INVERTIBLE CONTINUOUS LATENT DYNAMIC FOR LONG-TERM DATA ASSIMILATION IN COMPLEX PHYSICAL SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Forward forecasting and data assimilation are the two important aspects in physical simulation: one propagates the state forward, the other recovers unknown states from sparse observations. Learned surrogates are normally built and benchmarked for forward forecasting, however, whether a surrogate could attain good performance in data assimilation tasks is valuable as well, as inverse problems are of paramount importance in the scientific domain. In this paper, we propose a continuous-time Koopman autoencoder whose latent dynamics obey $\frac{dz}{dt} = \mathbf{K}_{\text{cont}} z$, yielding closed-form inference via $z(\tau) = \exp(\mathbf{K}_{\text{cont}}\tau)z(0)$ at any horizon τ in a single step. This decouples forecast cost from forecast length at inference time, showing long-term stability and high efficiency in forward simulation, and also supports data assimilation as gradient-based optimization with cost independent of the assimilation window. Experiments are performed on the Kuramoto–Sivashinsky equation and a transient flow, and we compare our method against a range of baselines on the forward problem, including diffusion models and operator-learning models, and obtain a $110\times$ inference speedup over strong diffusion baselines. We further test these baselines on an initial-state inference data assimilation task, and find that a strong forecaster does not guarantee a strong assimilator, while the continuous-time Koopman autoencoder achieves both higher accuracy and efficiency than surrogates of comparable forward performance.

1 INTRODUCTION

There are two complementary problems in physical simulation. The forward problem focuses on forecasting the long-term evolution of physical systems, such as turbulent flows and atmospheric dynamics, which requires models that remain stable over hundreds of steps. Traditional numerical solvers (Ghasemian et al., 2017; Slotnick et al., 2014) are prohibitively expensive at these horizons. Data-driven surrogates based on autoregressive rollouts (Li et al., 2021; Kohl et al., 2026) provide a cheaper alternative, but accumulate errors step-by-step, with inference cost scaling linearly with forecast length. Data assimilation aims to recover unknown states from sparse observations. Classical approaches use a numerical solver as the forward model and are correspondingly slow, while learned forward surrogates inherit the same autoregressive rollout and error accumulation (Tang et al., 2020; Chattopadhyay et al., 2022).

To address this, we propose a continuous-time Koopman autoencoder based on Koopman theory (Koopman, 1931). By embedding the nonlinear state into a latent space with linear dynamics, $dz/dt = \mathbf{K}\text{cont}z$, the state at any future time τ is given by $z(\tau) = \exp(\mathbf{K}\text{cont}\tau)z(0)$—a single matrix-vector product regardless of horizon. This enables data assimilation as a single-step gradient-based optimization for initial-state recovery, with cost independent of τ. Unlike prior Koopman methods (Lusch et al., 2018; Azencot et al., 2020), which learn a discrete operator tied to a fixed Δt, our continuous-time formulation naturally handles irregularly sampled observations without resampling or interpolation.

In this paper, we study forward forecasting and data assimilation together, comparing a range of models on both tasks. On the forward problem, we evaluate forecasting accuracy, inference speed,

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.

and long-term stability; on the data assimilation problem, we evaluate initial-state recovery across a range of settings. Our contributions are:

- We introduce a continuous-time Koopman autoencoder whose linear latent dynamics admit a closed-form solution, enabling single-step inference at any horizon. We demonstrate that it handles irregular sampling and unseen time steps.
- On realistic fluid flows and chaotic systems, it remains stable over long rollouts at a fraction of the inference cost of diffusion baselines.
- We demonstrate that our model enables fast and accurate data assimilation, outperforming autoregressive and score-based baselines at initial-state recovery with a cost independent of the assimilation window.

2 Related work

Koopman autoencoders. Koopman theory (Koopman, 1931) represents nonlinear dynamics as a linear operator acting on observables, while Koopman autoencoders (KAEs) learn these observables with an autoencoder whose latent dynamics are linear (Takeishi et al., 2017; Lusch et al., 2018; Morton et al., 2018; Otto and Rowley, 2019). Many variants have since been proposed (Azencot et al., 2020; Nayak et al., 2025; Xiong et al., 2024; Halder et al., 2026), but most learn a discrete operator $\mathbf{A}$ tied to the training timestep Δt. Continuous-time latent dynamics, as in neural ODEs (Chen et al., 2018), allow irregular time steps but typically require numerical integration. In contrast, our linear latent dynamics admit a closed-form solution. Related continuous-time Koopman models (Menier et al., 2025; Buzhardt et al., 2025) either retain nonlinear terms requiring integration or focus on highly idealized, low-dimensional systems.

Forward surrogates and data assimilation. Neural operators such as FNO (Li et al., 2021) and convolutional U-Nets (Ronneberger et al., 2015) provide efficient autoregressive forecasting, but error accumulation limits long-horizon reliability (Brandstetter et al., 2022). Diffusion models (Kohl et al., 2026) improve sample quality but incur stochastic error accumulation and high per-step cost. Learned surrogates have also been used to replace numerical solvers in ensemble and variational data assimilation (Chattopadhyay et al., 2022; Penny et al., 2022; Adrian et al., 2025; Hatfield et al., 2021). Frion et al. (2023; 2024) showed that a Koopman autoencoder can provide an effective prior for variational assimilation, but focused on low-dimensional, near-periodic satellite time series with a regularized discrete-time operator.

3 Background on latent models and data assimilation

3.1 Continuous-Time Koopman Autoencoders

Koopman operator theory shifts focus from nonlinear state-space evolution $x_{t+\Delta t} = f(x_t)$, $x_t \in \mathbb{R}^{N_d}$, to an infinite-dimensional space of measurement functions where dynamics are linear. Since finding a finite invariant subspace analytically is intractable for chaotic PDEs, Koopman Autoencoders (KAEs) learn this mapping from data: an encoder $\mathcal{E}$ maps the state to a latent representation $z_t \in \mathbb{R}^{N_z}$, a finite-dimensional operator propagates the latent dynamics, and a decoder $\mathcal{D}$ reconstructs the physical state.

Standard discrete-time KAEs learn a fixed-step operator $\mathbf{A}$ such that $z_{t+\Delta t} = \mathbf{A} z_t$, tying the latent dynamics to the training resolution and requiring regular sampling. Our approach instead trains the continuous generator directly, with latent dynamics governed by a linear ODE whose exact solution is $z(t_0 + \tau) = e^{\mathbf{K}_{\text{cont}}\tau} z(t_0)$. This closed-form solution enables single-step inference at any horizon, independent of the training timestep, and naturally accommodates irregularly sampled observations. A full derivation is provided in Appendix A.

3.2 Data assimilation with a latent based model

In data assimilation, we infer a physical state $x \in \mathbb{R}^{T \times P}$ from observations $y = H(x) + \varepsilon$, where H is the observation operator and ε the observation noise. This can be formulated as Bayesian inference

over a *control variable* c, which determines the full state through a mapping $c \mapsto x(c)$. The posterior is

$$\log p(c \mid y) = \log p(y \mid x(c)) + \log p(c) - \log p(y), \tag{1}$$

where the first two terms correspond to the likelihood and prior.

We consider two main families of data assimilation methods, which differ primarily in their choice of control variable and mapping $c \mapsto x(c)$.

Score model based assimilation. Score-based methods Rozet and Louppe (2023); Chung et al. (2023) use a noisy version of the full spatio-temporal state as the control variable, $c \triangleq x_\sigma$. A generative model provides the prior score $\nabla_c \log p_\sigma(c)$, which is combined with the likelihood gradient to steer the state towards the posterior.

Simulation based assimilation. Simulation-based methods instead use a low-dimensional control variable, typically the initial state x_0, which a forecasting model maps to the full trajectory $x \in \mathbb{R}^{T \times P}$. This includes 4D-Var (Bouttier and Courtier, 2002; Navon, 2009), using either numerical solvers (Courtier et al., 1994) or learned emulators (Fablet et al., 2021). Optimisation is performed on x_0, with gradients propagated through the forecast. Although effective, this can be computationally expensive and suffer from gradient instabilities over long assimilation windows.

Emulator-based assimilation also imposes an *implicit prior*: the assimilated states are restricted to those reachable by the forecasting model. This can enforce physically consistent trajectories, but limits assimilation to states within the model's predictive range.

Optimisation for data assimilation. All three approaches share the same optimisation backbone (Algorithm 1), updating the control variable using the prior and likelihood gradients. They differ only in the choice of control variable and mapping $c \mapsto x(c)$. For emulator-based assimilation, the central challenge is therefore computing gradients through the forecasting model, which we discuss next.

assimilate(y, c_{init})

1: input: y, c_{init}, $p(c)$, η, K
2: $c \leftarrow c_{\text{init}}$
3: **for** $k = 1, \dots, K$ **do**
4: $\quad \nabla_p \leftarrow \nabla_c \log p(c)$
5: $\quad \nabla_l \leftarrow \nabla_c \log p(y \mid x(c))$
6: $\quad c \leftarrow c + \eta\,(\nabla_p + \nabla_l)$
7: **end for**
8: **return** $x(c)$

Likelihood

$$\log p(y \mid x(c)) = -\tfrac{1}{2} \left\| y - H(x(c)) \right\|^2_{R^{-1}} + \text{cst}$$

	c	$x(c)$
Generative	x_σ	$\mathbb{E}[x \mid x_\sigma = c]$
Forecasting	x_0	$\mathcal{M}(c, \mathbf{t})$
Latent	z_0	$\psi(\mathcal{M}_z(c, \mathbf{t}))$

Figure 1: Optimisation backbone shared by the three assimilation methods (left), and their choice of control variable c and mapping $x(c)$ (right).

3.3 Sensitivity in forecasting models

As described above, forecasting-based assimilation requires the gradient of the log-likelihood $\log p(y \mid x(c))$ with respect to the control variable, which means differentiating through the forecasting model $\mathcal{M}$. By the chain rule, this gradient is the product of the likelihood gradient with respect to the state and the *sensitivity* of the forecast to its initial condition, $\partial x_t / \partial x_0$. Beyond data assimilation, forecast sensitivities are used in many applications, for instance to estimate the impact of individual observations on a forecast (Langland and Baker, 2004), to compute optimal perturbations (singular vectors) for ensemble forecasting (Buizza and Palmer, 1995), or to quantify the uncertainty of an estimated initial state.

Sensitivities can be computed for any differentiable autoregressive model with backpropagation through time, but a latent linear model has a strong advantage for this task. For an autoregressive model $x_{k+1} = f(x_k)$ with time step Δt, and a Koopman model with encoder ϕ, decoder ψ and continuous latent dynamic $z_t = e^{At} z_0$, $z_0 = \phi(x_0)$, the sensitivities at time $t = K\Delta t$ can be written as :

$$\underbrace{\frac{\partial x_t}{\partial x_0} = J_f(x_{K-1})\, J_f(x_{K-2}) \cdots J_f(x_0)}_{\text{autoregressive}} \qquad \underbrace{\frac{\partial x_t}{\partial x_0} = J_\psi(z_t)\, e^{At}\, J_\phi(x_0)}_{\text{Koopman}}$$

The Koopman formulation does not involve a product of nonlinear Jacobians: the dependence on the horizon is entirely carried by the matrix exponential, evaluated once, while the encoder and decoder

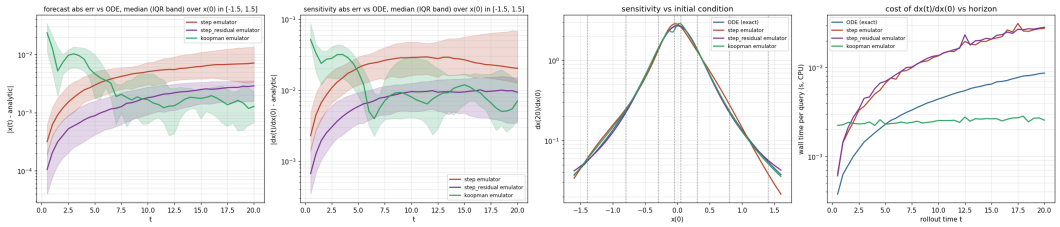


Figure 2: Autoregressive, residual autoregressive and Koopman emulators on the double-well system. From left to right: forecast error and sensitivity error over time; sensitivity as a function of the initial condition; cost of a sensitivity query as a function of the horizon.

are only differentiated at the two ends of the trajectory. This has two consequences. First, *the cost of a sensitivity query does not grow with the number of forecast steps*, where backpropagation through an autoregressive rollout has a time and memory cost linear in K. Second, long products of Jacobians are prone to exploding or vanishing gradients (Bengio et al., 1994; Pascanu et al., 2013), and small errors in the one-step derivative compound multiplicatively along the rollout. The Koopman model is not immune to unstable gradients, but they are entirely controlled by the spectrum of A: constraining the real parts of its eigenvalues is much easier than controlling the product of K state-dependent Jacobians.

3.4 Illustrative experiment

We illustrate these differences on the double-well system $\dot{x} = r\,(x - x^3)$, whose flow and sensitivity are known in closed form. We train three emulators with the same rollout loss, without any supervision on derivatives: an autoregressive MLP $x_{k+1} = f_\theta(x_k)$, its residual variant $x_{k+1} = x_k + \Delta t\, f_\theta(x_k)$, and a Koopman model with a continuous latent space (Section 1).

As shown in Figure 2, the three models reach a similar forecasting accuracy and recover the analytic sensitivity well. On this simple dissipative system, the approaches thus give similar gradients, but at a very different cost: the cost of a sensitivity query grows linearly with the horizon for autoregressive models, while it stays constant for the Koopman model. On longer rollouts or chaotic systems, however, the product of one-step Jacobians is expected to suffer from vanishing or exploding gradients (Bengio et al., 1994; Pascanu et al., 2013; Mikhaeil et al., 2022), whereas for the Koopman model this issue reduces to controlling the spectrum of A. In paragraph 5, we show both this gap in computational cost and a gap in assimilation performance.

4 Method

Training. The training procedure follows a recurrent rollout scheme. Given an input sequence context $\{x_{t_{i-1}}, x_{t_i}\}$, the model predicts a future trajectory of length N, denoted as $\{\hat{x}_{t_{i+1}}, \hat{x}_{t_{i+2}}, \ldots, \hat{x}_{t_{i+N}}\}$. A detailed architecture overview is provided Figure 11. The training objective is defined as a weighted combination of several loss components that enforce accurate reconstruction, long-horizon prediction, latent-space consistency, and structural regularization. All losses are computed as mean squared errors (MSE) unless stated otherwise.

Overall objective. The final training objective is given by

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \alpha\,\mathcal{L}_{\text{pred}} + \beta\,\mathcal{L}_{\text{latent}} + \lambda_{\text{phys}}\,\mathcal{L}_{\text{phys}}, \tag{2}$$

where α and β control the relative contributions of the rollout prediction and latent consistency losses, respectively, and λ_{phys} weights the optional physics-conditioned regularization terms. More details about how the hyperparameters were chosen can be found in Appendices E.1 to E.3. An ablation study removing different components of the loss can be found in Table 11

Inference and Integration via Matrix Exponentiation. A major advantage of restricting the latent dynamics to a linear Koopman form is that, at inference time, we can leverage the analytical solution of the learned system through matrix exponentiation. This bypasses iterative ODE solvers entirely and yields efficient long-horizon forecasting. Given the linear system in (9), the exact solution

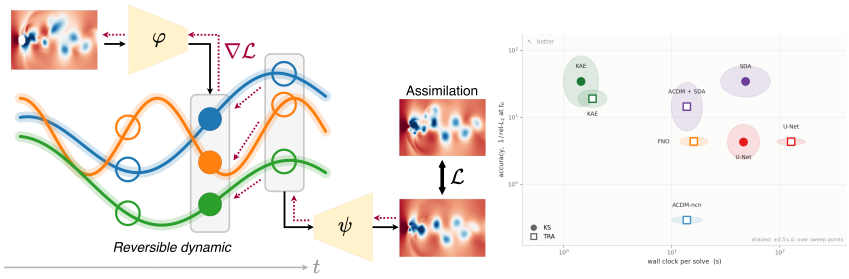


Figure 3: Left : summary figure of the method, φ represent the encoder network from the physical space to continuous latent space and ψ the decoder from latent space to observation. Forward prediction path is depicted as black arrow and gradients for a Cost against accuracy for every method in both regimes. Each marker is an aggregate over the whole set of experiments at $\delta_f = 9$. The spread is variation *across conditions*, and its horizontal extent is set mainly by the δ_l.

Method	Inc_{low}	Inc_{high}	Tra_{ext}	Tra_{int}	Tra_{long}	**Cost**
ResNet-dil (He et al., 2016a)	1.6 ± 1.8	1.5 ± 0.8	1.7 ± 1.0	1.7 ± 1.4	22.0 ± 2.4	3.46
FNO_{16} (Li et al., 2021)	2.8 ± 3.1	8.9 ± 3.8	4.8 ± 1.2	5.5 ± 2.6	20.8 ± 2.0	1.17
TF_{Enc} (Vaswani et al., 2017)	1.5 ± 1.7	$\mathbf{0.6 \pm 0.3}$	3.3 ± 1.2	6.2 ± 4.2	22.2 ± 3.9	0.60
Refiner (Lippe et al., 2023)	$\mathbf{1.3 \pm 1.4}$	3.5 ± 2.2	5.4 ± 2.1	7.1 ± 2.1	Div.	10.31
ACDM (Kohl et al., 2026)	1.7 ± 2.2	$\mathbf{0.8 \pm 0.4}$	2.3 ± 1.4	$\mathbf{2.7 \pm 2.1}$	22.6 ± 4.0	126.57
KAE (Ours)	1.3 ± 1.7	2.9 ± 1.1	$\mathbf{2.2 \pm 0.9}$	5.2 ± 2.4	$\mathbf{14.9 \pm 1.3}$	**1.15**

Table 1: Forecasting accuracy and inference cost across incompressible and transonic flow regimes. Lower MSE and LSiM indicate better performance. MSE values are scaled by 10^{-4}, 10^{-5}, and 10^{-3} for Inc_{low}, Inc_{high}, and the transonic regimes, respectively. Full results and ablations are given in Table 2 in Appendix C.

for any future state z_τ given z_0 is defined by the matrix exponential $z_\tau = \exp(\mathbf{K}_{\text{cont}}\tau)z_0$. This formulation allows us to predict the state at an arbitrary future time τ in a single computational step, independent of the training step size Δt, providing our model the ability to perform zero-shot temporal super-resolution as well.

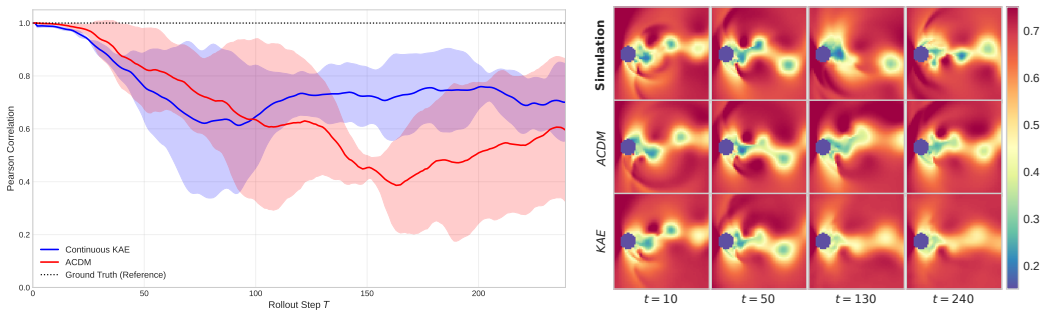
5 Experiments and Results

Is KAE a stable forward surrogate? We first evaluate the models as forward surrogates. We consider two fluid benchmarks from Autoregressive Conditional Diffusion Models (ACDM) Kohl et al. (2026): incompressible wake flow ($Re \in [100, 1000]$) and transonic cylinder flow ($Ma \in [0.50, 0.90]$). Models predict $N = 8$ future states from two conditioning frames. We also evaluate the Kuramoto–Sivashinsky (KS) equation with irregular timesteps. Details of the datasets and preprocessing are given in Appendices B.1 to B.3.

Long-Horizon Forecasting Stability and Cost Table 1 reports the main results. On incompressible flow, KAE outperforms ACDM on both extrapolation regimes, Inc_{low} and Inc_{high}. For transonic flow, ACDM and other baselines generally achieve lower short-term errors on Tra_{int} and Tra_{ext}. At the 240-step horizon, however, KAE gives the best stability among the compared methods. It is also over $110\times$ faster than ACDM at inference and uses fewer training epochs. Refiner and FNO-32 diverge at long horizons.

Long Testing (1000 Steps): In a 1000-step rollout, ACDM develops large phase and spatial errors, while KAE remains bounded (Figure 17). KAE gradually loses fine-scale detail but preserves the main flow structure, as also seen in the shorter 240-step rollout (Figure 4). Further error, spectral, and extrapolation analyses are provided in Appendices F and G.1–G.2.

Effect of rollout training: For the chaotic KS equation, training with a single rollout step ($R = 1$) leads to divergence during 100-step evaluation (Figure 6a). Increasing the training rollout to $R = 2$



(a) Spatial Correlation (240 Steps)

(b) Long Rollout (240 Steps)

Figure 4: Evaluation of long-horizon structural stability in the chaotic Transonic regime. (a) Spatial Pearson correlation over a 240-step rollout, demonstrating that KAE maintains stable structural alignment while the diffusion baseline diverges. (b) Visual snapshots over a 240-step rollout. While ACDM compounds stochastic errors into numerical noise, KAE diffuses smoothly into a stable limit cycle.

substantially reduces the error, with further improvements up to $R = 10$. Thus, multi-step training is important for long-term stability.

Recovery of the Continuous-Time Koopman Operator We test whether the learned $\mathbf{K}_{\text{cont}}$ represents a continuous-time generator. RK4 integration and the closed-form matrix exponential produce closely matching trajectories across the incompressible and transonic datasets (Figures 5a–5b). The eigenvalue spectrum is also mostly in the stable half-plane, $\text{Re}(\lambda) < 0$ (Appendix G.3). The continuous-time formulation and its relationship to discrete Koopman consistency are further discussed in Appendix K, while numerical solver behavior and physical stiffness are analyzed in Appendix N. These results support the use of $\mathbf{K}_{\text{cont}}$ as a continuous-time operator rather than a discrete approximation. The model also supports inference at unseen timesteps. Figure 5c shows rollouts using $\Delta t \in \{0.05, 0.1, 0.2\}$ without retraining.

Generalization to Unseen Dynamics via Physics Conditioning We test whether the physics-conditioned LoRA operator allows the model to generalize to unseen flow regimes. Figure 14 shows MSE across Reynolds numbers from 100 to 1000. KAE gives smoother error changes across the range, while ACDM shows larger variation, especially at high Reynolds numbers and for pressure. On Tra_{ext}, KAE achieves a low, comparable MSE. Removing physics conditioning increases the error by two orders of magnitude. This shows that the physics-conditioned LoRA is important for generalization across flow regimes. Full ablations are provided in Appendix L.

Training on Irregularly Sampled Data Finally, we test whether KAE can train on irregularly sampled data. We randomly remove KS frames with dropout probabilities $p \in [0.1, 0.9]$. The model remains stable across all dropout levels; even with 90% dropout, the 100-step MSE increases only slightly (Figure 6b). This shows that the continuous formulation does not require a fixed timestep during training. Further results and ablations are given in Appendix M.

Does forecasting accuracy predict assimilation accuracy? We next evaluate the same models as state estimators using variational data assimilation. The goal is to recover the unknown state $u(t_0)$ from observations collected after t_0. We vary the first-observation lead time δ_f, while keeping the recovery horizon $\delta_l = 25$ frames and the number of observations $N = 5$ fixed unless stated otherwise. The N observations span $[\delta_f, \delta_l]$, with the endpoints fixed and interior observations geometrically spaced.

For KAE, the observations are encoded and mapped back to t_0 using the learned continuous operator, $z_0 =$

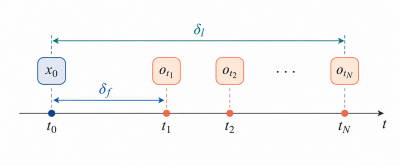
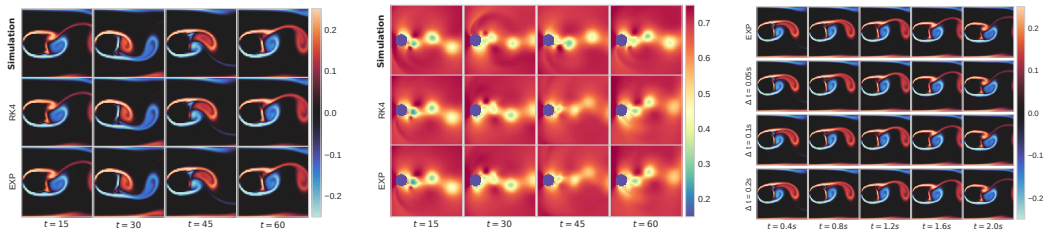
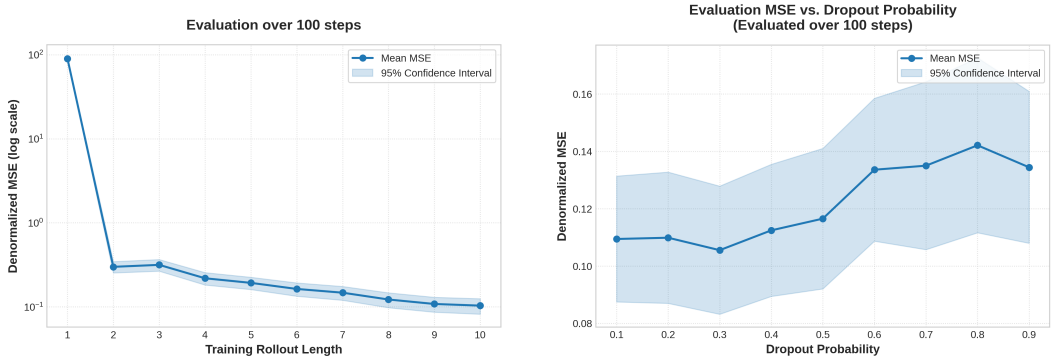


Figure 7: Qualitative representation of the parameters considered in the assimilation process.



(a) Vorticity (RK4 vs Exp) (b) Pressure (RK4 vs Exp) (c) Variable Δt Inference

Figure 5: Validation of continuous-time latent dynamics. (a,b) Close agreement between RK4 integration and the matrix exponential for incompressible and transonic regimes. (c) Rollouts at three unseen integration step sizes ($\Delta t \in \{0.05, 0.1, 0.2\}$ s), demonstrating zero-shot temporal generalization.



(a) Failure at Rollout $R = 1$ (b) Irregular-Timestep Robustness

Figure 6: KS stability analysis. (a) Single-step training ($R = 1$) causes catastrophic divergence during 100-step evaluation. (b) The continuous generator maintains stable performance even when 80% of training frames are randomly discarded.

$e^{-\mathbf{K}_{\text{cont}}\tau_1}\mathcal{E}(y_1)$. For the autoregressive baselines, the nearest observation is used as the initial state. This warm start gives all methods an initial estimate before optimisation.

The transonic experiment provides a direct comparison between the two tasks. When given the true initial state, U-Net gives the lowest 20-step forecasting error, while KAE gives the highest. When the initial state must instead be recovered from observations, the ordering reverses: KAE gives the lowest error in both examples, while U-Net is among the worst. Details of algorithms implementations are in Appendix I.1.

How does assimilation change as observations become less informative? The previous comparison uses a fixed assimilation configuration. We next vary the first-observation lead time δ_f to measure how the models behave as the state being recovered becomes increasingly separated from the available observations.

Kuramoto–Sivashinsky We first test how assimilation changes as the first observation is moved further from the unknown initial state. We sweep δ_f from 1 to 20 frames while keeping $\delta_l = 25$ and $N = 5$ fixed (Figure 9a). Across the sweep, KAE improves over SDA by a more than twice. SDA eventually becomes worse than simply copying the nearest observation at large δ_f. Figure 10 shows four recovered KS states. The models receive only observations after t_0; the true state is never provided.

Experimental setup. The KS equation is $u_t = -uu_x - u_{xx} - u_{xxxx}$ on a periodic domain with 64 spatial points and $\Delta t = 0.1$. Training uses 100 trajectories of 1000 frames. Assimilation uses 64 separate trajectories, with an additional 32 trajectories of 5000 frames for post-assimilation forecasts.

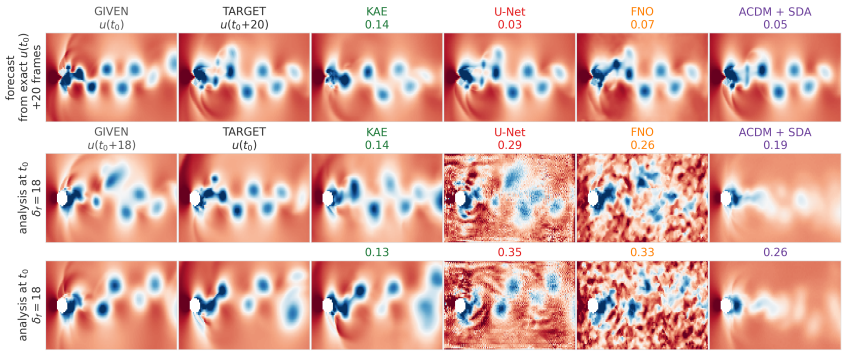
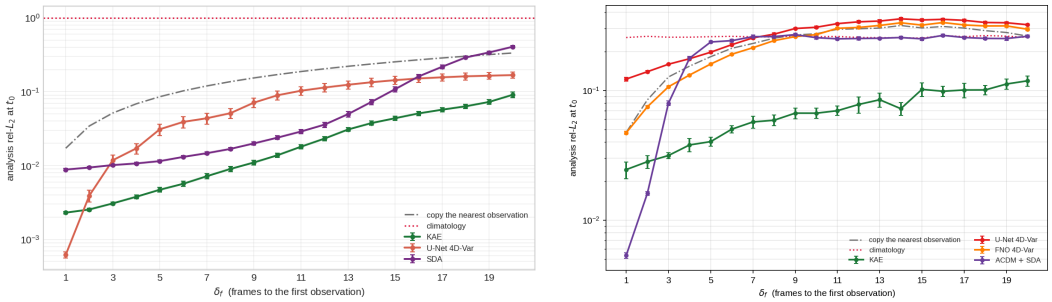


Figure 8: **Transonic fields: the same four models as forecasters and as assimilators.** Pressure is shown and the number above each panel is the relative L_2 over all four channels. **Top block — forecasting.** Each model is handed the exact $u(t_0)$ and rolled forward 20 frames; no assimilation and no optimisation. **Lower blocks — assimilation.** $u(t_0)$ is never given. It is recovered from five observations, so the earliest sits $\delta_f = 18$ frames after the state being recovered.



(a) Kuramoto–Sivashinsky

(b) Transonic cylinder flow

Figure 9: **Assimilation error versus first-observation lead time.** The first observation is moved from $\delta_f = 1$ to 20 frames while $\delta_l = 25$ and $N = 5$ remain fixed. (a) On KS, KAE maintains lower assimilation error than SDA across the lead-time range, while SDA eventually becomes worse than copying the nearest observation. (b) On transonic flow, ACDM+SDA performs well at small δ_f but degrades rapidly as the observation gap increases. KAE remains below the reference baselines across the full range. The dashed line shows the nearest-observation baseline and the dotted line shows climatology.

The Lyapunov time is $T_L = 22.6$ time units. KAE and U-Net are trained on the same data using 10-frame rollouts. The score model is trained on nine-frame segments. For assimilation, KAE and U-Net use 8000 Adam iterations, while SDA uses its fixed sampling procedure. Detailed KS assimilation settings are given in Appendix I.2, including the convergence analysis in Figure 23; the corresponding KS ablations and parameter sweeps are reported in Appendix J and Figure 25.

Transonic Cylinder Flow We repeat the lead-time experiment on transonic flow (Figure 9b). Here, δ_f again varies from 1 to 20 frames, while $\delta_l = 25$ and $N = 5$ remain fixed. The evaluation uses unseen intermediate Mach numbers $Ma \in \{0.66, 0.67, 0.68\}$.

Experimental setup. The transonic dataset contains compressible flow past a cylinder at $Re = 10^4$, with resolution 128×64 and four physical fields. The data are sampled at $\Delta t = 0.1$ s. Training covers $Ma \in [0.53, 0.63] \cup [0.69, 0.89]$. Assimilation uses the unseen intermediate range $Ma \in \{0.66, 0.67, 0.68\}$.Full transonic model and optimization details are provided in Appendix I.3, while the corresponding TRA sweeps and assimilation ablations are summarized in Figure 24 and Appendix J.

ACDM's error increases rapidly with lead time and is rapidly worse than KAE, and quickly reaching the climatology error. U-Net and FNO are also unable to improve consistently over simply copying

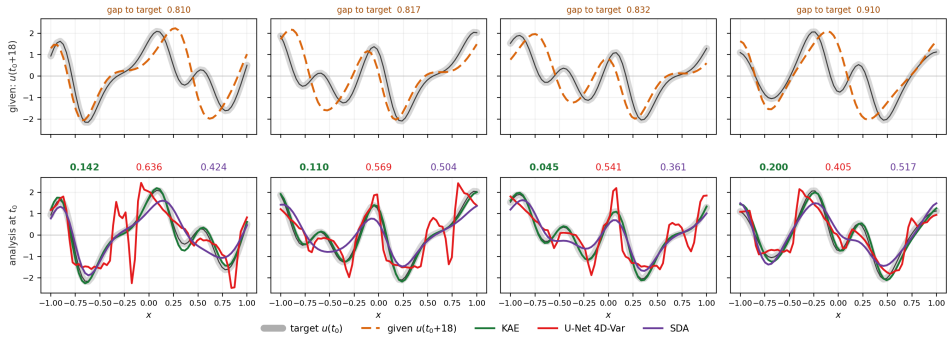


Figure 10: **KS states recovered at t_0, four trajectories.** *Top row:* the same target with the field each method was actually *given* ($\delta_f = 18$), which is also where every solve starts. *Bottom row:* the target $u(t_0)$, with all three analyses plotted on it; the three numbers above each panel are that panel's relative L_2, coloured to match the curve they belong to. We see a similar pattern in (Lippe et al., 2023).

the nearest observation. KAE is the only method that remains below both reference baselines across the full lead-time range.

6 Discussion and Limitations

The experiments show that forecasting and data assimilation test different properties of learned dynamical surrogates. KAE provides stable long-horizon forecasts, but forecasting accuracy does not consistently predict assimilation accuracy. In the transonic experiments, U-Net achieves lower short-horizon forecasting error, while KAE achieves lower assimilation error, with the same separation observed in the controlled ablation. Thus, forecast accuracy alone does not characterize a surrogate's usefulness for state estimation.

The continuous-time formulation propagates latent states directly to arbitrary observation times without intermediate autoregressive steps, making it suitable for irregularly spaced observations and long assimilation windows. Experiments with irregular sampling and unseen integration timesteps support this property.

There are several limitations. The assimilation experiments use a controlled variational state-recovery setting and only two fluid systems and the Kuramoto–Sivashinsky equation. Assimilation still requires iterative optimisation, so the computational benefit depends on the observation configuration and optimisation procedure. Finally, although KAE remains stable over long horizons, it loses fine-scale detail at 1000 steps. Further evaluation on more complex systems and observation settings is needed.

7 Conclusion

We introduced a continuous-time Koopman autoencoder for long-term forecasting and data assimilation in complex physical systems. Its continuous latent dynamics enable direct propagation to arbitrary timesteps and observation times, without intermediate autoregressive steps.

Across incompressible flow, transonic flow, and the Kuramoto–Sivashinsky equation, KAE provides stable long-horizon forecasts at low inference cost. More importantly, the experiments show that forecasting accuracy does not determine assimilation accuracy: models that perform well as forward predictors can perform poorly at initial-state recovery, while KAE can show the opposite behaviour. The controlled loss ablation confirms this separation within the same architecture.

These results highlight that forecasting and data assimilation should be evaluated as distinct properties of learned physical surrogates. For state recovery from observations, forward forecasting accuracy alone is not sufficient to characterize the usefulness of the learned dynamics.

AI USE DISCLOSURE

We used generative AI tools during the preparation of this work to assist with drafting sections of the manuscript, organizing and editing the presentation of the research, and developing and automating parts of the experimental workflow, including experiment execution and plot generation. All AI-assisted text was reviewed and modified by the authors before inclusion in the manuscript.

Generative AI tools were also used to reproduce existing implementation code and generate the Kuramoto–Sivashinsky datasets used in this work. The underlying dataset-generation procedure was based on the existing code and was reviewed and verified by the authors. AI tools were not used to independently conduct the research: experimental procedures, generated data, experimental outputs, plots, numerical results, and reported claims were checked by the authors.

All AI-assisted code and outputs were reviewed by the authors and verified against the intended experimental procedures. The authors take full responsibility for the final content of this work, including all text, code, datasets, experimental results, analyses, figures, and claims.

REFERENCES

Melissa Adrian, Daniel Sanz-Alonso, and Rebecca Willett. Data assimilation with machine learning surrogate models: A case study with fourcastnet. *Artificial Intelligence for the Earth Systems*, 4(3): e240050, 2025.

Omri Azencot, N. Benjamin Erichson, Vanessa Lin, and Michael W. Mahoney. Forecasting sequential data using consistent koopman autoencoders. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, 2020.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

F Bouttier and P Courtier. *Data assimilation concepts and methods*. ECMWF, 2002.

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

Roberto Buizza and Tim N. Palmer. The singular-vector structure of the atmospheric global circulation. *Journal of the Atmospheric Sciences*, 52(9):1434–1456, 1995.

Jake Buzhardt, C Ricardo Constante-Amores, and Michael D Graham. On the relationship between koopman operator approximations and neural ordinary differential equations for data-driven time-evolution predictions. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 35(4), 2025.

Ashesh Chattopadhyay, Mustafa Mustafa, Pedram Hassanzadeh, Eviatar Bach, and Karthik Kashinath. Towards physics-inspired data-driven weather forecasting: integrating data assimilation with a deep spatial-transformer-based u-net in a case study with era5. *Geoscientific Model Development*, 15(5):2221–2237, 2022.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023.

PHILIPPE Courtier, J-N Thépaut, and Anthony Hollingsworth. A strategy for operational implementation of 4d-var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120(519):1367–1387, 1994.

Ronan Fablet, Bertrand Chapron, Lucas Drumetz, Etienne Mémin, Olivier Pannekoucke, and François Rousseau. Learning variational data assimilation models and solvers. *Journal of Advances in Modeling Earth Systems*, 13(10):e2021MS002572, 2021.

Anthony Frion, Lucas Drumetz, Mauro Dalla Mura, Guillaume Tochon, and Abdeldjalil Aïssa-El-Bey. Leveraging neural Koopman operators to learn continuous representations of dynamical systems from scarce data. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

Anthony Frion, Lucas Drumetz, Mauro Dalla Mura, Guillaume Tochon, and Abdeldjalil Aissa El Bey. Neural koopman prior for data assimilation. *IEEE Transactions on Signal Processing*, 72: 4191–4206, 2024.

Masoud Ghasemian, Z. Najafian Ashrafi, and Ahmad Sedaghat. A review on computational fluid dynamic simulation techniques for darrieus vertical axis wind turbines. *Energy Conversion and Management*, 149:87–100, 2017.

Rakesh Halder, Benet Eiximeno, and Oriol Lehmkuhl. Reduced-order modeling of large-scale turbulence using koopman β-variational autoencoders. *Physics of Fluids*, 38(1), 2026.

Sam Hatfield, Matthew Chantry, Peter Dueben, Philippe Lopez, Alan Geer, and Tim Palmer. Building tangent-linear and adjoint models for data assimilation with neural networks. *Journal of Advances in Modeling Earth Systems*, 13(9):e2021MS002521, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016a. doi: 10.1109/CVPR.2016.90.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016b.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Georg Kohl, Li-Wei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. *Neural Networks*, page 108641, 2026.

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

Rolf H. Langland and Nancy L. Baker. Estimation of observation impact using the NRL atmospheric variational data assimilation adjoint system. *Tellus A*, 56(3):189–201, 2004.

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

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9, 2018.

Emmanuel Menier, Sebastian Kaltenbach, Mouadh Yagoubi, Marc Schoenauer, and Petros Koumoutsakos. Interpretable learning of effective dynamics for multiscale systems. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 481(2305), 2025.

Jonas M. Mikhaeil, Zahra Monfared, and Daniel Durstewitz. On the difficulty of learning chaotic dynamics with RNNs. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. *Advances in neural information processing systems*, 31, 2018.

Ionel M Navon. Data assimilation for numerical weather prediction: a review. *Data assimilation for atmospheric, oceanic and hydrologic applications*, pages 21–65, 2009.

Indranil Nayak, Ananda Chakrabarti, Mrinal Kumar, Fernando L. Teixeira, and Debdipta Goswami. Temporally-consistent koopman autoencoders for forecasting dynamical systems. *Scientific Reports*, 15, 2025.

Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

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

Stephen G Penny, Timothy A Smith, T-C Chen, Jason A Platt, H-Y Lin, Michael Goodliff, and Henry DI Abarbanel. Integrating recurrent neural networks with data assimilation for scalable data-driven state estimation. *Journal of Advances in Modeling Earth Systems*, 14(3), 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

François Rozet and Gilles Louppe. Score-based data assimilation. *Advances in Neural Information Processing Systems*, 36:40521–40541, 2023.

Jeffrey Slotnick, Abdollah Khodadoust, Juan Alonso, David Darmofal, William Gropp, Elizabeth Lurie, and Dimitri Mavriplis. Cfd vision 2030 study: A path to revolutionary computational aerosciences. In *High-Performance Computing (HPC) User Forum*, 2014.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems 30*, pages 1130–1140, 2017.

Meng Tang, Yimin Liu, and Louis J Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *Journal of Computational Physics*, 413:109456, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

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

A Detailed explanation of Koopman Autoencoders

Consider a discrete-time dynamical system with states $x_t \in \mathbb{R}^{N_d}$, where x_t denotes the state of the system at time $t \in \mathbb{R}$. The state evolves to its next timestep $t + \Delta t$ according to a flow map $f : \mathbb{R}^{N_d} \to \mathbb{R}^{N_d}$:

$$x_{t+\Delta t} = f(x_t). \tag{3}$$

Koopman operator theory shifts the focus from the state variables x_t to a set of measurement functions $h : \mathbb{R}^{N_d} \to \mathbb{R}$, which are elements of an infinite-dimensional Hilbert space. The Koopman operator $\mathcal{K}$ acts on measurement functions h linearly with the dynamics:

$$(\mathcal{K}h)(x_t) = h(f(x_t)) = h(x_{t+\Delta t}). \tag{4}$$

In this formulation, the underlying nonlinear system (3) is transformed into a linear evolution in the infinite-dimensional space of observables. The challenge lies in choosing or constructing a suitable set of observables. KAEs address this by learning the observables from data, training an encoder which maps the state into a latent representation $\mathcal{E} : \mathbb{R}^{N_d} \to \mathbb{R}^{N_z}$,

$$z_t = \mathcal{E}(x_t), \tag{5}$$

which serves as a finite-dimensional approximation of the observable space. In this latent space, the evolution can be modelled as an approximately linear finite-dimensional Koopman operator $\mathbf{A}$,

$$z_{t+\Delta t} \approx \mathbf{A} z_t. \tag{6}$$

A decoder $\mathcal{D}$ then reconstructs the state from latent variables,

$$\hat{x}_{t+\Delta t} = \mathcal{D}(z_{t+\Delta t}). \tag{7}$$

In our continuous-time formulation, we replace the discrete operator with the linear ODE $dz/dt = \mathbf{K}_{\text{cont}} z$, which allows inference at any horizon τ in a single matrix-vector product, regardless of Δt used during training.

A.1 Proposed Model

The proposed KAE consists of three main components: a dual-stream Transformer encoder, a parametric latent Koopman operator, and a CNN-based decoder (Appendix D.1-D.3), including the temporal history encoder in Appendix D.1.1. Although the continuous formulation naturally supports irregularly sampled data (Paragraph 5), our experiments use uniformly spaced timesteps to enable direct comparison with existing baselines.

We employ a Transformer-based encoding architecture consisting of two symmetric streams: a history encoder ($\mathcal{E}_{\text{history}}$) and a present encoder ($\mathcal{E}_{\text{present}}$). To maintain consistency with benchmark evaluations, we formulate the temporal prediction as an Auto-Regressive process of order 2 (AR-2). Mathematically, we approximate a deterministic expected value mapping of the future state, rather than generating a full conditional probability distribution.

Fluid flows are inherently non-Markovian in observable space due to hidden state variables (e.g., pressure fields missing from velocity observations). Therefore, relying on a single spatial snapshot is dynamically insufficient. To construct a richer, fully Markovian initial state z_{t_i} in the latent space, we aggregate temporal context. Specifically, the immediate past ($x_{t_{i-1}}$) and present (x_{t_i}) states are independently processed by the convolutional encoder. To break temporal symmetry and correctly capture flow directionality, these representations are then processed by a Transformer encoder (Vaswani et al., 2017) with sinusoidal positional encodings before being averaged across time:

$$z_{t_i} = \text{MeanPooling}\Big(\text{Transformer}\big(\mathcal{E}_{\text{history}}(x_{t_{i-1}}), \mathcal{E}_{\text{present}}(x_{t_i})\big)\Big). \tag{8}$$

A decoder $\mathcal{D}$ reconstructs the physical state from the latent vector, $\hat{x}_{t_i} = \mathcal{D}(z_{t_i})$.

A.1.1 Koopman Operator

Next, we discuss the structure of the continuous Koopman operator. While most KAE formulations operate in discrete time, directly mapping $z_t \to z_{t+\Delta t}$ as discussed in Section 3, such updates are tied to the temporal resolution of training data. This dependence can hinder generalization across

timesteps and reduce robustness in scenarios with different or irregular temporal resolutions. Our proposed continuous-time model interprets the latent evolution as an ODE,

$$\frac{dz}{dt} = \mathbf{K}_{\text{cont}}(\phi)z, \tag{9}$$

where $\mathbf{K}_{\text{cont}}$ governs the linear evolution in latent space. The state at a future time, $t_i + \Delta t$, is obtained by integrating this ODE:

$$z_{t_i+\Delta t} = z_{t_{i+1}} = z_{t_i} + \int_0^{\Delta t} \frac{\mathrm{d}z}{\mathrm{d}\tau} d\tau, \tag{10}$$

where any solver could used to conduct this integration stably during training, in practice we use fourth-order Runge–Kutta. During inference, exact matrix exponentiation could be used for maximum efficiency, which will be discussed in Section 4.

Finally, we define the structure of the operator. To capture the dependency on physical parameters ϕ, we express $\mathbf{K}_{\text{cont}}(\phi)$ as the sum of a static base dynamics matrix and a parametric adjustment:

$$\mathbf{K}_{\text{cont}}(\phi) = \mathbf{K}_0 + \mathcal{N}_\psi(\phi). \tag{11}$$

While conditioning neural surrogate models on physical parameters is common in PDE forecasting, incorporating this conditioning directly into a Koopman operator has received comparatively little attention. Therefore, an innovation in our approach is explicitly conditioning the continuous-time Koopman operator on these external control parameters, denoted by ϕ. In our setting, ϕ represents physical parameters such as Reynolds or Mach numbers. Unlike traditional Koopman approaches where a single static operator is learned for the entire domain, our operator adapts dynamically based on the specific physical regime. Here, the base dynamics are governed by a learnable matrix $\mathbf{K}_0$, which captures the global, invariant behavior shared across all flow regimes. $\mathcal{N}_\psi$ is a Low-Rank Adaptation (LoRA (Hu et al., 2022)) module parameterized by ψ. We utilize a LoRA parameterization rather than a full-rank conditional matrix due to strict computational constraints. For a latent dimension N_z, a full-rank regime-conditioned operator requires $O(N_z^2)$ parameters, leading to severe VRAM fragmentation during batched training. LoRA reduces this to $O(2rN_z)$, acting as a structural regularizer that prevents $\mathbf{K}_{\text{cont}}$ from deviating too radically. This parameterization implicitly constrains regime-specific dynamics to remain close to a shared global flow manifold, improving stability when extrapolating across unseen physical regimes. During training, we inject small Gaussian noise into the conditioning parameters ϕ. This acts as a regularizer that improves robustness to sparse parameter sampling and discretization artifacts. In practice, this encourages the learned Koopman operator to vary smoothly across parameter space, improving interpolation between regimes.

B Data Details

To rigorously evaluate the proposed continuous-time Koopman framework, we introduce a suite of dynamical system datasets ranging from canonical fluid flow benchmarks to chaotic spatiotemporal PDEs. All models were trained on the public data from (Kohl et al., 2026) or on synthetical, locally generated datasets simulating the Kuramoto–Sivashinsky equation. For a thorough comparison, we used the exact same split in terms of train/evaluation. The incompressible model was trained on all the sequences with Reynolds numbers in the $[200, 900]$ interval with a step of 10, and evaluated on the strictly disjoint $Inc_{low} = [100, 180]$, $Inc_{high} = [920, 1000]$ extreme intervals.

The transonic model was also trained on the sequences with Mach numbers in the $[0.53, 0.54, ..., 0.62, 0.63] \cup [0.69, 0.70, ..., 0.89, 0.90]$ and evaluated on the sequences with $Tra_{ext} = [0.50, 0.51, 0.52]$, $Tra_{int} = [0.66, 0.67, 0.68]$ for 60 time steps and $Tra_{long} = [0.64, 0.65]$ for 240 time steps. The main difference in training and loading the data is that we treat them in an exhaustive manner. We follow the same stride (a sub-sampling factor of 2), but we load the data such that the previous sequence's prediction can be initial conditions for the following sequence.

Moreover, we used a different structure. The original Diffusion Model was using a simple tensor with multiple channels, each of them representing a different variable (v_x, v_y, etc.). For expressiveness we used TensorDicts, which are PyTorch's (Paszke et al., 2019) version of dictionaries, where the values are tensors. Each of the channels from the original formulation was translated into a specific variable.

B.1 INCOMPRESSIBLE WAKE FLOW

This dataset consists of a fully developed incompressible Karman vortex street behind a cylindrical obstacle with varying Reynolds numbers $Re \in [100, 1000]$, comprising 91 simulations in total (Kohl et al., 2026). For training, both the baseline models and our approach utilize simulations within the range $Re \in [200, 900]$. There are two distinct test sets used to measure out-of-distribution extrapolation: Inc_{low}, covering $Re \in [100, 180]$, and Inc_{high}, covering $Re \in [920, 1000]$. Both test sets use a prediction horizon of $T = 60$ steps. While the raw datasets employ a time step of $\Delta t = 0.05$s, we follow the baseline protocol and subsample the trajectories by a factor of 2, resulting in an effective time step of $\Delta t = 0.1$s for model training and evaluation.

B.2 TRANSONIC CYLINDER FLOW

Transonic flows exhibit highly chaotic behavior, characterized by the formation of shock waves that dynamically interact with the fluid flow (Kohl et al., 2026). The dataset consists of 40 sequences of fully developed compressible Karman vortex streets at a fixed Reynolds number $Re = 10,000$, with Mach numbers varying across the range $Ma \in [0.50, 0.90]$. Models are trained on sequences with $Ma \in [0.53, 0.63] \cup [0.69, 0.90]$. For evaluation, two test sets, Tra_{ext} and Tra_{int}, represent simulations with $Ma \in [0.50, 0.52]$ and $Ma \in [0.66, 0.68]$ respectively, with standard rollout steps of $T = 60$. To test extreme stability over roughly 8 vortex shedding periods, an additional long rollout case, Tra_{long}, is defined with $Ma \in [0.64, 0.65]$ and $T = 240$. Similar to the incompressible case, the raw simulation interval is $\Delta t = 0.05$s, subsampled to $\Delta t = 0.1$s for our experiments.

B.3 1D KURAMOTO-SIVASHINSKY (KS) EQUATION

To rigorously validate our continuous-time formulation's ability to handle irregularly sampled data and chaotic phase turbulence, we evaluate our model on the Kuramoto-Sivashinsky (KS) equation. The dynamics are governed by the following partial differential equation:

$$u_t = -uu_x - u_{xx} - u_{xxxx} + f(x, t)$$

where $f(x, t)$ represents an optional external forcing term. For our primary experiments, we simulate the unforced system ($f = 0$). The spatial domain is defined with periodic boundary conditions over a length $L = 22.0$, a standard regime for evaluating chaotic KS dynamics.

The system is numerically integrated using a pseudospectral method for spatial discretization ($n_x = 64$ points, utilizing the 2/3 rule for dealiasing the nonlinear term) and an explicit Runge-Kutta method (`RK45`, rtol $= 10^{-6}$, atol $= 10^{-8}$) for time stepping. We generate 100 independent trajectories, each integrated for 1000 steps with a sampling interval of $\Delta t = 0.1$. To align with standard spatial resolutions and convolutional frameworks, the final outputs are downsampled to 64 spatial grid points and cast with a dummy spatial dimension. The resulting dataset is split into 80% training, 10% validation, and 10% testing subsets.

C FULL MODELS COMPARISON

D IMPLEMENTATION AND MODEL DETAILS

The entire model is implemented and trained using PyTorch (Paszke et al., 2019). All wights in the architecture were optimized using AdamW (Loshchilov and Hutter, 2017) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate follows a Cosine Warm-up schedule, linear increasing for the first 20 epochs from 0.0 to 5×10^{-4}, then decreasing on a cosine curve to 10^{-5} for the following 180 epochs, making it 200 epochs in total for training, with a batch size of 64.

We also follow an end-to-end training procedure, learning the encoder, operator and decoder simultaneously. While testing different architectures and training schemes, we have experimented with curriculum learning too. As the model can be rather complex, especially for large latent dimensionality, while the data, more specifically the transonic flow, can be very chaotic and rough, we tried training the Auto-Encoder's core (encoder and decoder) for the first 50 epochs alone, freezing the operator's side, as we first wanted to build a smooth manifold for the data. For the following

Method	Inc_{low}		Inc_{high}		Tra_{ext}		Tra_{int}		Tra_{long}		Avg. Step (ms)	Mean VRAM (MB)
	MSE ($\times 10^{-4}$)	LSiM ($\times 10^{-2}$)	MSE ($\times 10^{-5}$)	LSiM ($\times 10^{-2}$)	MSE ($\times 10^{-3}$)	LSiM ($\times 10^{-1}$)	MSE ($\times 10^{-3}$)	LSiM ($\times 10^{-1}$)	MSE ($\times 10^{-3}$)	LSiM ($\times 10^{-1}$)		
ResNet	10 ± 9.1	17 ± 7.8	16 ± 3.0	5.9 ± 1.6	2.3 ± 0.9	1.4 ± 0.2	1.8 ± 1.0	$\mathbf{1.0 \pm 0.3}$	24.2 ± 4.6	7.6 ± 1.7	3.67	188.0
ResNet-dil	1.6 ± 1.8	7.7 ± 5.5	1.5 ± 0.8	2.6 ± 0.7	1.7 ± 1.0	1.2 ± 0.3	1.7 ± 1.4	1.0 ± 0.5	22.0 ± 2.4	5.5 ± 2.3	3.46	**178.6**
FNO_{16}	2.8 ± 3.1	8.8 ± 7.1	8.9 ± 3.8	2.5 ± 1.2	4.8 ± 1.2	3.4 ± 1.1	5.5 ± 2.6	2.6 ± 1.1	20.8 ± 2.0	11.5 ± 1.1	1.17	184.1
FNO_{32}	160 ± 50	80 ± 5.4	$1k \pm 140$	57 ± 4.9	4.9 ± 1.9	3.6 ± 0.9	6.8 ± 3.4	3.1 ± 1.1	Diverged	Diverged	1.17	183.9
TF_{Enc}	1.5 ± 1.7	6.3 ± 4.2	0.6 ± 0.3	1.0 ± 0.3	3.3 ± 1.2	1.8 ± 0.3	6.2 ± 4.2	2.2 ± 0.7	22.2 ± 3.9	3.8 ± 0.4	0.60	3448.6
TF_{MGN}	5.7 ± 4.3	13 ± 6.4	10 ± 2.9	3.5 ± 0.4	3.9 ± 1.0	1.8 ± 0.3	6.3 ± 4.4	2.2 ± 0.7	18.9 ± 4.5	4.0 ± 0.2	0.69	3498.0
TF_{VAE}	5.4 ± 5.5	13 ± 7.2	14 ± 19	4.1 ± 1.4	4.1 ± 0.9	2.4 ± 0.2	7.2 ± 3.0	2.7 ± 0.6	20.6 ± 2.1	4.0 ± 0.2	**0.30**	13749.9
U-Net	1.0 ± 1.1	5.8 ± 3.2	2.7 ± 0.6	2.6 ± 0.6	3.1 ± 2.1	3.9 ± 2.8	2.3 ± 2.0	3.3 ± 2.8	30.3 ± 6.1	9.1 ± 1.2	6.19	183.7
$U\text{-}Net_{ut}$	$\mathbf{0.8 \pm 1.1}$	$\mathbf{4.5 \pm 4.0}$	$\mathbf{0.2 \pm 0.1}$	$\mathbf{0.5 \pm 0.2}$	1.6 ± 0.7	$\mathbf{1.1 \pm 0.2}$	$\mathbf{1.5 \pm 1.5}$	1.0 ± 0.5	22.2 ± 3.6	3.8 ± 0.4	6.16	184.1
$U\text{-}Net_{tn}$	1.0 ± 1.0	5.6 ± 3.1	0.9 ± 0.6	1.5 ± 0.6	$\mathbf{1.4 \pm 0.8}$	1.1 ± 0.3	1.8 ± 1.1	1.0 ± 0.4	22.4 ± 3.9	3.9 ± 0.4	6.16	184.1
Refiner	1.3 ± 1.4	7.1 ± 4.2	3.5 ± 2.2	2.5 ± 1.0	5.4 ± 2.1	2.3 ± 0.5	7.1 ± 2.1	3.0 ± 1.7	Diverged	8.8 ± 3.3	10.31	642.4
$ACDM_{ncn}$	0.9 ± 0.8	6.6 ± 2.7	5.6 ± 2.7	3.6 ± 1.2	4.1 ± 1.9	1.9 ± 0.6	2.8 ± 1.3	1.7 ± 0.4	22.8 ± 3.8	8.2 ± 3.0	126.3	649.2
ACDM	1.7 ± 2.2	6.9 ± 5.6	0.8 ± 0.4	1.0 ± 0.3	2.3 ± 1.4	1.3 ± 0.3	2.7 ± 2.1	1.3 ± 0.6	22.6 ± 4.0	$\mathbf{3.8 \pm 0.4}$	126.57	659.2
KAE	1.3 ± 1.7	6.1 ± 4.8	2.9 ± 1.1	1.7 ± 0.3	2.2 ± 0.9	1.8 ± 0.3	5.2 ± 2.4	2.1 ± 0.6	$\mathbf{14.9 \pm 1.3}$	4.1 ± 0.3	1.15	2751.3
Exponential KAE	1.3 ± 1.7	6.1 ± 4.8	2.9 ± 1.1	1.7 ± 0.3	2.2 ± 0.9	1.8 ± 0.3	5.2 ± 2.4	2.1 ± 0.6	$\mathbf{14.9 \pm 1.3}$	4.1 ± 0.3	1.15	2751.3

Table 2: Quantitative comparison across incompressible and transonic flow regimes alongside computational efficiency. Metrics reported include MSE and LSiM, as well as average per-step inference time and VRAM footprint. This view highlights KAE's state-of-the-art long-horizon stability (Tra_{long}, evaluated over an extended 240-step horizon) combined with its massive inference speedup over diffusion baselines.

100 epochs, we have frozen the encoder and decoder and only trained the operator. Finally, for the remaining epochs, we fine-tuned them together. However, this proved challenging as regularizing the latent space was not as straightforward as expected, leaving the linear operator to struggle.

Our model follows a modular $encoder$–$latent$–$dynamic$–$decoder$ design, combining convolutional feature extraction with physics-conditioned Koopman operators for latent-space temporal evolution. The architecture is specifically designed for stability, interpretability, and robustness in strongly conditioned fluid dynamics regimes.

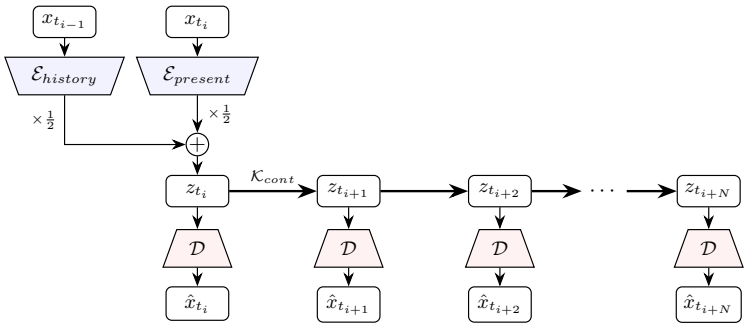


Figure 11: Architecture overview. The history encoder and present encoder (top) produce the initial latent state z_{t_i}, which is propagated by the continuous Koopman operator $\mathcal{K}_{\text{cont}}$ and decoded at each rollout step.

D.1 Encoder Architecture

The spatial encoder is a convolutional residual encoder operating on input fields augmented with explicit coordinate information. For each input snapshot of shape $C \times H \times W$, two additional coordinate channels corresponding to normalized spatial coordinates are concatenated, resulting in $C + 2$ input channels. This coordinate injection enables the model to reason about absolute spatial position without relying on implicit convolutional biases.

The encoder backbone begins with an initial 3×3 convolution, followed by a sequence of pre-activation residual blocks (He et al., 2016b) with Group Normalization (Wu and He, 2018) and SiLU activations. Each resolution level consists of one downsampling residual block with stride 2, and one refinement residual block with stride 1.

We use three resolution levels with channel widths $[64, 128, 256]$, leading to an effective downsampling factor of $2^3 = 8$. All convolutional and linear layers in the encoder employ spectral normalization, which we found crucial for stabilizing gradients in downstream Koopman dynamics learning.

At the lowest spatial resolution, we apply a Convolutional Block Attention Module (CBAM) (Woo et al., 2018), consisting of sequential channel-wise and spatial attention. This allows the encoder to focus on dynamically active regions such as vortices while suppressing background noise.

The resulting feature map is flattened and projected into a latent representation via a linear layer. This latent representation significantly improves numerical stability of the Koopman operator. When late fusion conditioning is enabled, the flattened spatial features are concatenated with a physics embedding before the final projection.

Conditioning and Physics Parameter Embeddings Physical parameters (e.g. Reynolds number, Mach number, forcing terms) are incorporated through radial basis function (RBF)–style expansions, mapping low-dimensional scalars into higher-dimensional embeddings. The expansion type is configurable and shared across encoder, decoder, and dynamics modules. During training, small Gaussian noise can be injected into the conditioning variables to improve robustness to discretization artifacts and sparse parameter sampling.

D.1.1 Temporal History Encoder

For scenarios involving multiple input timesteps, we introduce a history encoder that aggregates temporal context before latent evolution. Each timestep is independently encoded using the convolutional encoder backbone, producing a sequence of latent vectors. These vectors are then processed by a Transformer encoder (Vaswani et al., 2017) with sinusoidal positional encodings. The Transformer output is averaged across time to obtain a single latent context vector, which serves as the initial condition for the Koopman dynamics. This design allows the model to capture temporal correlations without explicitly unrolling convolutional operations over time.

D.2 Decoder Architecture

The architectural asymmetry between the temporal Transformer encoder and the convolutional decoder is highly intentional. While the Transformer globally aggregates temporal dependencies across the 1D time axis, the decoder must reconstruct the physical 2D flow fields. Retaining CNNs in the decoder enforces the spatial translation invariance and local inductive biases necessary to cleanly map the latent state back to the physical grid without artifacts. The latent vector is first linearly expanded and reshaped into a low-resolution feature map. Upsampling proceeds through a sequence of resolution levels $[256, 128, 64]$, each consisting of a conditioned residual block, and a resize–convolution upsampling block (nearest-neighbor upsampling followed by a 3×3 convolution).

To integrate physical parameters, we employ Adaptive Group Normalization (AdaGN) within the decoder residual blocks. Here, scale and shift parameters of the normalization layers are modulated by the physics embeddings, enabling strong, spatially uniform conditioning while preserving convolutional locality. Dropout is applied inside conditioned residual blocks for regularization, using element-wise dropout to avoid suppressing entire physical channels. The final output is produced via a 3×3 convolution mapping back to the original number of physical variables.

D.3 Koopman Dynamics Module

Latent evolution is governed by a physics-conditioned Koopman operator implemented in continuous-time form. We strictly enforce a linear Koopman parameterization, where the dynamics are governed by a base linear operator with low-rank, condition-dependent updates predicted by a hyper-network (LoRA-style adaptation) (Hu et al., 2022). Avoiding a nonlinear MLP ensures the global linearity of the latent space remains intact, which is mathematically required to utilize the $O(1)$ matrix exponential at inference.

Condition dependence is handled via a hyper-network that predicts operator modifications from the physics embeddings. A dissipative inductive bias is encouraged through dissipative initialization and spectral regularization, though stability is not enforced as a hard constraint—it emerges empirically from rollout training.

For continuous dynamics, we model $\frac{d\mathbf{z}}{dt} = K(\phi)\mathbf{z}$, and integrate using a fourth-order Runge–Kutta (RK4) scheme by default, with optional implicit midpoint or Radau IIA solvers for stiff regimes. All implicit solvers internally upcast to $float32$ to ensure numerical robustness.

E Training Objective and Loss Function Details

To ensure that the learned latent dynamics are not only accurate in terms of Euclidean error but also physically consistent, stable, and topologically faithful to the fluid flows, we employ a composite loss function. The total objective $\mathcal{L}_{\text{total}}$ is a weighted sum of reconstruction accuracy, temporal rollout consistency, latent space regularization, and physics-conditioned constraints:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \alpha\mathcal{L}_{\text{pred}} + \beta\mathcal{L}_{\text{latent}} + \lambda_{\text{phys}}\mathcal{L}_{\text{phys}} \tag{12}$$

E.1 Loss Weighting Strategy

The scalar weights α, β, and λ_{phys} are critical for balancing the trade-offs between spatial reconstruction fidelity, temporal stability, and adherence to theoretical Koopman constraints.

- **Reconstruction Base ($\mathcal{L}_{recon}$):** Kept at a unit weight, this anchors the overall scale of the loss. It is heavily prioritized in the early epochs to ensure the encoder and decoder construct a valid spatial manifold before temporal dynamics are strictly enforced.
- **Rollout Prediction (α):** This weight dictates the model's resistance to autoregressive error accumulation. A sufficiently high α forces the Koopman operator to learn a globally stable trajectory rather than a greedy one-step mapping. However, if α dominates too early in training, the network struggles to converge on the underlying spatial representations.
- **Latent Consistency (β):** This parameter controls the strictness of the theoretical Koopman constraints (detailed below). Balancing β is crucial: if it is too low, the latent space evolves non-linearly, risking divergence during long rollouts; if it is too high, it over-constrains the autoencoder, resulting in overly smoothed spatial reconstructions that fail to capture complex flow features.
- **Physics Regularization (λ_{phys}):** Because spatial gradients and spectral amplitudes operate on different numerical scales than standard pixel-wise MSE, λ_{phys} scales these high-frequency penalties. It acts as a structural fine-tuning mechanism to sharpen edges and correct phase shifts once the base dynamics are established.

Below, we detail the mathematical formulation of the latent and physics-insipred components implemented in our framework.

E.2 Latent Space Consistency

To ensure the latent manifold respects the theoretical properties of the Koopman operator, we enforce structural constraints via $\mathcal{L}_{\text{latent}}$. This includes three specific sub-terms:

1. Forward-Backward Linearity Consistency. Inspired by the discrete-time constraints of Azencot et al. (2020), we enforce a continuous-time generalization of operator invertibility. If $\mathbf{K}(\Delta t)$ propagates the state forward, integrating backward via $\mathbf{K}(-\Delta t)$ must recover the previous state. This ensures the continuous generator does not learn a trivial "shrink-to-zero" solution.

$$\mathcal{L}_{\text{lin}} = \underbrace{\|\mathbf{K}(z_t, \Delta t) - z_{t+1}\|_2^2}_{\text{Forward}} + \underbrace{\|\mathbf{K}(z_{t+1}, -\Delta t) - z_t\|_2^2}_{\text{Backward}}. \tag{13}$$

2. Directional Cosine Similarity. To decouple magnitude errors (decay) from directional errors (dynamics), we enforce alignment between the predicted latent vector update and the true encoder trajectory:

$$\mathcal{L}_{\text{cos}} = 1 - \frac{z_{t+1} \cdot z_{t+1}^{\text{target}}}{\|z_{t+1}\|\|z_{t+1}^{\text{target}}\|} \tag{14}$$

3. Energy Conservation Regularization. Ideally, the "energy" (norm) of the latent state should evolve smoothly. We penalize abrupt changes in the latent norm to encourage smooth trajectories:

$$\mathcal{L}_{\text{energy}} = \left(\|z_{t+1}\|_2 - \|z_t\|_2\right)^2. \tag{15}$$

This constraint acts as a soft regularizer against rapid latent norm explosion beyond the training horizon, rather than enforcing strict energy conservation which would contradict the desired dissipative nature of the linear ODE. Without it, the continuous latent norm remains unconstrained beyond the training horizon, leading to catastrophic divergence when rolling out 30x beyond the training data. The total latent loss is given by $\mathcal{L}_{\text{latent}} = \mathcal{L}_{\text{lin}} + w_{cos}\mathcal{L}_{\text{cos}} + \mathcal{L}_{\text{energy}}$.

E.3 Physics-Inspired Loss

Standard MSE losses often result in "pacing" errors (phase shift) or blurred edges. We mitigate this using a physics-conditioned loss $\mathcal{L}_{\text{phys}}$ comprising Sobolev norms and spectral analysis.

Temporal Sobolev Loss (Velocity Matching). Enforces consistency in the time-derivative (velocity) of the flow:

$$\mathcal{L}_{\text{time}} = \left\| \frac{\partial \hat{x}}{\partial t} - \frac{\partial x}{\partial t} \right\|_2^2 \approx \frac{1}{\Delta t^2} \left\| (\hat{x}_{t+1} - \hat{x}_t) - (x_{t+1} - x_t) \right\|_2^2 . \tag{16}$$

Spatial Sobolev Loss (Structure Matching). Enforces consistency in spatial gradients to preserve sharp edges (e.g., shock waves):

$$\mathcal{L}_{\text{space}} = \|\nabla_x \hat{x} - \nabla_x x\|_2^2 + \|\nabla_y \hat{x} - \nabla_y x\|_2^2. \tag{17}$$

Spectral Consistency Loss. To correct phase errors and ensure the model captures the correct shedding frequencies, we compute the loss in the frequency domain using the Fast Fourier Transform ($\mathcal{F}$). This term penalizes discrepancies in both amplitude (energy spectrum) and phase:

$$\mathcal{L}_{\text{spectral}} = \| |\mathcal{F}(\hat{x})| - |\mathcal{F}(x)| \|_1. \tag{18}$$

F Error Analysis

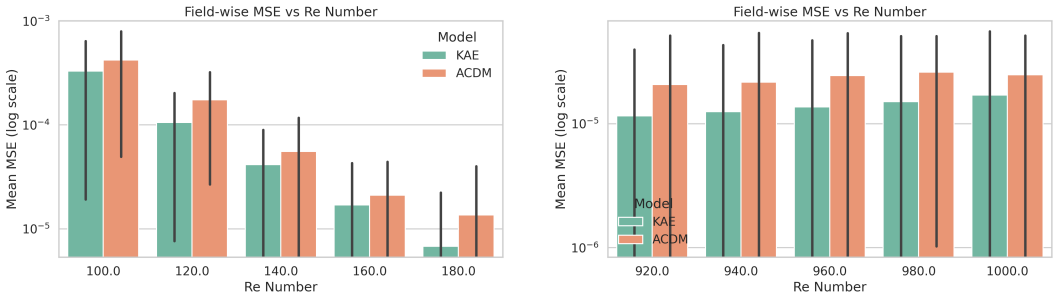
In this appendix, we provide a detailed quantitative and qualitative comparison between our method and ACDM, focusing on accuracy, stability, and robustness across flow regimes. Beyond aggregate error metrics, we emphasize distributional and temporal error characteristics to better understand when and why our approach outperforms diffusion-based forecasting.

We visualize prediction errors using a combination of time-resolved error curves, error bar plots, and violin plots. Error bars report the mean and standard deviation of normalized ℓ_2 errors across test trajectories, highlighting both average performance and variability. Violin plots are used to capture the full error distribution over space and time, revealing differences in tail behavior and robustness that are not visible from mean metrics alone.

Across all evaluated scenarios, our model exhibits consistently lower variance in prediction error, indicating improved stability under long-horizon rollouts. In particular, the error distributions produced by ACDM show heavier tails, corresponding to occasional but severe prediction failures, whereas our Koopman-based model yields tighter, more concentrated distributions. This effect is especially pronounced in transonic regimes, where diffusion-based models tend to suffer from mode collapse or over-smoothing.

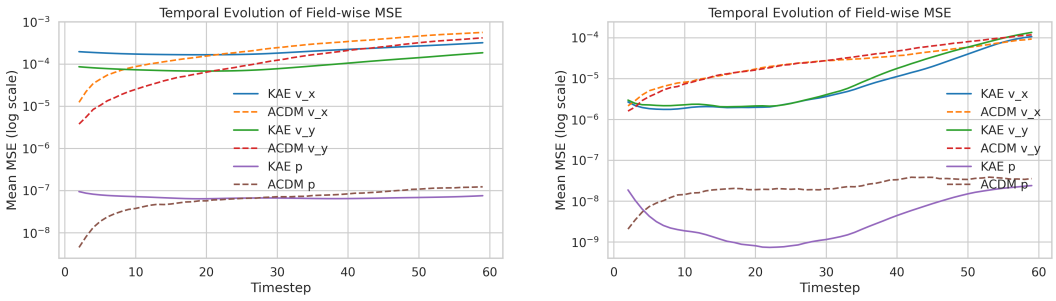
We further include timestep-sensitivity plots, where the same trained model is evaluated under varying rollout step sizes. These visualizations demonstrate that our continuous-time latent dynamics maintain accuracy across a wide range of timesteps, while ACDM performance degrades significantly when evaluated outside its training resolution. This highlights a key advantage of the continuous-time formulation in terms of temporal generalization.

Finally, spatial error maps are presented to illustrate qualitative differences in failure modes. Our model localizes errors primarily in dynamically active regions such as shocks or vortex cores, while maintaining low background error. In contrast, ACDM exhibits spatially diffuse errors that accumulate over time, consistent with the absence of an explicit latent evolution operator.



(a) Low Reynolds regime (b) High Reynolds regime

Figure 12: Field-averaged mean squared error (MSE) comparison between KAE and ACDM for incompressible flows. Error bars denote one standard deviation across test trajectories. Our method consistently achieves lower error and reduced variance, with the performance gap widening in the high-Re regime where chaotic dynamics dominate.



(a) Low Reynolds regime (b) High Reynolds regime

Figure 13: Temporal evolution of field-wise MSE for incompressible flows. Errors are shown on a logarithmic scale. KAE maintains stable error growth over long horizons, while ACDM exhibits accelerated error accumulation, indicative of compounding stochastic prediction drift.

G Extended Stability and Spectral Analysis

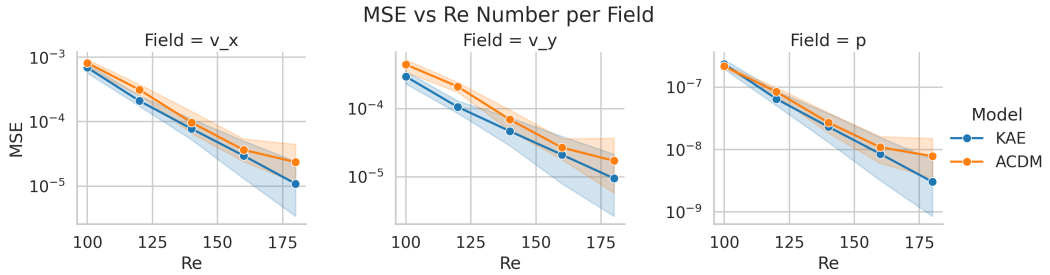
To thoroughly address the accuracy-stability trade-offs between deterministic latent dynamics and stochastic diffusion, this section provides an extended analysis of the models' spectral biases and their behavior under extreme autoregressive rollouts.

G.1 Spectral Bias and Frequency Smoothing

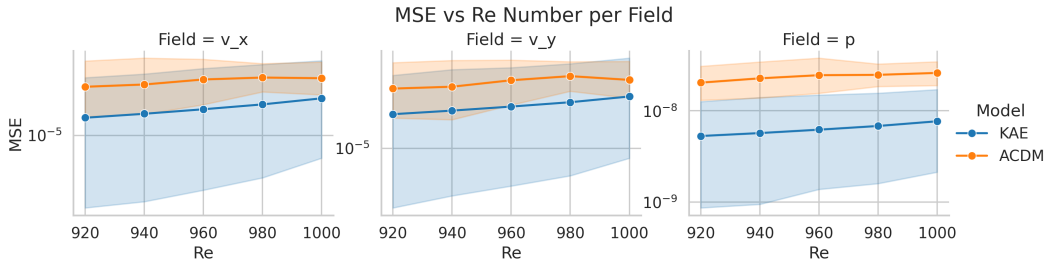
As established in turbulence research (Lorenz, 1969), even highly accurate numerical solvers eventually decorrelate from a target simulation over extended timeframes due to the chaotic nature of fluid dynamics. Therefore, we evaluate the temporal and spatial frequency spectra to measure whether the predicted trajectories statistically match the physical characteristics of the reference simulation. Figure 16 presents the temporal frequency of the vertical motion (v_y) and the spatial wavenumber of the horizontal motion (v_x) evaluated downstream.

The temporal analysis (Figure 16, left) demonstrates that KAE accurately identifies and locks onto the primary vortex shedding frequency (the dominant energy peak), exhibiting a very tight variance band. ACDM also captures this primary frequency, though its stochastic formulation introduces slightly more variance.

In the spatial domain (Figure 16, right), ACDM accurately synthesizes the high-frequency turbulent textures. Conversely, KAE exhibits a steeper energy drop-off at high wavenumbers. This confirms that our model suppresses high-frequency chaotic structures in favor of a stable limit cycle. It acts as a low-pass physical filter, discarding the chaotic, unpredictable high-frequency turbulent cascade to guarantee that the primary, macro-scale shedding frequencies are preserved with near-zero variance.

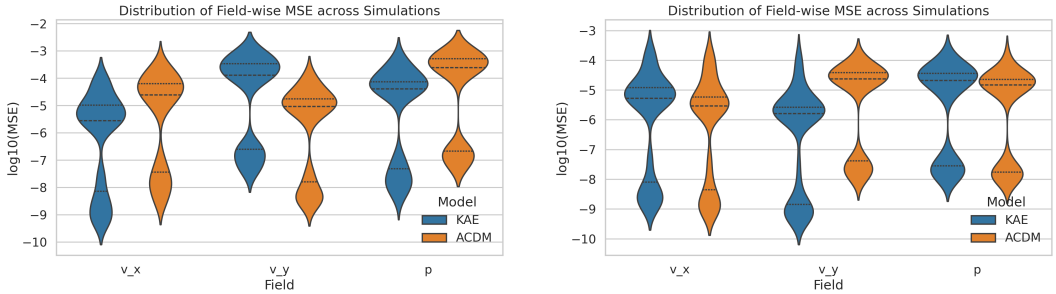


(a) Low Reynolds regime



(b) High Reynolds regime

Figure 14: Field-wise MSE as a function of Reynolds number for incompressible flows. Velocity components and pressure are shown separately. KAE exhibits consistently smoother error scaling with Reynolds number, while ACDM shows increased sensitivity and variance, particularly in pressure-dominated regimes.



(a) Low Reynolds number

(b) High Reynolds number

Figure 15: Error distributions under low and high Reynolds number regimes. While both models perform comparably at low Reynolds numbers, ACDM exhibits pronounced heavy-tailed error distributions at high Reynolds numbers. In contrast, KAE maintains controlled variance, demonstrating superior robustness in turbulent regimes.

G.2 Extreme Long-Horizon Stability ($T = 1000$)

To test the absolute limits of the learned latent dynamics and investigate potential numerical divergence, we subjected both models to an extreme 1000-step autoregressive rollout (as visualized in Figure 4).

As shown in Figure 17, the danger of ACDM's textural hallucinations becomes catastrophic over extreme horizons. The stochastic nature of the diffusion model leads to severe phase divergence: its relative L_2 error spikes erratically with massive variance, and its spatial correlation completely collapses as the high-frequency details compound into unphysical numerical noise.

In contrast, KAE degrades gracefully. Because the latent space is governed by a globally stable linear Koopman operator, the system is asymptotically bounded. The transient, high-frequency modes naturally decay, leaving only the stable base flow. While fine-scale textures diffuse out over time,

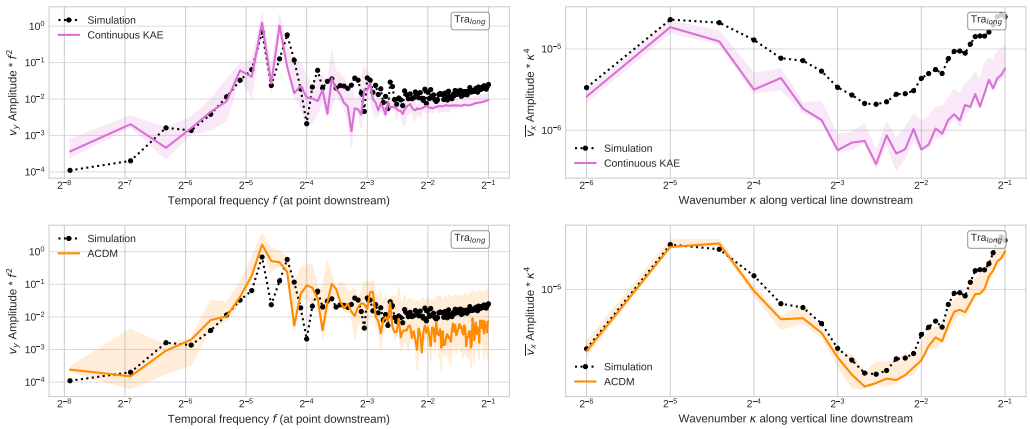
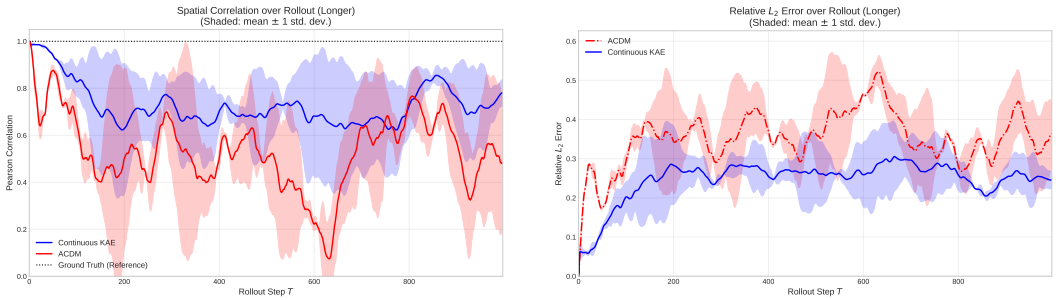


Figure 16: Temporal and spatial frequency analysis on a sequence from Tra_{long}. The shaded area represents the 5th to 95th percentile. KAE successfully captures the dominant physical frequencies but attenuates high-frequency turbulent noise compared to the stochastic baseline.

the model maintains a bounded L_2 error and a highly stable periodic correlation with the reference simulation, proving its reliability for robust long-term forecasting without structural collapse.



(a) Spatial Correlation (1000 steps)

(b) Relative L_2 Error (1000 steps)

Figure 17: Quantitative metrics over an extreme 1000-step rollout. ACDM exhibits severe instability and variance, while KAE remains strictly bounded by its linear latent dynamics.

G.3 Latent Dynamics Eigenvalue Spectrum

To understand the long-horizon stability of the continuous-time Koopman Autoencoder, we analyze the spectral properties of the learned latent dynamics. For a linear continuous-time system $dz/dt = \mathbf{K}z$, stability requires that the real parts of all eigenvalues of $\mathbf{K}$ are negative.

As shown in Figure 18, we plot the eigenvalue distribution of the learned generator $\mathbf{K}_{\text{cont}}(\phi)$ across several conditioning parameters. The majority of eigenvalues lie in the left half of the complex plane ($\text{Re}(\lambda) < 0$), reflecting a dissipative inductive bias that emerges from rollout training rather than being enforced as a hard constraint. This empirically stable spectrum suppresses high-frequency unstable growth modes, so numerical errors introduced during latent projection tend to decay over time rather than compound exponentially, producing bounded predictions over long horizons.

H Spatial Error Distribution and Difference Maps

To analyze the physical nature of the prediction errors, we provide absolute spatial difference maps across all evaluated regimes. While raw MSE metrics often favor stochastic models in chaotic flows, these visual distributions highlight the trade-off between textural detail and structural coherence.

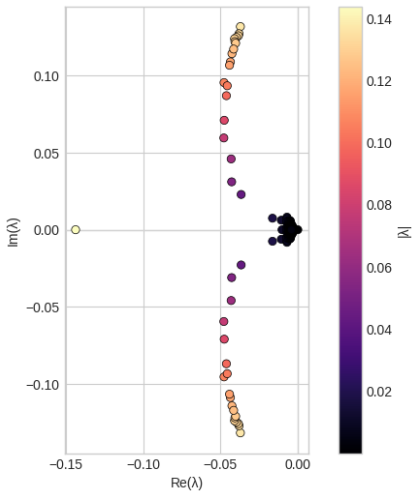
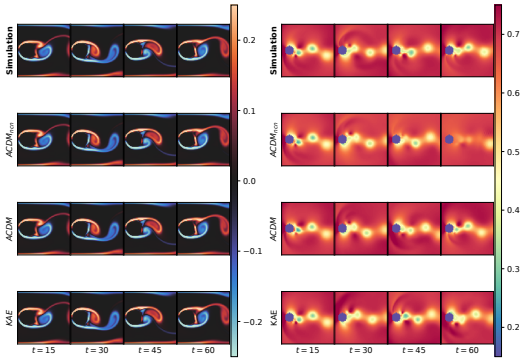


Figure 18: Eigenvalue spectrum of the learned Koopman generator across several conditioning parameters. The majority of eigenvalues lie in the stable region ($Re(\lambda) < 0$), indicating dissipative latent dynamics.



(a) Vorticity for Inc_{high} with $Re = 1000$ (b) Pressure for Tra_{ext} with $Ma = 0.50$

Figure 19: Qualitative validation rollouts for extrapolation regimes. (Left) Vorticity prediction for the incompressible wake flow in the high-Reynolds-number regime ($Inc_{high}, Re = 1000$). (Right) Pressure prediction for the transonic cylinder flow in the low-Mach extrapolation regime ($Tra_{ext}, Ma = 0.50$).

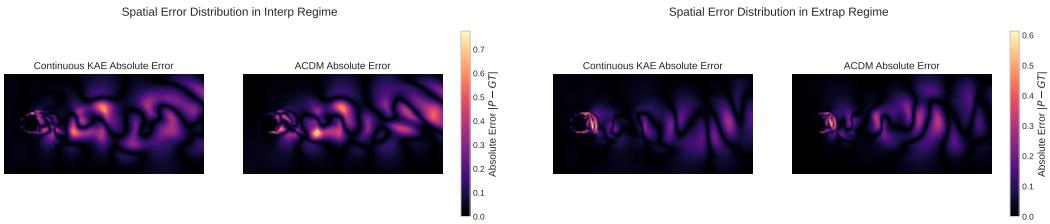
H.1 Comparative Analysis: KAE vs. Diffusion Models

The fundamental difference between KAE and the ACDM diffusion model lies in how they handle high-frequency information and temporal evolution:

- **KAE (Deterministic Stability):** KAE is constrained by a linear Koopman operator in a continuous latent space. This structure ensures asymptotic stability, meaning errors do not grow exponentially over time. However, the deterministic L_2 objective leads to a "spectral bias" where the model prefers to smooth out high-frequency spatial discontinuities to ensure global structural alignment.
- **Diffusion Models (Stochastic Detail):** ACDM uses a stochastic process to synthesize fine-scale turbulent textures. While this allows the model to match the sharp 'look' of the ground truth, it lacks the rigid mathematical constraints of KAE. Consequently, minor stochastic errors compound over time, leading to widespread spatial noise and phase divergence.

H.2 Visual Error Localization across All Regimes

The difference maps reveal two distinct archetypes of model failure. Errors generated by KAE are highly localized along sharp discontinuities such as shock fronts and vortex edges. In contrast, errors from the ACDM baseline are more diffusely distributed throughout the spatial domain.

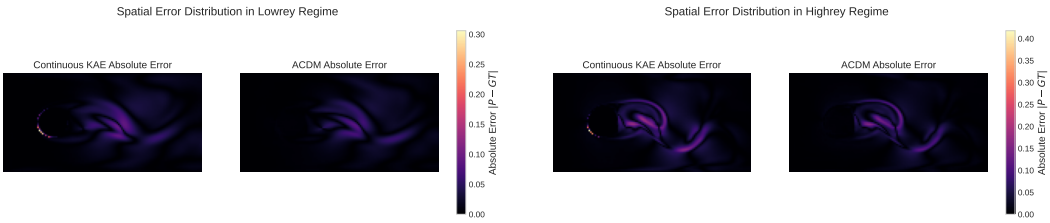


(a) Transonic Interpolation (b) Transonic Extrapolation

Figure 20: Spatial error distribution in Transonic regimes. KAE errors are concentrated precisely at the sharp shock fronts, while ACDM exhibits broader spatial noise.



Figure 21: Spatial error distribution in the long-rollout regime. KAE maintains structural stability with localized errors, while ACDM shows diffuse error growth across the wake.



(a) Incompressible (Low Reynolds) (b) Incompressible (High Reynolds)

Figure 22: Comparison of error magnitudes across incompressible flow regimes. Note the higher error localization on the obstacle boundary for KAE.

Algorithm 1 KAE 4D-Var. All weights frozen; the control is the latent alone. The propagators are built once, before the loop, because $\mathbf{K}$ does not change during it.

Input: observations $\{y_i\}_{i=1}^N$, masks $\{M_i\}$, leads $\{\tau_i\}$; frozen $\mathcal{E}, \mathcal{D}, \mathbf{K}$; step size η; budget T
Output: $\hat{u}(t_0)$

$\Phi_i \leftarrow \exp(\mathbf{K}\,\tau_i)$ for $i = 1, \dots, N$
if warm start **then**
  $\tilde{y} \leftarrow y_1 +$ null history
  $z \leftarrow \exp(-\mathbf{K}\,\tau_\star)\,\mathcal{E}(\tilde{y})$
else
  $z \leftarrow \mathbf{0}$
end if
for $t = 1, \dots, T$ **do**
  $J \leftarrow \frac{1}{N}\sum_{i=1}^N \left\| M_i \odot (\mathcal{D}(\Phi_i z) - y_i) \right\|^2$
  $z \leftarrow \mathrm{Adam}(z,\ \nabla_z J,\ \eta)$
end for
return $\mathcal{D}(z)$

Algorithm 2 Autoregressive 4D-Var (U-Net, FNO). The control is the k-frame window ending at t_0. The rollout is rebuilt inside every iteration, and the leads n_i must be integers.

Input: observations $\{y_i\}_{i=1}^N$, masks $\{M_i\}$, integer leads $\{n_i\}$; frozen F_θ; window k; step size η; budget T
Output: $\hat{u}(t_0)$

$c \leftarrow \mathbf{0} \in \mathbb{R}^{k \times C \times H \times W}$
for $t = 1, \dots, T$ **do**
  $x_{1:k} \leftarrow c$
  for $s = k, \dots, k + \max_i n_i - 1$ **do**
    $x_{s+1} \leftarrow F_\theta(x_{s-k+1:s})$
  end for
  $J \leftarrow \frac{1}{N}\sum_{i=1}^N \left\| M_i \odot (x_{k+n_i} - y_i) \right\|^2$
  $c \leftarrow \mathrm{Adam}(c,\ \nabla_c J,\ \eta)$
end for
return c_k

I Data assimilation training and evaluation protocols

I.1 Algorithms used

I.2 Kuramoto–Sivashinsky

Data. $u_t = -uu_x - u_{xx} - u_{xxxx}$ on a periodic domain, of 64 points, stored every $\Delta t = 0.1$. Training uses 100 trajectories of 1000 frames split $80/10/10$. Assimilation uses a separate set of 64 trajectories generated from the same simulator configuration, plus 32 trajectories of 5000 frames for post-assimilation forecasts. There is no overlap between these and the training and validation trajectories. Lyapunov time is $T_L = 22.6$ time units, measured on 12 held-out trajectories.

Training. KAE trains for 200 epochs, batch 1024, Adam at 10^{-3} with 20 epochs of warm-up and linear decay to 2×10^{-5}, on a 10-frame rollout (best validation loss 1.473×10^{-3}). The U-Net baseline uses the same data, settings, objective, rollout and optimiser schedule. The score network trains for 128 epochs, batch 128, Adam at 2×10^{-4} with weight decay 10^{-3}, on segments of nine frames under the cosine VP schedule. All three see the same exact training split.

Models. *KAE.* Because $\mathbf{K}_{cont}$ is frozen, the N matrices $e^{\mathbf{K}_{cont}\tau_i}$ are computed once before the optimisation loop, each iteration then costs N matrix–vector products and N decoder evaluations, with no intermediate integration state stored or differentiated through. Observation times do not need to lie on the training grid. As an optimiser, we use Adam at 10^{-2}, 8000 iterations.

U-Net. A 1-D autoregressive map F_θ advancing the state by Δt. The target is the 64-point physical state x_0 and $\mathcal{H}_{x_0}(\tau) = F_\theta^{\tau/\Delta t}(x_0)$, so the rollout sits inside every iteration and must be backpropagated. We use the same optimiser, Adam at 10^{-2}, 8000 iterations.

SDA. A score network trained here on nine-frame segments ($k = 4$), with Algorithm 2 of Rozet and Louppe (2023) composing the local scores into a score over the whole trajectory. Observations use a Gaussian likelihood to return the Tweedie estimate of the clean trajectory. Sampling uses 128 exponential-integrator steps with four Langevin corrections ($\tau = 0.5$) and $\sigma_y = 0.05$ for clean observations. There is no control variable and no iteration count, so the reported value is independent of the optimization budget.

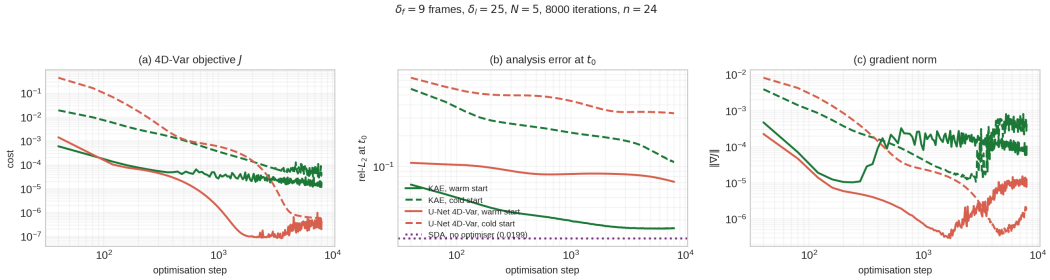


Figure 23: Convergence, analysis and gradient visualization for KAE and UNet 4D-Var optimization process for a cold and warm start in the optimization process, across 8000 steps.

I.3 Transonic cylinder flow

Data. Compressible Kármán vortex streets past a cylinder at $Re = 10^4$, 128×64, four fields (ρ, p, v_x, v_y) plus an obstacle mask, from Kohl et al. (2026). The raw interval $\Delta t = 0.05$ s is subsampled to 0.1 s. Training covers $Ma \in [0.53, 0.63] \cup [0.69, 0.89]$ (32 sequences of 1000 frames). Assimilation is run on Tra_{int} ($Ma \in \{0.66, 0.67, 0.68\}$, 6 sequences of 60 frames), which sits inside the gap in the training range and was seen by no model.

Training. KAE is trained on the transonic training set with a latent space of size 128. The four baselines are the pretrained checkpoints (on the same data splits) of Kohl et al. (2026), unmodified: U-Net, FNO with 32 modes, and ACDM and ACDM$_{\text{ncn}}$.

Models. *KAE.* As on KS, with two differences. Two conditioning frames only are encoded separately and aggregated by a Transformer encoder with sinusoidal positional encodings, matching the AR-2 conditioning of the diffusion baseline. The operator is conditioned on the Mach number through a LoRA update (Hu et al., 2022), so $\mathbf{K}_{cont}$ is fixed per problem but differs between problems. The target is again $z_0 \in \mathbb{R}^{128}$. For this, we use Adam at 10^{-1}, 2000 iterations.

U-Net and FNO 4D-Var. Both are autoregressive maps, where the target is in the model's own normalised space and propagation is $\tau/\Delta t$ passes of the network inside each iteration. We use a tuned Adam at 3×10^{-2}, 2000 iterations.

ACDM + SDA. ACDM is not run autoregressively here. Its network predicts ϵ for all three frames it sees (two conditioning frames and the target), and all three outputs are kept as a local score over a three-frame window, composed over $t_0, \ldots, t_0 + \delta_l$ by the same algorithm used on KS. The likelihood covariance is $\sigma_y^2 + (\sigma^2/\mu^2)\Gamma$ with $\Gamma = 0.3$ and $\sigma_y = \max(\sigma, 0.05)$. There are three deviations as compared to the sampler used on KS: a three-frame blanket instead of five, 20 denoising steps instead of 256, and ACDM's own DDPM schedule rather than an exponential integrator. These were chosed for consistency with the previous models and optimized for ACDM's best DA results.

ACDM$_{\text{ncn}}$. The same architecture trained with clean conditioning, so its outputs on the conditioning frames are not a score and the composition is ill-posed. We report it once, at 2.4 analysis error, and exclude it from every sweep.

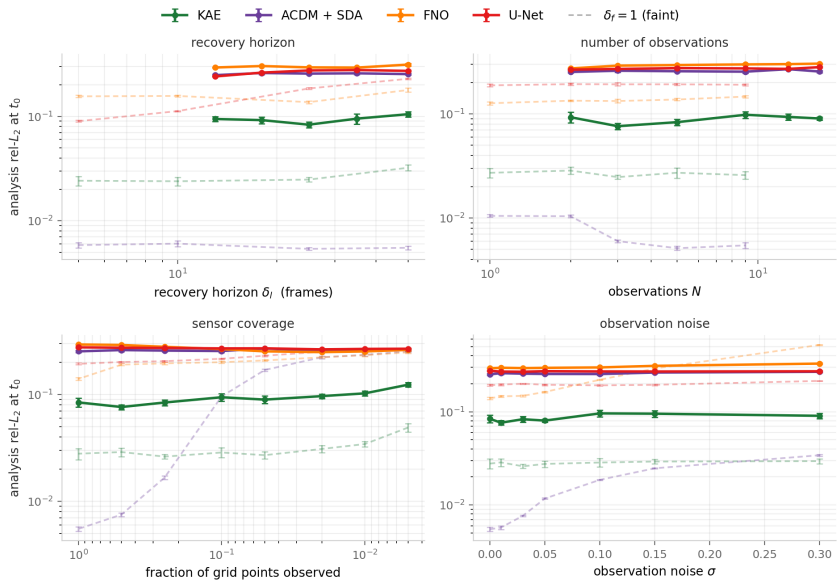


Figure 24: Tra Sweeps at $\delta_f = 1$ and $\delta_f = 9$.

J Data Assimilation Ablations

TRA sweep ($\delta_f = 9$), first $\rightarrow$ last	KAE	ACDM	FNO	U-Net
δ_l: 13 $\rightarrow$ 50 frames	0.094±0.005	0.248±0.005	0.294±0.002	0.241±0.002
	$\rightarrow$ 0.105±0.006	$\rightarrow$ 0.254±0.004	$\rightarrow$ 0.313±0.010	$\rightarrow$ 0.272±0.004
N: 2 $\rightarrow$ 17	0.093±0.011	0.253±0.005	0.273±0.002	0.266±0.002
	$\rightarrow$ 0.090±0.003	$\rightarrow$ 0.255±0.006	$\rightarrow$ 0.304±0.003	$\rightarrow$ 0.281±0.005
obs. fraction: 1 $\rightarrow$ 0.005	0.084±0.008	0.253±0.005	0.293±0.001	0.275±0.004
	$\rightarrow$ 0.123±0.006	$\rightarrow$ 0.261±0.007	$\rightarrow$ 0.261±0.005	$\rightarrow$ 0.267±0.003
noise σ: 0 $\rightarrow$ 0.3	0.084±0.008	0.253±0.005	0.293±0.001	0.275±0.004
	$\rightarrow$ 0.091±0.006	$\rightarrow$ 0.269±0.004	$\rightarrow$ 0.329±0.005	$\rightarrow$ 0.272±0.005

Table 3: TRA, the same sweeps with δ_f pinned at 9 frames; first and last point of each sweep, mean ± standard error.

$\delta_f = 1$

δ_l (t.u.)	KAE	SDA	U-Net
0.5	0.0027±0.0001	0.0079±0.0003	0.1877±0.0046
1	0.0027±0.0001	0.0074±0.0002	0.1911±0.0052
2.5	0.0033±0.0002	0.0088±0.0003	0.1924±0.0038
5	0.0040±0.0002	0.0099±0.0003	0.1886±0.0057
10	0.0059±0.0004	0.0106±0.0004	0.1593±0.0089
20	0.0068±0.0004	0.0129±0.0006	0.1688±0.0125
40	0.0082±0.0005	0.0146±0.0008	0.8355±0.0108
70	0.0096±0.0006	0.0182±0.0013	0.9930±0.0072

$\delta_f = 9$

δ_l (t.u.)	KAE	SDA	U-Net
1.3	0.0234±0.0027	0.0146±0.0007	0.4230±0.0074
2.5	0.0191±0.0025	0.0209±0.0013	0.4576±0.0105
5	0.0244±0.0022	0.0216±0.0014	0.4452±0.0148
10	0.0294±0.0033	0.0219±0.0013	0.4291±0.0139
20	0.0424±0.0053	0.0237±0.0018	0.3565±0.0178
40	0.0583±0.0077	0.0243±0.0016	0.8788±0.0127
70	0.0766±0.0045	0.0278±0.0023	0.9900±0.0077

Table 4: KS, recovery horizon δ_l. Mean ± standard error over the problems at each point, everything else pinned.

$\delta_f = 1$

N	KAE	SDA	U-Net
2	0.0089±0.0005	0.0181±0.0008	0.0873±0.0059
3	0.0026±0.0001	0.0133±0.0006	0.0931±0.0056
5	0.0022±0.0001	0.0091±0.0003	0.0965±0.0057
8	0.0022±0.0001	0.0074±0.0002	0.0994±0.0058

$\delta_f = 9$

N	KAE	SDA	U-Net
2	0.0626±0.0039	0.0229±0.0013	0.3189±0.0071
3	0.0261±0.0023	0.0225±0.0013	0.3245±0.0076
5	0.0105±0.0009	0.0215±0.0012	0.3293±0.0081
9	0.0112±0.0010	0.0190±0.0010	0.3318±0.0084
13	0.0115±0.0010	0.0184±0.0010	0.3333±0.0086

Table 5: KS, number of observations N. Mean ± standard error over the problems at each point, everything else pinned.

$\delta_f = 1$

σ	KAE	SDA	U-Net
0.0	0.0024±0.0001	0.0098±0.0003	0.0960±0.0057
0.01	0.0065±0.0002	0.0038±0.0001	0.0953±0.0036
0.05	0.0354±0.0017	0.0135±0.0005	0.1748±0.0059
0.1	0.0740±0.0040	0.0242±0.0010	0.2587±0.0071
0.2	0.1419±0.0074	0.0417±0.0018	0.3780±0.0102
0.4	0.2777±0.0131	0.0770±0.0039	0.5831±0.0152

$\delta_f = 9$

σ	KAE	SDA	U-Net
0.0	0.0142±0.0016	0.0214±0.0007	0.3671±0.0121
0.01	0.0554±0.0019	0.0120±0.0003	0.3695±0.0118
0.05	0.2199±0.0096	0.0257±0.0011	0.4962±0.0180
0.1	0.3500±0.0186	0.0389±0.0019	0.7886±0.0429
0.2	0.4821±0.0279	0.0617±0.0037	1.5046±0.0902
0.4	0.6686±0.0379	0.1027±0.0066	2.8842±0.1582

Table 6: KS, observation noise σ (Gaussian). Mean ± standard error over the problems at each point, everything else pinned.

$\delta_f = 1$

sensors	KAE	SDA	U-Net
100%	0.0022±0.0001	0.0094±0.0002	0.0952±0.0060
50%	0.0067±0.0005	0.0147±0.0005	0.1084±0.0053
25%	0.0348±0.0029	0.0257±0.0014	0.1690±0.0077
10%	0.5734±0.0248	0.2213±0.0368	1.0289±0.0246

$\delta_f = 9$

sensors	KAE	SDA	U-Net
100%	0.0121±0.0015	0.0215±0.0009	0.3491±0.0100
50%	0.0251±0.0015	0.0281±0.0015	0.3683±0.0118
25%	0.0836±0.0059	0.0378±0.0023	0.4110±0.0128
10%	0.5752±0.0241	0.2471±0.0409	1.0176±0.0224

Table 7: KS, sensor coverage (at $N = 5$). Mean ± standard error over the problems at each point, everything else pinned.

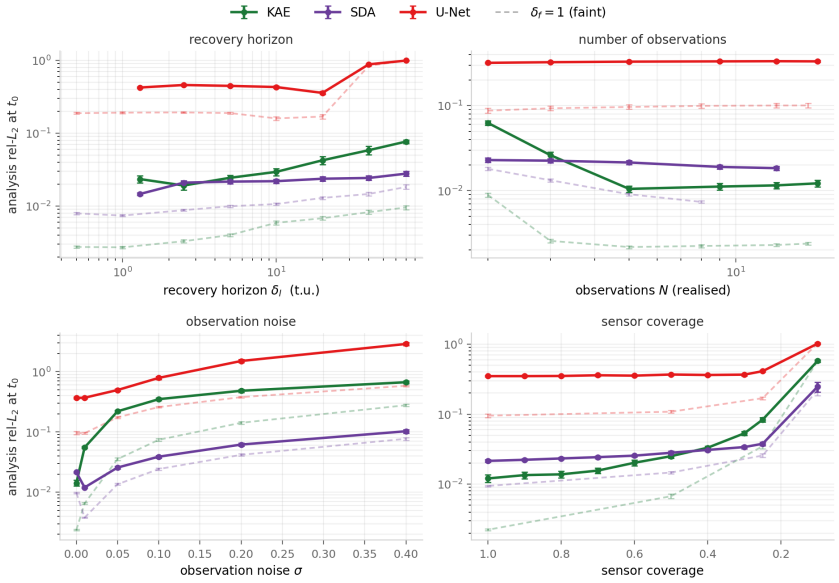


Figure 25: KS Sweeps at $\delta_f = 1$ and $\delta_f = 9$.

Sweep ($\delta_f = 9$)	KAE	ACDM	U-Net
δ_l (frames)	$0.0944 \pm 0.0055 \to 0.1048 \pm 0.0060$	$0.2481 \pm 0.0050 \to 0.2537 \pm 0.0040$	$0.2409 \pm 0.0022 \to 0.2724 \pm 0.0037$
N	$0.0928 \pm 0.0109 \to 0.0902 \pm 0.0034$	$0.2526 \pm 0.0053 \to 0.2548 \pm 0.0060$	$0.2657 \pm 0.0020 \to 0.2810 \pm 0.0049$
obs. fraction	$0.0840 \pm 0.0079 \to 0.1230 \pm 0.0061$	$0.2526 \pm 0.0053 \to 0.2615 \pm 0.0068$	$0.2754 \pm 0.0039 \to 0.2673 \pm 0.0034$
noise σ	$0.0840 \pm 0.0079 \to 0.0906 \pm 0.0061$	$0.2526 \pm 0.0053 \to 0.2686 \pm 0.0042$	$0.2754 \pm 0.0039 \to 0.2721 \pm 0.0048$

Table 8: The same sweeps at $\delta_f = 9$, mean $\pm$ standard error at each endpoint. ACDM/KAE never drops below 2.1.

K Continuous-Time Generalization of Discrete Koopman Consistency

Consistent Koopman autoencoders (Azencot et al., 2020) learn separate forward and backward latent operators $A, B \in \mathbb{R}^{N_z \times N_z}$ satisfying

$$z_{n+1} = Az_n, \qquad z_n = Bz_{n+1}. \tag{19}$$

Their supervised consistency objective is

$$\mathcal{L}_{\text{disc}} = \|Az_n - z_{n+1}\|_2^2 + \|Bz_{n+1} - z_n\|_2^2. \tag{20}$$

In our continuous-time formulation, latent dynamics are governed by the linear system

$$\frac{dz}{dt} = Kz, \tag{21}$$

whose exact solution over a time interval Δt is

$$z(t + \Delta t) = e^{K\Delta t} z(t). \tag{22}$$

Defining the discrete forward operator as

$$A := e^{K\Delta t}, \tag{23}$$

immediately recovers forward evolution:

$$z_{n+1} = Az_n. \tag{24}$$

Backward evolution follows from the same generator evaluated at negative time:

$$z(t - \Delta t) = e^{-K\Delta t} z(t), \tag{25}$$

which implies

$$B := e^{-K\Delta t} = A^{-1}. \tag{26}$$

Substituting these expressions into (20) gives

$$\mathcal{L}_{\text{disc}} = \|e^{K\Delta t} z_n - z_{n+1}\|_2^2 + \|e^{-K\Delta t} z_{n+1} - z_n\|_2^2, \tag{27}$$

which is exactly the latent forward–backward consistency loss used in our model.

Unlike discrete formulations, where A and B are learned independently and invertibility must be encouraged through explicit regularization, the continuous generator enforces

$$B = A^{-1} \tag{28}$$

by construction. Therefore, the proposed latent consistency loss is the exact continuous-time counterpart of discrete consistent Koopman training under matrix exponential flow.

L Ablation Studies: Operator Parameterization, Temporal Weighting and Solvers

To validate our architectural design choices, we conduct an ablation study analyzing the parameterization of the continuous Koopman generator ($\mathbf{K}_{\text{cont}}$) and the temporal weighting schedule used during rollout training. The results are summarized in Table 9.

LoRA vs. Full-Rank MLP Parameterization. We compare our default Low-Rank Adaptation (LoRA) formulation against a full-rank MLP parameterization, where a neural network directly predicts the entire $N_z \times N_z$ Koopman generator matrix from the conditioning parameters. While both approaches strictly preserve the linearity of the latent state evolution (enabling $O(1)$ matrix exponentiation), their generalization capabilities differ significantly.

As shown in Table 9, the MLP parameterization heavily degrades performance on extrapolation tasks (e.g., MSE increases from 1.3 to 10.4 on Inc_{low} and 2.2 to 3.6 on Tra_{ext}). This confirms our structural hypothesis: predicting a full-rank matrix directly from physical parameters is highly prone to overfitting the training regimes. LoRA resolves this by anchoring the dynamics to a globally stable, invariant base matrix $\mathbf{K}_0$, acting as a powerful structural regularizer that enables robust generalization to unseen Reynolds and Mach numbers, while simultaneously reducing the parameter footprint from $O(N_z^2)$ to $O(2rN_z)$.

Cosine vs. Uniform Temporal Weighting. We further ablate the temporal weighting schedule applied to the rollout prediction loss ($\mathcal{L}_{\text{pred}}$). On the relatively smooth Incompressible flow, both schedules converge to identical minima, demonstrating general robustness. However, on the highly chaotic Transonic dataset, the cosine schedule strictly outperforms uniform weighting, improving the long-horizon 240-step MSE from 17.0×10^{-3} to 14.9×10^{-3}. This validates our training strategy: because autoregressive rollouts are highly susceptible to compounding errors, applying a decaying cosine schedule forces the model to prioritize strict local phase alignment in the early epochs, establishing a correct base trajectory before optimizing for long-term asymptotic stability.

Model Configuration		Inc_{low} ($\times 10^{-4}$)		Inc_{high} ($\times 10^{-5}$)		Tra_{ext} ($\times 10^{-3}$)		Tra_{int} ($\times 10^{-3}$)		Tra_{long} ($\times 10^{-3}$)	
Conditioning	**Weighting**	MSE	LSiM	MSE	LSiM	MSE	LSiM	MSE	LSiM	MSE	LSiM
LoRA (Proposed)	Cosine	$\mathbf{1.3 \pm 1.7}$	$\mathbf{6.1 \pm 4.8}$	2.9 ± 1.1	1.7 ± 0.3	$\mathbf{2.2 \pm 0.9}$	1.8 ± 0.3	$\mathbf{5.2 \pm 2.4}$	$\mathbf{2.1 \pm 0.6}$	$\mathbf{14.9 \pm 1.3}$	5.0 ± 0.4
LoRA	Uniform	$\mathbf{1.3 \pm 1.7}$	$\mathbf{6.1 \pm 4.8}$	2.9 ± 1.1	1.7 ± 0.3	2.5 ± 0.8	$\mathbf{1.7 \pm 0.3}$	6.5 ± 1.6	2.2 ± 0.4	17.0 ± 2.3	$\mathbf{4.1 \pm 0.4}$
MLP (Full-Rank)	Cosine	10.4 ± 17.5	12.0 ± 12.8	21.4 ± 7.1	4.5 ± 0.7	3.6 ± 1.0	$\mathbf{1.7 \pm 0.3}$	5.7 ± 3.0	2.2 ± 0.6	15.1 ± 1.9	5.2 ± 0.5
Base (Unconditional)	Cosine	116.5 ± 31.0	48.3 ± 2.5	2991.2 ± 12.5	54.5 ± 0.2	13.9 ± 0.8	3.3 ± 0.2	21.0 ± 2.7	3.7 ± 0.4	18.1 ± 1.7	6.3 ± 0.2

Table 9: Ablation study on operator parameterization and temporal weighting. The proposed LoRA + Cosine configuration provides the optimal balance of interpolation accuracy, robust extrapolation, and extreme long-horizon stability.

Structural Constraints and Consistency. We incrementally removed key architectural constraints to isolate their impact on long-horizon stability (Table 10). Each constraint contributes a comparable improvement of roughly 3–4 points on Tra_{long} MSE, with the constraints together accounting for a $\sim 30\%$ reduction relative to the unconstrained baseline (14.9 vs. $\sim$ 19–20 $\times 10^{-3}$). This flatness is a feature, not a weakness: no single trick dominates, indicating a well-balanced system. The strongest single ablation is rollout training depth: reducing to $R = 1$ (single-step training) causes catastrophic divergence on the KS equation, confirming that multi-step rollout training is the non-negotiable foundation (Figure 6a–b).

- **Continuous-Time Invertibility (Azencot Consistency):** Azencot et al. (2020) enforce discrete forward (A) and backward (B) consistency via $AB \approx I$. We enforce the continuous-time equivalent by integrating the dynamics at Δt and $-\Delta t$, penalizing the discrepancy between the forward/backward continuous flow and the true encoded representations (Eq. 13). Removing this constraint increases 240-step MSE from 14.9×10^{-3} to 18.5×10^{-3}.
- **History Encoder (Temporal context aggregation):** Fluid observations are inherently non-Markovian (e.g., velocity alone does not fully capture pressure/density dynamics). Removing the history encoder forces a strictly Markovian initialization, degrading long-term MSE to 18.6×10^{-3}.
- **Structural Regularization (Physics-Informed):** Sobolev (spatial gradients) and Fourier spectral norms preserve sharp wavefronts and shedding frequencies. Removing these priors degrades long-horizon MSE to 19.3×10^{-3}.

L.1 Is the forecasting–assimilation difference an architecture effect?

The comparison above runs across architectures, which leaves open whether the difference is a property of the training objective or an accident of how five different models happen to behave.

Method	Tra_{ext}		Tra_{int}		Tra_{long}	
	MSE ($\times 10^{-3}$)	LSiM ($\times 10^{-1}$)	MSE ($\times 10^{-3}$)	LSiM ($\times 10^{-1}$)	MSE ($\times 10^{-3}$)	LSiM ($\times 10^{-1}$)
w/o Structural Regularization	2.5 ± 0.5	3.5 ± 0.4	5.6 ± 2.3	3.7 ± 0.2	19.3 ± 1.2	6.6 ± 0.4
w/o Latent Consistency	2.6 ± 0.6	3.3 ± 0.3	5.8 ± 2.7	4.0 ± 0.3	18.5 ± 1.4	6.5 ± 0.3
w/o Directional Cosine	$\mathbf{2.4 \pm 0.6}$	3.5 ± 0.3	$\mathbf{5.3 \pm 2.4}$	$\mathbf{3.6 \pm 0.4}$	$\mathbf{17.8 \pm 1.5}$	6.6 ± 0.3
w/o Latent norm	2.5 ± 0.6	$\mathbf{3.3 \pm 0.2}$	5.9 ± 2.3	3.7 ± 0.4	18.3 ± 2.0	$\mathbf{6.0 \pm 0.2}$
w/o History encoder	2.6 ± 0.8	3.6 ± 0.5	5.9 ± 3.5	3.7 ± 0.3	18.6 ± 0.5	7.4 ± 0.3

Table 10: Ablation study on Continuous Linear 128 for trajectory forecasting tasks. Best values are highlighted in bold.

Variant	forecast	assimilation
original	14.9 ± 1.3	0.0907 ± 0.0102
w/o directional cosine	17.8 ± 1.5	0.0850 ± 0.0105
w/o latent norm	18.3 ± 2.0	0.0801 ± 0.0076
w/o latent consistency	18.5 ± 1.4	0.0846 ± 0.0145
w/o history encoder	18.6 ± 0.5	0.0733 ± 0.0094
w/o structural regularisation	19.3 ± 1.2	0.0767 ± 0.0103

Table 11: The same five checkpoints scored as forecasters and as assimilators; only the loss differs between them, with identical architecture, data, schedule and budget. Forecasting is the 240-step MSE ($\times 10^{-3}$), mean $\pm$ s.d.. Assimilation is the analysis error at $\delta_f = 9$, mean $\pm$ s.d. over the 26 points of the δ_l, N, coverage and σ sweeps.

We answer the question by scoring five variations that differ only in which loss term was dropped (identical architecture, data, schedule and budget) first as forecasters and then as assimilators.

Table 11 reproduces the effect under this controlled ablation. The variant that forecasts best, dropping the directional-cosine term, assimilates worst, and the variant that forecasts worst, dropping structural regularisation, is among the two best assimilators. The loss terms that buy long-horizon forecast accuracy are therefore not the terms that make the model invertible. For $\delta_l = 25$ frames and $N = 5$, KAE achieves lower assimilation error than SDA across $\delta_f = 1$–20 frames.

M Irregular Temporal Sampling and Rollout Stability: Kuramoto-Sivashinsky

The main results and figure for this experiment (rollout-training necessity and irregular-timestep robustness on the Kuramoto–Sivashinsky equation) are presented in the main text: the rollout-scaling analysis in Paragraph 5 and the dropout experiment in Paragraph 5. The KS dataset details are given in Appendix B.3. This appendix section is retained as an anchor for cross-references from the appendix ablation tables. An ablation study with the results in Figures 6a–6b is provided in Table 12, where we computed the MSE at for each individual training regime, from a training horizon of 1 to 10, from a dropout probability of 0.1 to 0.9 and two aditional experiments: 1) we have completely eliminated the latent regularization (No Latent); 2) we have only kept the L_2 latent norm between the true and predicted encodings, disregarding any physics or other structural losses (L_2 Latent).

N ODE Solvers and Physical Stiffness

To validate the continuous-time nature of the learned generator ($\mathbf{K}_{\text{cont}}$) and its robustness to irregular temporal sampling, we evaluated the latent ODE using multiple numerical integrators—including the adaptive-step `Dopri5`—across integration step sizes ranging from $\Delta t = 0.05$s up to $\Delta t = 1.00$s (a $10\times$ extrapolation beyond the training resolution). The results are summarized in Table 13.

- **Solver Parity at Small Steps:** At step sizes $\Delta t \leq 0.15$s, all solvers, including the adaptive `Dopri5`, perform comparably to standard `RK4`.
- **Physical Stiffness:** At larger step sizes ($\Delta t \geq 0.50$s), explicit solvers like the first-order `Euler` method diverge numerically (yielding $> 10^{13}$ error), and the second-order

Table 12: Ablation study on rollout horizon and dropout rate. We report mean MSE with 95% confidence intervals. Lower is better.

Setting	Mean MSE	95% CI
Rollout Horizon		
Rollout 1	89.866175	± 2.777168
Rollout 2	0.299346	± 0.045892
Rollout 3	0.315966	± 0.049891
Rollout 4	0.218818	± 0.036771
Rollout 5	0.192893	± 0.032105
Rollout 6	0.163325	± 0.029480
Rollout 7	0.147364	± 0.027526
Rollout 8	0.122315	± 0.024598
Rollout 9	0.108214	± 0.021765
Rollout 10	**0.103406**	± 0.021593
Dropout Rate		
Dropout 0.1	0.109436	± 0.021946
Dropout 0.2	0.109879	± 0.022887
Dropout 0.3	**0.105531**	± 0.022342
Dropout 0.4	0.112445	± 0.023031
Dropout 0.5	0.116533	± 0.024497
Dropout 0.6	0.133597	± 0.024915
Dropout 0.7	0.135008	± 0.029283
Dropout 0.8	0.142115	± 0.030503
Dropout 0.9	0.134434	± 0.026484
Structural changes		
No Latent	2.401261	± 0.141642
L_2 Latent	0.314729	± 0.035736

`Midpoint` method exhibits significantly inflated errors. This occurs because the Koopman operator accurately captures dissipative, high-frequency fluid modes, resulting in a mathematically "stiff" ODE.

- **Higher-Order Stability:** Higher-order methods (`RK4`, `Dopri5`) possess stability regions large enough to encompass these stiff dissipative eigenvalues, maintaining bounded physical stability even at extreme 1.00s jump steps.

Table 13: Comparison of MSE and LSIM across solvers for step sizes $0.05s, 0.10s, 0.15s, 0.20s, 0.25s, 0.30s, 0.50s$ and $1.00s$ (mean $\pm$ std).

Step	Solver	MSE			LSIM		
		Extrap	Interp	Longer	Extrap	Interp	Longer
0.05	RK4	1.9 ± 1.1	5.8 ± 3.4	14.6 ± 0.9	1.5 ± 0.2	2.1 ± 0.5	5.3 ± 0.2
	Dopri5	1.9 ± 1.1	5.8 ± 3.4	14.6 ± 0.9	1.5 ± 0.2	2.1 ± 0.5	5.3 ± 0.2
	Bosh3	1.9 ± 1.0	5.8 ± 3.3	14.4 ± 0.9	1.5 ± 0.2	2.1 ± 0.5	5.5 ± 0.2
	Adaptive Heun	1.9 ± 1.1	5.8 ± 3.4	14.6 ± 0.9	1.5 ± 0.2	2.1 ± 0.5	5.3 ± 0.2
	Midpoint	1.8 ± 1.0	5.9 ± 3.4	14.6 ± 1.1	1.5 ± 0.2	2.1 ± 0.5	5.3 ± 0.2
	Euler	1.8 ± 1.1	6.2 ± 3.6	18.8 ± 1.3	1.4 ± 0.3	2.0 ± 0.5	3.7 ± 0.2
	Explicit Adams	1.9 ± 1.1	5.8 ± 3.4	14.6 ± 0.9	1.5 ± 0.2	2.1 ± 0.5	5.3 ± 0.2
0.10	RK4	2.1 ± 1.1	5.5 ± 3.3	14.6 ± 0.9	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Dopri5	2.0 ± 1.1	5.5 ± 3.3	14.6 ± 0.9	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Bosh3	2.1 ± 1.1	5.5 ± 3.2	14.5 ± 0.9	1.5 ± 0.2	2.0 ± 0.5	5.4 ± 0.2
	Adaptive Heun	2.0 ± 1.1	5.5 ± 3.3	14.6 ± 0.9	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Midpoint	1.7 ± 1.0	5.9 ± 3.3	14.7 ± 1.4	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Euler	2.9 ± 1.2	6.0 ± 3.3	19.0 ± 1.3	1.8 ± 0.2	2.1 ± 0.5	4.2 ± 0.3
	Explicit Adams	2.1 ± 1.1	5.5 ± 3.3	14.6 ± 0.9	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
0.15	RK4	2.3 ± 1.2	5.3 ± 3.2	14.6 ± 0.8	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Dopri5	2.3 ± 1.2	5.3 ± 3.2	14.6 ± 0.9	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Bosh3	2.3 ± 1.2	5.3 ± 3.2	14.5 ± 0.8	1.5 ± 0.2	2.0 ± 0.5	5.4 ± 0.2
	Adaptive Heun	2.3 ± 1.2	5.3 ± 3.2	14.6 ± 0.9	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Midpoint	1.6 ± 0.9	6.0 ± 3.3	15.0 ± 1.9	1.4 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Euler	5.9 ± 1.2	6.2 ± 2.6	221.3 ± 101.8	2.3 ± 0.1	2.3 ± 0.4	14.4 ± 0.6
	Explicit Adams	2.3 ± 1.2	5.3 ± 3.2	14.6 ± 0.8	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
0.20	RK4	2.6 ± 1.2	5.1 ± 3.1	14.6 ± 0.8	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Dopri5	2.5 ± 1.2	5.1 ± 3.1	14.6 ± 0.8	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Bosh3	2.5 ± 1.2	5.1 ± 3.1	14.5 ± 0.8	1.5 ± 0.2	2.0 ± 0.5	5.4 ± 0.2
	Adaptive Heun	2.5 ± 1.2	5.1 ± 3.1	14.6 ± 0.8	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
	Midpoint	1.6 ± 0.8	6.4 ± 3.3	15.6 ± 2.2	1.4 ± 0.2	2.0 ± 0.5	5.1 ± 0.2
	Euler	9.9 ± 1.1	7.1 ± 2.4	$(3.36 \pm 1.49) \times 10^5$	2.7 ± 0.1	2.6 ± 0.3	16.2 ± 0.5
	Explicit Adams	2.6 ± 1.2	5.1 ± 3.1	14.6 ± 0.8	1.5 ± 0.2	2.0 ± 0.5	5.3 ± 0.2
0.25	RK4	2.9 ± 1.2	5.0 ± 3.0	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.4 ± 0.3
	Dopri5	2.9 ± 1.2	5.0 ± 3.0	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.3 ± 0.3
	Bosh3	2.9 ± 1.2	5.0 ± 3.0	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.4 ± 0.3
	Adaptive Heun	2.9 ± 1.2	5.0 ± 3.0	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.3 ± 0.3
	Midpoint	1.9 ± 0.5	7.1 ± 3.2	17.0 ± 1.8	1.4 ± 0.2	2.0 ± 0.5	4.4 ± 0.2
	Euler	14.0 ± 0.9	8.7 ± 2.9	$(1.44 \pm 0.58) \times 10^8$	2.9 ± 0.1	2.8 ± 0.3	16.3 ± 0.3
	Explicit Adams	2.9 ± 1.2	5.0 ± 3.0	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.4 ± 0.3
0.30	RK4	3.4 ± 1.3	4.9 ± 2.9	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.4 ± 0.3
	Dopri5	3.2 ± 1.3	4.9 ± 2.9	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.4 ± 0.3
	Bosh3	3.2 ± 1.3	4.9 ± 2.9	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.4 ± 0.3
	Adaptive Heun	3.2 ± 1.3	4.9 ± 2.9	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.4 ± 0.3
	Midpoint	2.6 ± 0.3	8.0 ± 3.2	19.9 ± 2.0	1.4 ± 0.2	2.0 ± 0.5	3.7 ± 0.2
	Euler	17.2 ± 0.7	10.9 ± 3.4	$(1.94 \pm 0.77) \times 10^{10}$	3.0 ± 0.2	2.9 ± 0.3	16.8 ± 0.4
	Explicit Adams	3.4 ± 1.3	4.9 ± 2.9	14.6 ± 0.8	1.5 ± 0.2	1.9 ± 0.5	5.4 ± 0.3
0.50	RK4	6.2 ± 1.1	5.6 ± 2.8	14.9 ± 1.4	1.8 ± 0.2	2.0 ± 0.4	5.9 ± 0.3
	Dopri5	5.0 ± 1.3	5.3 ± 2.7	14.8 ± 1.0	1.7 ± 0.3	1.9 ± 0.4	5.5 ± 0.3
	Bosh3	5.0 ± 1.3	5.3 ± 2.7	14.8 ± 1.0	1.7 ± 0.2	1.9 ± 0.4	5.5 ± 0.3
	Adaptive Heun	5.0 ± 1.3	5.3 ± 2.7	14.8 ± 1.0	1.7 ± 0.2	1.9 ± 0.4	5.5 ± 0.3
	Midpoint	4.7 ± 0.2	10.6 ± 2.7	$(2.86 \pm 0.69) \times 10^4$	2.1 ± 0.1	2.2 ± 0.3	16.1 ± 0.5
	Euler	19.1 ± 0.8	20.9 ± 3.3	$(4.88 \pm 1.22) \times 10^{14}$	3.1 ± 0.2	3.4 ± 0.2	16.9 ± 0.4
	Explicit Adams	6.2 ± 1.1	5.6 ± 2.8	14.9 ± 1.4	1.8 ± 0.2	2.0 ± 0.4	5.9 ± 0.3
1.00	RK4	6.0 ± 0.5	9.7 ± 2.4	15.1 ± 2.0	3.1 ± 0.5	3.4 ± 0.2	6.5 ± 0.6
	Dopri5	8.5 ± 1.1	8.1 ± 3.5	14.7 ± 1.4	2.2 ± 0.4	2.0 ± 0.4	6.2 ± 0.4
	Bosh3	8.5 ± 1.1	8.1 ± 3.5	14.7 ± 1.4	2.2 ± 0.4	2.0 ± 0.4	6.2 ± 0.4
	Adaptive Heun	8.5 ± 1.1	8.1 ± 3.5	14.7 ± 1.4	2.2 ± 0.4	2.0 ± 0.4	6.2 ± 0.4
	Midpoint	16.8 ± 0.6	11.0 ± 2.1	$(3.44 \pm 2.17) \times 10^{19}$	2.5 ± 0.1	2.5 ± 0.2	17.0 ± 0.4
	Euler	13.2 ± 0.6	20.0 ± 2.3	$(7.14 \pm 5.10) \times 10^{13}$	2.8 ± 0.3	2.8 ± 0.3	17.0 ± 0.4
	Explicit Adams	6.0 ± 0.5	9.7 ± 2.4	15.1 ± 2.0	3.1 ± 0.5	3.4 ± 0.2	6.5 ± 0.6