scored jointly rather than frame by frame. Beyond this common training objective, we vary two design choices that may affect trajectory fidelity.

Conditioning distribution. At inference, the Completer is conditioned on endpoints generated by the Jumper, rather than exact future states. To investigate this training–inference mismatch, we compare training with true endpoints and with Jumper-generated endpoints. In the latter case, we sample several endpoints and select the one closest to the true endpoint, maintaining compatibility with the target trajectory while introducing endpoint prediction errors, albeit smaller than those encountered at inference.

Pathwise stochasticity. We investigate how stochasticity is coupled across the generated trajectory. By default, a single latent z' is shared across all time steps, coupling the stochastic generation of the entire path. We compare this with independent latents sampled at each time step. Both approaches can generate plausible marginal states, but they need not represent the same temporal joint distribution.

3 Experiments

We evaluate on the 2D Kuramoto–Sivashinsky (KS) and Kolmogorov flow (Kolmo) systems from APEBench [Koehler et al., 2024]. Unless stated otherwise, the jump horizon is $H = 16$ ($\approx 0.98 T_L$ for KS and $0.92 T_L$ for Kolmo). Data-generation and implementation details are given in Appendix B. Each experiment shows a temporal effect that standard scores miss: (i) tracking accuracy need not imply trajectory fidelity; (ii) on Kolmo, the training endpoint distribution affects the dynamics learned by the Completer; and (iii) temporal coupling of stochasticity matters even when the ES changes little.

3.1 Tracking Does Not Imply Trajectory Fidelity

We first test whether matched-time error remains informative beyond the predictability horizon. We compare a stochastic distributional Jumper, a deterministic RMSE-trained Jumper, persistence,

and IID climatology (independent draws from the empirical stationary distribution). The last two baselines fail in opposite ways: persistence imposes excessive temporal dependence, whereas IID climatology preserves the distribution of individual states but removes temporal dependence entirely.

For Kolmo with $H = 24$ ($\approx 1.4T_L$), Fig. 1 reports matched-time RMSE, the spatial energy spectrum $E(k)$, and one-jump correlation C_H , which measures the correlation between states separated by the forecast horizon H .

As forecasts decorrelate from the reference, models with markedly different dynamics approach similar RMSE. Persistence and IID climatology can therefore appear competitive under tracking error even though their temporal behavior is incorrect by construction. In contrast, C_H exposes these temporal failures. The ensemble mean also attains low RMSE while exhibiting a damped spectrum and incorrect temporal correlation.

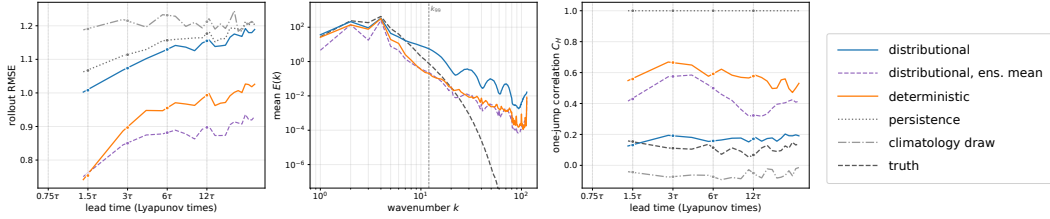


Figure 1: Tracking accuracy does not imply trajectory fidelity. Kolmo rollouts with $H = 24$. “Distributional” denotes stochastic Jumper samples and “ens. mean” their ensemble mean; “persistence” repeats the last observed state, while “climatology draw” independently samples from the empirical stationary distribution. Left: RMSE. Center: snapshot energy spectra $E(k)$. Right: one-jump correlation C_H .

Beyond the predictability horizon, tracking error cannot by itself distinguish plausible dynamics; state- and path-space diagnostics are needed. Below-horizon and cross-system controls appear in Appendix C.1 (Figs. 2–3).

3.2 Endpoint Conditioning Affects Trajectory Fidelity

At inference, the Completer receives a Jumper-generated endpoint, whereas standard training uses the exact future state. To examine this mismatch, we compare training with true endpoints, truth-compatible Jumper samples, and no endpoint conditioning (forecast).

We evaluate composed, autoregressively chained Jumper–Completer rollouts at leads $16H$ – $31H$, with $H = 16$. On Kolmo, RMSE is indistinguishable across strategies, but temporal diagnostics differ (Table 1): Jumper-conditioned training better matches adjacent-frame correlation and low-frequency temporal power, while oracle conditioning better matches high-frequency power. On KS, differences remain within across-seed variability (Table 9).

Table 1: Composed Kolmo rollouts over $16H$ – $31H$. Metrics are defined in Appendix A; “low”/“high” denote the lowest/highest temporal-frequency bands and “sm” small spatial scales. Lower RMSE is better; Δ metrics are better closer to zero. Large-scale $\Delta \log E(k) < 0.005$ for all strategies (within seed variability; Table 10).

	RMSE	$\Delta C_1 [10^{-3}]$	$\Delta \log P_{t,\text{low}}$	$\Delta \log P_{t,\text{high}}$	$\Delta \log E(k)_{\text{sm}}$
Jumper+Completer _{jumper}	0.877 ± 0.002	$+0.8 \pm 0.3$	-0.027 ± 0.004	$+0.075 \pm 0.010$	-0.004 ± 0.000
Jumper+Completer _{oracle}	0.877 ± 0.002	$+1.6 \pm 0.3$	-0.041 ± 0.003	$+0.014 \pm 0.012$	-0.012 ± 0.001
Forecast	0.877 ± 0.005	$+2.7 \pm 0.4$	-0.100 ± 0.011	$+0.090 \pm 0.003$	-0.015 ± 0.001

Endpoint perturbations also reveal different sensitivities in the learned representations: oracle training increases sensitivity to the terminal condition with depth; noisy-endpoint training reduces it; Jumper conditioning lies between them (Appendix C.3).

The endpoint distribution used during training influences both the Completer’s reliance on terminal information and the temporal statistics of its generated paths.

3.3 Temporal Coupling of Stochasticity Matters

Finally, we compare a Completer using one shared latent across all queried times with a variant that independently resamples its latent at each time step. Both can produce plausible individual states, but they impose different dependencies across the generated path.

Independent per-query noise increases adjacent-frame correlation error by roughly an order of magnitude, although its ES is similar or slightly better on both systems (Table 2).

Table 2: Interior completion with one shared path latent or independent per-query latents. Lower ES is better; ΔC_1 is generated–reference correlation; closer to zero is better.

Latent	Kolmo		KS	
	ES ↓	$\Delta C_1 [10^{-3}]$	ES ↓	$\Delta C_1 [10^{-3}]$
shared z	0.3307 ± 0.0035	2.3 ± 0.4	0.0318 ± 0.0004	-0.1 ± 0.0
per-query z	0.3214 ± 0.0013	-29.3 ± 2.0	0.0315 ± 0.0007	-1.0 ± 0.1

Scoring complete trajectories is not sufficient in practice to ensure temporal coherence; how randomness is coupled across time is itself a consequential modeling choice.

4 Discussion and Conclusion

We investigated long-horizon forecasting of chaotic systems from the perspective of generating dynamically plausible trajectories rather than tracking individual futures. Factoring generation into a Jumper that samples distant endpoints and a Completer that generates the connecting paths exposed two design choices that standard scores do not reveal. Beyond the predictability horizon, plausible states are insufficient for plausible dynamics: models with similar RMSE or ES can still generate trajectories with different temporal structure. Our results suggest three practical considerations for stochastic forecasting of chaotic systems: account for Jumper-generated endpoint errors when training the Completer, which changes which temporal statistics are matched rather than improving all of them; share each latent draw across the whole generated path; and evaluate with path-space diagnostics rather than pointwise error alone.

Our study is limited to two chaotic PDE systems and a specific Jump-and-Complete architecture. The effects of endpoint conditioning on composed rollouts are clearest on Kolmogorov flow and within seed variability on KS, whereas stochastic coupling affects temporal correlations on both systems. Future work should investigate these effects on broader chaotic systems and explore training objectives that more explicitly target temporal dependence.

Acknowledgments and Disclosure of Funding

We acknowledge the financial support provided by the Agence Nationale de la Recherche (ANR) under the France 2030 programme, reference 22-PETA-0002, DEEPNUM (ANR-21-CE23-0017-02), PHLUSIM (ANR-23-CE23-0025-02), MELISSA (ANR-24-CE23-7140-01), PEPR Sharp (ANR-23-PEIA0008, ANR, FRANCE 2030), and cluster IA PostGenAI@Paris (ANR-23-IACL-0007). This project was provided with computer and storage resources by GENCI at IDRIS thanks to the grants AD011015480R2, 2025-A0191016890 on the supercomputer Jean Zay’s V100/A100/H100 partitions.

References

- Martin Andrae, Tomas Landelius, Joel Oskarsson, and Fredrik Lindsten. Continuous ensemble weather forecasting with diffusion models. In *International Conference on Learning Representations*, volume 2025, pages 26392–26416, 2025.
- Guido Boffetta and Stefano Musacchio. Chaos and predictability of homogeneous-isotropic turbulence. *Physical review letters*, 119(5):054102, 2017.
- Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural PDE solvers. *arXiv preprint arXiv:2202.03376*, 2022.

- Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve your own correlation: A noise prior for video diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22873–22884. IEEE, 2023.
- Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.
- Felix Koehler, Simon Niedermayr, Rüdiger Westermann, and Nils Thuerey. APEBench: A benchmark for autoregressive neural emulators of PDEs. *Advances in Neural Information Processing Systems*, 37:120252–120310, 2024.
- Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.
- Edward N Lorenz. Atmospheric predictability as revealed by naturally occurring analogues. *Journal of Atmospheric Sciences*, 26(4):636–646, 1969.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.
- Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for spatiotemporal surrogate models. *Advances in Neural Information Processing Systems*, 37:119301–119335, 2024.
- Lorenzo Pacchiardi, Rilwan A Adewoyin, Peter Dueben, and Ritabrata Dutta. Probabilistic forecasting with generative networks via scoring rule minimization. *Journal of Machine Learning Research*, 25(45):1–64, 2024.
- Chris Pedersen, Laure Zanna, and Joan Bruna. Thermalizer: Stable autoregressive neural emulation of spatiotemporal chaos. *arXiv preprint arXiv:2503.18731*, 2025.
- Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025.
- Salva Rühling Cachay, Bo Zhao, Hailey Joren, and Rose Yu. DYffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *Advances in Neural Information Processing Systems*, 36:45259–45287, 2023.
- Yair Schiff, Zhong Yi Wan, Jeffrey B Parker, Stephan Hoyer, Volodymyr Kuleshov, Fei Sha, and Leonardo Zepeda-Núñez. DySLIM: Dynamics stable learning by invariant measure for chaotic systems. *arXiv preprint arXiv:2402.04467*, 2024.
- Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

A Diagnostic Definitions

For a field $x(r, t)$, let $\tilde{x}(r, t) = x(r, t) - \langle x(r, \cdot) \rangle_t$ denote the temporally centered signal; averages run over space, channels, and, for ensembles, batch. The spatially averaged temporal autocorrelation is

$$C_x(\tau) = \frac{1}{|\Omega|} \int_{\Omega} \frac{\langle \tilde{x}(r, t) \tilde{x}(r, t + \tau) \rangle_t}{\langle \tilde{x}(r, t)^2 \rangle_t} dr, \quad (5)$$

with lags restricted to the jump cadence for coarse chains and the reference trajectory subsampled identically. We additionally track the adjacent-frame correlation as a function of position in the rollout, $C_1(t) = \langle \tilde{x}_t \tilde{x}_{t+1} \rangle / \sqrt{\langle \tilde{x}_t^2 \rangle \langle \tilde{x}_{t+1}^2 \rangle}$, and its jump-cadence analogue $C_H(t) = \text{Corr}(x_t, x_{t+H})$. ΔC_1 denotes the generated-minus-reference value of C_1 , averaged over a lead window. Both records span the same lead times and are centered identically, so the finite-record bias cancels in the difference. For interior completion, ΔC_1 is computed identically on completed segments and their matched true segments at frame cadence.

The temporal power spectral density is

$$P_t(f) = \frac{1}{|\Omega|} \int_{\Omega} |\mathcal{F}_t[x(r, \cdot)](f)|^2 dr, \quad (6)$$

and $E(k)$ is the radially averaged spatial power spectrum of individual snapshots. $\Delta \log P_t$ and $\Delta \log E(k)$ denote the $\log_{10}$ ratio of generated to reference power, averaged within frequency and wavenumber bands. For $\Delta \log P_t$, the positive-frequency axis of each lead window is split into thirds. The lowest and highest thirds are reported and the middle third is dropped. The high band, periods of 2–3 frames up to the Nyquist frequency, is identical across windows, whereas the low band extends from each window’s fundamental frequency to periods of approximately 6 frames ($f \in [0.0078, 0.164]$ cycles per frame for the $< 8H$ window and $[0.0042, 0.167]$ for the $16H$ – $31H$ window).

For $\Delta \log E(k)$, the resolved range is bounded by k_{99} , the wavenumber below which the reference field holds 99% of its spectral energy, and split at $k_{99}/3$. This gives large scales $k \in [1, 4]$ and small scales $k \in [5, 12]$ on Kolmo ($k_{99} = 12$), and $k \in [1, 5]$ and $[6, 15]$ on KS ($k_{99} = 15$). The under-resolved tail $k > k_{99}$ is excluded.

B Data, Implementation, and Training Details

B.1 Data Generation

Both systems are generated with APEBench [Koehler et al., 2024] on a periodic square domain at 160×160 , single channel, from Fourier initial conditions (`fourier;5>true>true`) after 500 warm-up steps discarded onto the attractor. We store 1000 training and 50 validation trajectories of 51 frames, and 50 test trajectories of 501 frames, with disjoint seeds.

The stored spacing is $\Delta t = 0.2$, twice the APEBench default. We obtain it by scaling *both* `dt` and `num_substeps` by two, so the inner solver step is unchanged and only the sampling is coarsened (Table 3). The factor is chosen from the measured largest Lyapunov exponent (Sec. B.5) so that the two systems are comparable on a common predictability axis: with $T_L \approx 16$ – 17 stored frames, a training window of 51 frames spans $\approx 3T_L$, a test trajectory $\approx 30T_L$, and the horizon $H = 16$ is $0.92T_L$ (Kolmo) and $0.98T_L$ (KS). Kolmo is forced 2D Navier–Stokes on $[0, 2\pi]^2$ with injection at mode 4 (scale 1.0), diffusivity 10^{-2} and drag -0.1 . KS is solved on $[0, 60]^2$ in the $a = (0, 0, -1, 0, -1)$, $b = (0, 0, -1)$ form.

Table 3: Generation parameters. `dt` and `num_substeps` are scaled together, leaving the inner solver step at the APEBench default.

	Kolmo	KS
Domain L	2π	60
Resolution	160^2	160^2
Stored Δt	0.2	0.2
<code>num_substeps</code>	40	2
Inner solver step	0.005	0.1
λ (per time)	0.2882	0.3076
T_L (frames)	17.35	16.25
H/T_L at $H = 16$	0.92	0.98

B.2 Architecture

Shared implementation and backbone. Both the Jumper J_θ and Completer C_ϕ are implemented in PyTorch with the same AViT-style spatial-temporal backbone inspired by MPP McCabe et al. [2024] and summarized in Table 4. A physical state is first projected channel-wise by a 1×1 convolution, embedded by a convolutional hMLP stem into D latent channels, processed by a stack of L space-time blocks, and decoded to the physical variables with an hMLP output head. Channel ordering is fixed for each physical system. Per-channel normalization statistics are computed only from the training split, cached, stored in the checkpoint, and reused unchanged for validation and test evaluation. The implementation supports distributed data-parallel training, automatic mixed precision, ensemble chunking, and gradient checkpointing of ensemble chunks.

Jumper architecture. Given context $c_t = x_{t-L_c+1:t}$, the Jumper represents $p(x_{t+H} | c_t)$. In the distributional model, the encoded context is augmented with one learned future-query token. For each ensemble member, a latent noise field z_m is projected to the model dimension and injected into this future query (additively in the experiments), after which the full space-time stack produces

$$\hat{x}_{t+H}^{(m)} = J_\theta(c_t, H, z_m).$$

Thus different latent draws produce different candidate endpoints while sharing the same observed context. The deterministic ablation removes the latent injection and uses the same backbone and forecasting interface with a single future query.

Completer architecture. The Completer represents the second factor in Eq. (2), $p(x_{t+1:t+H-1} | c_t, \tilde{x}_{t+H})$, and predicts the interior jointly rather than frame by frame. Tokens are ordered chronologically as

$$[c_t | q_1, \dots, q_K | \tilde{x}_{t+H}],$$

where the experiments use the interior of the jump and each query corresponds to a fixed fractional lead $s_i = k_i/H$. Query i is initialized as

$$q_i = q_{\text{int}} + \phi(s_i) + (1 - s_i)h_t + s_i h_{t+H},$$

where q_{int} is a learned interior-token identity, ϕ is an MLP applied to a sinusoidal lead-time embedding, and h_t, h_{t+H} are the latent encodings of the last observed state and the conditioning endpoint. This latent interpolation provides a bridge between the boundary states, while the transformer learns the dynamical deviation from that bridge.

A single latent noise field z_m is shared by all interior queries belonging to ensemble member m . Consequently, one draw defines one complete path rather than a collection of independently perturbed frames. In bridge mode the final token is the encoded, given endpoint plus a learned endpoint identity. For the bridge-forecast and forecast-only variants, an additional learned endpoint-query token allows the endpoint to be withheld and predicted jointly with the interior; the same path latent is used for both the predicted endpoint and its interior trajectory.

Table 4: Shared architecture for the Jumper and Completer.

Hyperparameter	Value
Patch size	8×8
Embedding dimension	384
Attention heads	12
Space-time blocks	8
Spatial / temporal attention	axial attention / attention
Relative positional bias	yes
Latent noise dimension	16

B.3 Objectives and Conditioning

Distributional objectives. The distributional Jumper is trained with the ES in the high-dimensional endpoint space, as in Eq. (4). The Completer uses the same proper scoring rule on the *joint path*: the K predicted interior states are treated as one structured target,

$$\mathcal{L}_{\text{comp}} = \text{ES}\left(\{\hat{x}_{t+1:t+H-1}^{(m)}\}_{m=1}^M, x_{t+1:t+H-1}\right).$$

Deterministic Jumper and Completer ablations use the corresponding pooled point-prediction RMSE objective in the supplied training implementation.

Completer conditioning regimes. We implement the different regimes through two independent training switches. First, the endpoint used on bridge steps is either the oracle future state x_{t+H} or a Jumper endpoint. In the latter case, 4 candidate endpoints are precomputed from a trained Jumper and the candidate nearest in ℓ_2 to the true x_{t+H} is selected, so that the conditioning endpoint remains compatible with the true interior segment used as supervision. For the oracle + noise control, we perturb the true endpoint with white Gaussian noise whose per-example scale matches the normalized RMSE of the corresponding Jumper endpoint, preserving error magnitude while removing its spatial structure. Second, a configurable bridge-forecast mixing probability determines whether the endpoint is supplied or withheld on a given optimization step. This yields: (i) oracle bridge training, (ii) Jumper-conditioned bridge training, (iii) Jumper-conditioned bridge plus endpoint-free forecast training, and (iv) forecast-only training. Validation uses the task appropriate to the trained model and otherwise retains the true endpoint for bridge evaluation, preventing a model from being selected merely because its validation conditioning matches a particular sampled-endpoint cache.

B.4 Optimization and Evaluation

Optimization. Both model families use AdamW [Loshchilov and Hutter, 2019]; the training hyperparameters are summarized in Table 5. Weight decay is applied to matrix/tensor weights but not to one-dimensional parameters or learned token parameters. Training uses bf16 autocast, and the best checkpoint is selected by the configured validation metric.

Table 5: Training configuration. Batch sizes are per GPU.

Hyperparameter	Jumper	Completer
Optimizer	AdamW, $\beta = (0.9, 0.95)$	
Learning rate	3×10^{-4}	
Weight decay	0.05	
Schedule	1000-step warmup, cosine to 10^{-6}	
Gradient clipping	1.0	
Training steps	120k	30k
Batch size	4	2
Training ensemble M	8	4
Interior states	–	15

Ensemble sizes. The stochastic Jumper objectives use $M = 8$ ensemble members during training and $M = 32$ during evaluation; the deterministic RMSE Jumper uses $M = 1$. The stochastic

Completer uses $M = 4$ during training and $M = 16$ during evaluation. The deterministic Completer uses a single prediction under teacher, persistence, and forecast evaluation; under composed and composed-interior evaluation, it is evaluated on 16 independently sampled Jumper endpoints, while remaining deterministic conditional on each endpoint.

B.5 Lyapunov-Time: Definition and Estimation

The largest Lyapunov exponent characterizes the asymptotic exponential growth rate of infinitesimal perturbations,

$$\lambda_{\max} = \lim_{t \rightarrow \infty} \frac{1}{t} \log \frac{\|\delta x(t)\|}{\|\delta x(0)\|}.$$

We define the Lyapunov time as $T_L = \lambda_{\max}^{-1}$, the characteristic e -folding time of perturbation growth.

We estimate $\lambda_{\max}$ for each reference PDE using the standard Benettin procedure, averaging over four training initial conditions. A random tangent perturbation is propagated through each simulator step and renormalized after every step. We discard the first 300 tangent-growth steps to allow alignment with the leading Lyapunov vector and average the log growth over the subsequent 1500 steps. If g_n denotes the tangent-norm growth during one stored-frame step, we report

$$\lambda_{\text{frame}} = \langle \log g_n \rangle, \quad \lambda_{\text{time}} = \frac{\lambda_{\text{frame}}}{\Delta t}, \quad T_L^{(\text{frames})} = \lambda_{\text{frame}}^{-1}.$$

Here λ_{time} is the exponent per unit simulation time, whereas λ_{frame} expresses the growth rate per stored frame and is used to express forecast horizons in units of Lyapunov time.

C Additional Experiments and Controls

C.1 Additional Controls for Tracking vs. Trajectory Fidelity

Figure 2 repeats the comparison of Fig. 1 at $H = 16$ ($\approx 0.92 T_L$), below the predictability horizon. The tracking–temporal decoupling is absent, with both learned Jumpers agreeing on matched-time RMSE and matching the reference consecutive-jump correlation, while persistence and the climatology draw remain clearly separated on RMSE. It is therefore a property of the post- T_L regime, not of the models or the diagnostics.

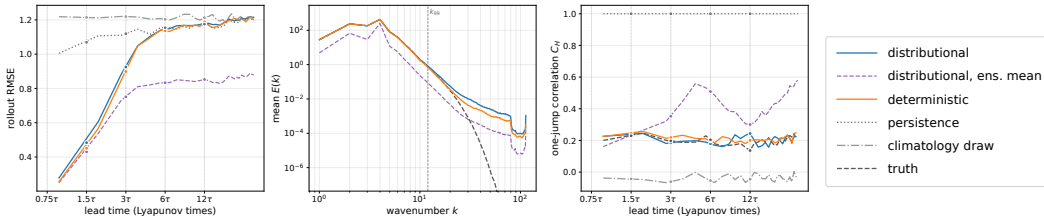


Figure 2: **Below- T_L control on Kolmo.** Chained rollouts at $H = 16$ ($\approx 0.92 T_L$). *Left*: matched-time rollout RMSE. *Center*: snapshot energy spectra $E(k)$; both learned Jumpers match the reference over the resolved range, while the ensemble-mean damping persists, reflecting decorrelation accumulated over chained jumps rather than the per-jump horizon. *Right*: consecutive-jump correlation $C_H(t)$.

Figure 3 reproduces the tracking–temporal mismatch of Fig. 1 on KS at $H = 16$ ($\approx 0.98 T_L$). Persistence approaches the deterministic Jumper in long-horizon matched-time RMSE while remaining maximally over-persistent ($C_H = 1$); both learned Jumpers closely reproduce the temporal dependence of the reference process.

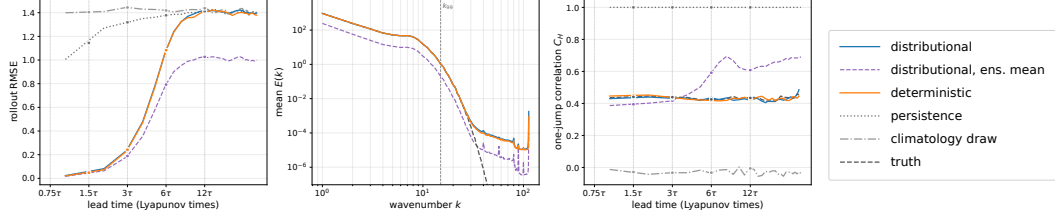


Figure 3: **Metric decoupling on KS.** Chained rollouts at $H = 16$ ($\approx 0.98 T_L$); panels and averaging as in Fig. 1. All jump maps reproduce the reference spectrum (center); persistence approaches the learned Jumpers in long-lead RMSE (left) while remaining maximally over-persistent, $C_H = 1$ (right).

C.2 Completer Training and Conditioning Ablations

Conditioning-distribution ablation. During composed inference, the Completer receives endpoints sampled by the Jumper rather than exact future states. We therefore compare Completers trained with oracle endpoints, noise-perturbed oracle endpoints, and truth-compatible Jumper samples. Table 6 reports interior completion Energy Score under both oracle and Jumper conditioning. Oracle training performs best when evaluated with the true endpoint, whereas Jumper-conditioned training is better matched to the composed setting. On Kolmo, it reduces ES under Jumper conditioning from 0.3833 to 0.3307; on KS, it matches the best noise-augmented variant within seed variability.

Table 6: **The conditioning distribution is part of the objective.** Interior completion ES under oracle and Jumper endpoints. Lower is better.

Train endpoint	Kolmo		KS	
	Oracle	Jumper	Oracle	Jumper
Oracle	0.1700 $\pm$ 0.0038	0.3833 $\pm$ 0.0026	0.0226 $\pm$ 0.0007	0.0336 $\pm$ 0.0005
Oracle + noise	0.1819 $\pm$ 0.0045	0.3651 $\pm$ 0.0017	0.0235 $\pm$ 0.0011	0.0315 $\pm$ 0.0008
Jumper	0.2203 $\pm$ 0.0035	0.3307 $\pm$ 0.0035	0.0258 $\pm$ 0.0005	0.0318 $\pm$ 0.0004

Bridge-forecast task ablation. We additionally ablate whether trajectory completion benefits from an endpoint-free forecasting objective. Table 7 compares bridge-only training with a mixture of bridge and forecast steps and with forecast-only training. All models are evaluated under composed inference, where the endpoint is sampled from the Jumper. Bridge-only training performs best on both systems. Adding the endpoint-free objective degrades completion regardless of whether bridge steps use Jumper or oracle endpoints, and forecast-only training performs worst. We therefore use bridge-only training for the conditioning-distribution experiments in the main text.

Table 7: **Bridge-forecast task ablation.** Interior completion Energy Score under composed inference with Jumper-sampled endpoints. Lower is better. The train endpoint specifies the endpoint supplied on bridge training steps; forecast steps withhold the endpoint. Single-seed runs from the task-ablation sweep; main-text values are three-seed means.

Training task	Train endpoint	Kolmo	KS
Bridge only	Jumper	0.3273	0.0317
Bridge only	Oracle	0.3818	0.0330
Bridge + Forecast	Jumper	0.3797	0.0427
Bridge + Forecast	Oracle	0.3965	0.0433
Forecast only	—	0.4614	0.0509

Endpoint-compatibility ablation. To verify that the Completer uses its conditioning endpoint rather than ignoring it, Table 8 evaluates the same Completer trained on Jumper-sampled endpoints under three inference-time conditioning choices: the true future endpoint x_{t+H} (teacher), a Jumper-sampled endpoint (composed), and the current context state x_t (persistence endpoint). ES is computed over the interior trajectory slots only, so oracle conditioning cannot improve the score merely by

supplying the exact final state. The persistence endpoint—individually plausible but temporally incompatible with the target trajectory—increases interior ES substantially on both systems, showing that the Completer is sensitive to the temporal compatibility of the supplied endpoint rather than using it as an arbitrary state-space signal.

Table 8: **Interior completion depends on endpoint compatibility.** Interior completion ES (lower is better) for Completers trained on Jumper-sampled endpoints. Single-seed runs from the task-ablation sweep; main-text values are three-seed means.

System	Teacher endpoint	Jumper endpoint	Persistence endpoint (x_t)
Kolmo	0.2165	0.3273	1.8580
KS	0.0257	0.0317	2.0350

C.3 Endpoint Sensitivity of Latent Representations

We measure how perturbations of the terminal boundary propagate through the Completer’s hidden representations. Let $r_{\ell,h}(c, e, z)$ denote the latent representation at bridge position h after block ℓ , conditioned on context c , endpoint $e = x_{t+H}$, and path latent z . Holding c and z fixed, we perturb only the endpoint, $e' = e + \delta$, and define

$$S_{\ell,h}^{\text{rel}} = \frac{\|r_{\ell,h}(c, e + \delta, z) - r_{\ell,h}(c, e, z)\|_2 / \|r_{\ell,h}(c, e, z)\|_2}{\|\delta\|_2 / \|e\|_2}. \quad (7)$$

This dimensionless local gain is averaged over test examples, perturbations, and training seeds. Values above one indicate amplification of relative endpoint perturbations.

Figure 4 shows that oracle conditioning produces stronger, depth-dependent endpoint sensitivity, while training with imperfect endpoints suppresses this amplification. Jumper-conditioned training yields an intermediate regime, indicating that the training-time endpoint distribution changes how terminal-boundary information is organized through the latent hierarchy.

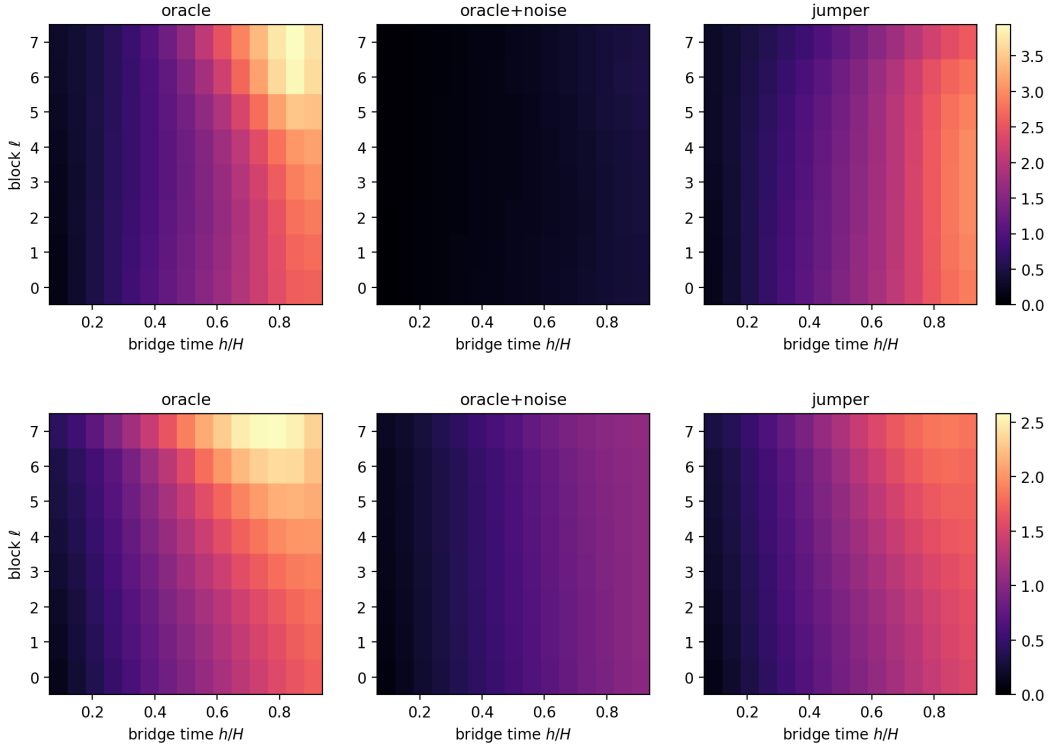


Figure 4: Relative endpoint sensitivity $S_{\ell,h}^{\text{rel}}$ across bridge time and network depth for different endpoint-conditioning distributions. **Top:** Kolmo. **Bottom:** KS.

C.4 Additional Composed-Rollout Diagnostics

Probe separation on KS. Table 9 shows the complementary regime discussed in Sec. 3.2. Matched-time RMSE again becomes nearly indistinguishable across completion strategies at long lead, but here the temporal diagnostics also show little separation: adjacent-frame correlations agree with the reference to within approximately 10^{-4} , and discrepancies in the coarse temporal spectrum remain small for all methods. The clear path-space separation seen on Kolmo is absent: second-order probes can expose temporal discrepancies when such differences are present, but need not distinguish surrogate rollouts whose low-order temporal statistics are already similar.

Table 9: Composed rollouts on KS over lead times $16H$ – $31H$. Δ measures generated–reference discrepancy; closer to zero is better. $\Delta \log E(k)$ at large scales is < 0.02 and similar across strategies (within across-seed variability; Table 10), and is therefore omitted here. Definitions in Appendix A.

	RMSE	$\Delta C_1 [10^{-3}]$	$\Delta \log P_{t,\text{low}}$	$\Delta \log P_{t,\text{high}}$	$\Delta \log E(k)_{\text{sm}}$
Jumper+Completer _{jumper}	1.021 ± 0.002	0.0 ± 0.0	$+0.007 \pm 0.002$	$+0.009 \pm 0.004$	-0.004 ± 0.000
Jumper+Completer _{oracle}	1.021 ± 0.002	0.0 ± 0.0	$+0.007 \pm 0.002$	$+0.008 \pm 0.004$	-0.005 ± 0.001
Forecast	1.023 ± 0.000	-0.1 ± 0.0	$+0.006 \pm 0.006$	$+0.009 \pm 0.013$	-0.006 ± 0.002

Table 10 collects the large-scale spectral discrepancies for both systems; they are similar across completion strategies, consistent with the bounds stated in Tables 1 and 9.

Table 10: Large-scale spatial spectral discrepancy for composed rollouts over lead times $16H$ – $31H$. $\Delta \log E(k)_{\text{large}}$ at large scales. Values close to zero indicate agreement with the truth.

Rollout	Kolmo	KS
Jumper+Completer _{jumper}	$+0.002 \pm 0.003$	$+0.014 \pm 0.009$
Jumper+Completer _{oracle}	$+0.001 \pm 0.003$	$+0.014 \pm 0.010$
Forecast	-0.004 ± 0.003	$+0.009 \pm 0.008$