LEARNING CONTINUOUS-TIME DYNAMICS GUIDANCE FOR DIFFUSION-BASED SPATIOTEMPORAL RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

We address the ill-posed inverse problem of reconstructing complete spatiotemporal fields from sparse measurements in systems governed by unknown PDEs. Diffusion models have emerged as powerful generative models for such inverse problems. However, conditioning on sparse measurements alone can produce reconstructions that match the observations but remain inconsistent with the underlying dynamics, limiting their reliability in unobserved regions of the domain. We propose **DiffSINDy**, a guided diffusion posterior sampling framework in which a diffusion model provides a prior over full spatiotemporal trajectories, while a learned surrogate provides dynamics guidance without requiring the ground-truth governing equations. We instantiate the surrogate with Sparse Identification of Nonlinear Dynamics (SINDy) and couple it with a differentiable method-of-lines solver, yielding an interpretable continuous-time guidance model that can be integrated between consecutive snapshots even when the data are temporally sparse. We evaluate the method on ODE and PDE benchmarks against guidance based on neural surrogates, two oracle variants with true PDEs, and observation-only DPS. Across benchmarks, SINDy-guided diffusion achieves competitive reconstruction error and improves physical consistency over learned neural surrogates and observation-only guidance.

1 INTRODUCTION

In fluid mechanics and atmospheric science, measurements often cover only a small fraction of an evolving field, while understanding the system requires recovering its behavior across space and time (Shu et al., 2023; Yao et al., 2025). Reconstructing the unobserved states is an inverse problem that can admit several solutions compatible with the same measurements (Huang et al., 2024). Sparse observations alone are therefore insufficient to determine a physically plausible reconstruction, motivating the use of learned structure over admissible fields and their dynamics.

Generative modeling offers a way to learn such prior information from data. In recent years, diffusion models (DMs, Song et al. 2021) and flow matching models (FMs, Lipman et al. 2023) have become prominent frameworks for learning and sampling high-dimensional distributions, with recent work also targeting efficient one-step generation (Geng et al., 2025; Deng et al., 2026). For inverse problems, learned generative priors can be combined with observations to produce conditional reconstructions. In particular, diffusion posterior sampling (DPS) (Chung et al., 2023) combines a learned diffusion prior with approximate observation-likelihood guidance during denoising.

For physical systems, however, observation consistency alone does not ensure that a reconstructed field follows the underlying dynamics. Physics-guided diffusion methods address this issue by incorporating known governing equations into reconstruction (Huang et al., 2024; Jacobsen et al., 2025). Their physics-based corrections therefore require equations that may not be available when the dynamics are known only through observed trajectories. Learned forward surrogates have also been used for likelihood guidance (Lin et al., 2026; Wang & Tartakovsky, 2026). These methods primarily infer unobserved PDE quantities, such as spatially varying coefficients or initial conditions, rather than assess temporal consistency of complete generated state fields under dynamics identified from data.

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

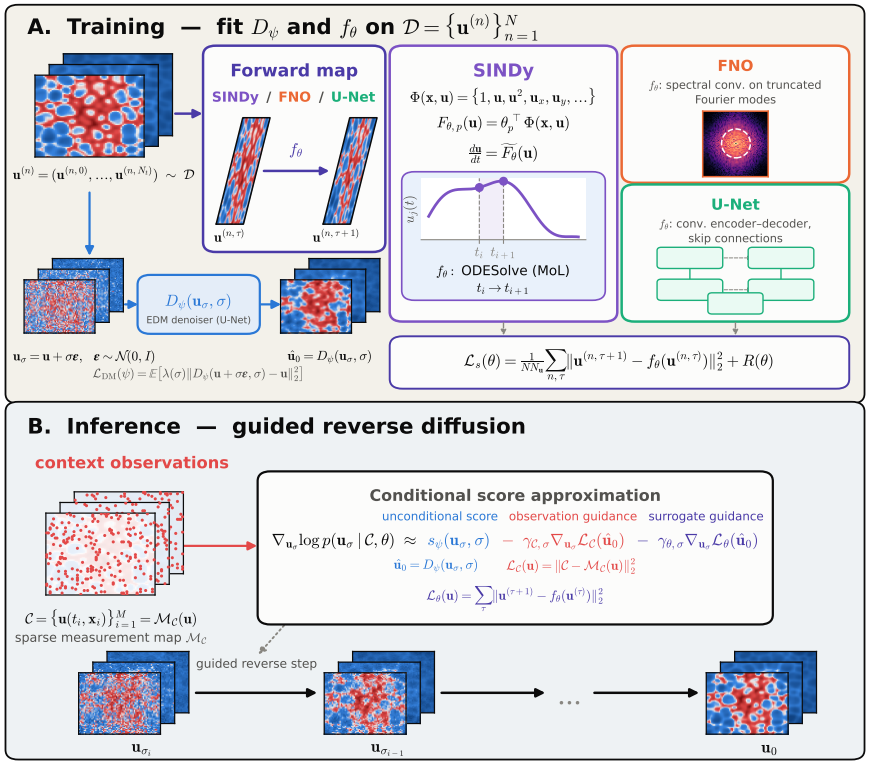


Figure 1: **Illustration of the method. (A) Training.** Clean field samples are used to train both the diffusion prior and a one-step forward surrogate, instantiated as SINDy, FNO, or U-Net; choosing SINDy recovers *DiffSINDy*. **(B) Inference.** We reconstruct a test field from sparse context observations via guided diffusion posterior sampling, correcting the unconditional score with both observation guidance and surrogate guidance at each reverse step.

Motivated by this gap, we consider reconstruction across trajectories that share the same unknown dynamics but differ in their initial conditions. [1] During training, complete trajectories are available to learn both the distribution of plausible solution fields and their common dynamics; at test time, only sparse space-time observations of a new trajectory are given. Formally, let $\boldsymbol{u}(t, \boldsymbol{x}) \in \mathbb{R}^P$ denote a P-component state field on a D-dimensional spatial domain $\Omega \subset \mathbb{R}^D$, governed by

$$\frac{\partial \boldsymbol{u}}{\partial t}(t, \boldsymbol{x}) = \boldsymbol{F}\big(\boldsymbol{u}(t, \boldsymbol{x}), \nabla \boldsymbol{u}(t, \boldsymbol{x}), \nabla^2 \boldsymbol{u}(t, \boldsymbol{x}), \ldots\big), \qquad (\boldsymbol{x}, t) \in \Omega \times [t_0, T]. \tag{1}$$

The governing operator $\boldsymbol{F}$ is shared across trajectories but is not assumed known. We observe a training set of complete trajectories $\mathcal{D} = \{\boldsymbol{u}^{(n)}\}_{n=1}^N$, generated under varying initial conditions and recorded on a common spatiotemporal grid, $\boldsymbol{u}^{(n)} = (\boldsymbol{u}^n(n, 0), \ldots, \boldsymbol{u}^{(n,N_t)})$ at times $t_0, \ldots, t_{N_t}$.

At test time, we observe a sparse set of context values $\mathcal{C} = \{\boldsymbol{u}(t_i), \boldsymbol{x}_i)\}_{i=1}^M$ at space-time locations from a new trajectory. Our goal is to sample a complete spatiotemporal field $\boldsymbol{u}^{(\mathrm{new})}$ that agrees with these observations and remains dynamically plausible. Hence, there is a need for reconstruction guidance that learns unknown dynamics from training trajectories and uses their continuous-time evolution to assess consistency across the reconstructed field (see Figure 1 for an overview).

Contributions. We combine a diffusion prior over complete fields with a learned continuous-time PDE surrogate. Its integration between reconstructed snapshots provides a differentiable dynamical-consistency loss that complements observation guidance during denoising. Our contributions are:

- **Conceptual.** We formulate sparse spatiotemporal reconstruction with unknown governing equations as conditional generation guided by both observations and learned dynamics.

[1] Here, "unknown" refers to the governing equations; boundary conditions are assumed known.

- **Methodological.** We introduce continuous-time physics guidance by combining Sparse Identification of Nonlinear Dynamics (SINDy) (Brunton et al., 2016) with differentiable method-of-lines (MoL) integration of the learned PDE.
- **Empirical.** We compare DiffSINDy against neural surrogates, oracle guidance, and observation-only DPS. On the evaluated systems, learned continuous-time guidance substantially improves physical consistency, even when reconstruction-error improvements are modest.

2 Related work

Physics-guided diffusion. A growing line of work incorporates physical structure into diffusion-based inverse solvers. Diffusion posterior sampling (DPS) (Chung et al., 2023) combines a learned diffusion prior with observation-likelihood guidance during sampling. For PDE-governed fields, DiffusionPDE (Huang et al., 2024) and CoCoGen (Jacobsen et al., 2025) incorporate governing equations into reconstruction, while Shu et al. (2023) develop physics-informed diffusion for flow-field recovery. These approaches assume that the relevant physical equations are available, whereas we learn the dynamics used for guidance from training trajectories.

Function-space and constrained generation. Recent generative methods extend diffusion and flow models to function spaces and explicit constraint enforcement. FunDPS (Yao et al., 2025) formulates posterior guidance in function spaces, while FunDiff (Wang et al., 2026) incorporates physical structure into function-space generative modeling. For flow-based generation, ECI (Cheng et al., 2025) and PCFM (Utkarsh et al., 2025) enforce constraints via corrective sampling, whereas HardFlow (Li et al., 2026) uses trajectory optimization. These methods address discretization or constraint enforcement, but do not identify unknown dynamics for continuous-time reconstruction guidance.

Learned-surrogate guidance. When exact forward models are unavailable or expensive, learned surrogates can provide differentiable guidance for inverse inference. DDIS (Lin et al., 2026) separates the diffusion prior from a learned forward operator, with optional PDE-based regularization during surrogate training. L-DPS (Wang & Tartakovsky, 2026) uses neural forward surrogates for likelihood guidance in latent space. These methods infer unobserved PDE quantities, such as spatially varying coefficients, or initial conditions, from observations of the solution field. FreqNO-DPS (Perrone et al., 2026) guides diffusion posterior sampling with a frozen neural operator for surface wavefield reconstruction and introduces frequency-dependent calibration to mitigate the surrogate's spectral bias. We instead sample complete spatiotemporal state fields and guide their dynamical consistency through continuous-time integration of an identified PDE.

Data-driven discovery of dynamics. A complementary line of work identifies governing dynamics directly from observed trajectories. SINDy (Brunton et al., 2016) identifies sparse governing equations from a candidate library, and PDE-FIND (Rudy et al., 2017) extends this approach to PDEs. WSINDy (Messenger & Bortz, 2021) uses a weak formulation to avoid pointwise derivative estimation, while integration-based sparse identification (Goyal & Benner, 2022) fits dynamics through numerical time stepping. These methods focus on identifying dynamical models. We instead use the identified dynamics as differentiable guidance for conditional field generation.

Table 1 compares the closest approaches. Appendix A discusses additional surrogate and generative methods for inverse problems. For a more comprehensive review of recent machine learning methods for PDE inverse problems, see (Tan et al., 2026; Daras et al., 2024).

3 Background

3.1 Diffusion Models

Diffusion models learn a data distribution by training a denoiser on noise-corrupted samples and using it to progressively transform noise into new samples (Sohl-Dickstein et al., 2015; Song et al., 2021). The EDM formulation (Karras et al., 2022) uses σ to denote the diffusion noise level; here t denotes physical time. For a clean discretized field $\boldsymbol{u} \sim p_{\text{data}}$, additive Gaussian noise defines

$$\boldsymbol{u}_\sigma = \boldsymbol{u} + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2}$$

Table 1: **Feature comparison of related guidance-based inverse-problem methods.** Green ✓ denotes supported features, red ✗ absent features, and orange ● partial support. We mark spatiotemporal reconstruction as partial when a method handles reconstruction for one-dimensional spatiotemporal data but not full trajectories for spatially two-dimensional systems. FunDPS has no physics-guidance term. DDIS can use physics-based regularization during neural operator training, but it is optional.

Method	Learned dynamics guidance	Dynamics unknown	Continuous-time guidance	Interpretable guidance	Spatiotemporal reconstruction
Shu et al. (Shu et al., 2023)	✗	✗	✗	✓	●
CoCoGen (Jacobsen et al., 2025)	✗	✗	✗	✓	●
DiffusionPDE (Huang et al., 2024)	✗	✗	✗	✓	●
NAMM (Feng et al., 2025)	✗	✗	✗	✓	●
FunDPS (Yao et al., 2025)	✗	✓	✗	✗	✗
DDIS (Lin et al., 2026)	✓	●	✗	✗	✗
L-DPS (Wang & Tartakovsky, 2026)	✓	✓	✗	✗	✗
FreqNO-DPS (Perrone et al., 2026)	✓	✓	✗	✗	✓
DiffSINDy	✓	✓	✓	✓	✓

with marginal density p_σ. A neural denoiser $D_\psi(\boldsymbol{u}_\sigma, \sigma)$ is then trained to recover the clean sample using the EDM weighted denoising objective

$$\mathcal{L}_{\mathrm{DM}}(\psi) = \mathbb{E}_{\boldsymbol{u},\sigma,\boldsymbol{\epsilon}} \left[\lambda(\sigma) \, \| D_\psi(\boldsymbol{u} + \sigma\boldsymbol{\epsilon}, \sigma) - \boldsymbol{u} \|_2^2 \right], \tag{3}$$

where $\sigma \sim p_{\mathrm{train}}$ samples a range of training noise scales and $\lambda(\sigma) > 0$ weights their contribution to the objective.

The population-optimal denoiser estimates the conditional mean of the clean field given its noisy version. Tweedie's identity therefore yields the score estimate (Park et al., 2025)

$$\nabla_{\boldsymbol{u}_\sigma} \log p_\sigma(\boldsymbol{u}_\sigma) \approx s_\psi(\boldsymbol{u}_\sigma, \sigma) := \frac{D_\psi(\boldsymbol{u}_\sigma, \sigma) - \boldsymbol{u}_\sigma}{\sigma^2}. \tag{4}$$

The denoiser D_ψ is parameterized by a neural network. Once trained, the score estimate is used to generate samples by following the reverse process from high to low noise levels. In the variance-exploding/EDM formulation, the corresponding deterministic probability-flow ODE can be parameterized directly by σ as

$$\frac{\mathrm{d}\boldsymbol{u}_\sigma}{\mathrm{d}\sigma} = -\sigma \, s_\psi(\boldsymbol{u}_\sigma, \sigma), \tag{5}$$

which is integrated from a high-noise state $\boldsymbol{u}_{\sigma_{\max}} \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ toward $\sigma = 0$.

For conditional generation, posterior guidance modifies this score using information from the observations. By Bayes' rule,

$$\nabla_{\boldsymbol{u}_\sigma} \log p(\boldsymbol{u}_\sigma \mid \mathcal{C}) = \nabla_{\boldsymbol{u}_\sigma} \log p_\sigma(\boldsymbol{u}_\sigma) + \nabla_{\boldsymbol{u}_\sigma} \log p(\mathcal{C} \mid \boldsymbol{u}_\sigma). \tag{6}$$

Diffusion posterior sampling (DPS) (Chung et al., 2023) approximates the second term using the denoised estimate $\hat{\boldsymbol{u}}_0 = D_\psi(\boldsymbol{u}_\sigma, \sigma)$.

3.2 Sparse Identification of Nonlinear Dynamics (SINDy)

SINDy (Brunton et al., 2016) identifies governing equations under the assumption that their dynamics admit a sparse representation in a chosen function library. For a state $\boldsymbol{z}(t) \in \mathbb{R}^d$ and a library $\Phi(\boldsymbol{z}) = (\phi_1(\boldsymbol{z}), \ldots, \phi_K(\boldsymbol{z}))^\top \in \mathbb{R}^K$, each component is modeled as

$$\dot{z}_p(t) \approx \boldsymbol{\theta}_p^\top \Phi(\boldsymbol{z}(t)), \qquad p = 1, \ldots, d, \tag{7}$$

where $\boldsymbol{\theta}_p \in \mathbb{R}^K$ contains only a few nonzero coefficients. Although the library functions may be nonlinear in the state, the model is linear in these coefficients.

Given state samples $\boldsymbol{z}_\ell$ and derivative estimates $\widehat{\dot{z}}_{\ell,p}$, identification becomes a sparse regression problem. One option is ℓ_1-regularized least squares,

$$\widehat{\boldsymbol{\theta}}_p \in \arg\min_{\boldsymbol{\theta}_p \in \mathbb{R}^K} \sum_{\ell=1}^{L} \left(\widehat{\dot{z}}_{\ell,p} - \boldsymbol{\theta}_p^\top \Phi(\boldsymbol{z}_\ell) \right)^2 + \alpha \|\boldsymbol{\theta}_p\|_1, \tag{8}$$

where L is the number of samples and $\alpha \geq 0$ controls the sparsity penalty. Another option is sequentially thresholded least squares (STLSQ) alternates between removing small coefficients and refitting the remaining terms.

For PDEs, the library also includes spatial derivatives and their interactions with the state (Rudy et al., 2017). The chosen library restricts which governing equations can be represented, and advancing the identified dynamics in time requires numerical integration.

4 METHOD

DiffSINDy combines a diffusion prior over full spatiotemporal fields with a learned surrogate of the dynamics. Both components are trained on fully observed trajectories $\mathcal{D}$ and used at inference time to sample a field conditioned on sparse context observations $\mathcal{C}$. The reverse diffusion process is guided by observation consistency and surrogate-based dynamics consistency, as summarized in Figure 1.

4.1 TRAINING

Surrogate model. The surrogate model provides a differentiable measure of dynamical consistency for candidate fields produced during diffusion sampling. We train a surrogate f_θ on the fully observed trajectories $\mathcal{D}$ so that, given a state at time t_i, it predicts the state at a later time t_{i+1}. For discrete-time surrogates this map is represented directly as

$$\boldsymbol{u}^{(n,i+1)} \approx f_\theta\big(\boldsymbol{u}^{(n,i)}\big), \tag{9}$$

whereas continuous-time surrogates define a vector field $\dot{\boldsymbol{u}} = F_\theta(\boldsymbol{u})$ and obtain f_θ by learning F_θ using numerical integration from t_i to t_{i+1}. All surrogates are trained by minimizing one-step prediction error over the training objective

$$\mathcal{L}_s(\theta) = \frac{1}{NN_{\boldsymbol{u}}} \sum_{n=1}^{N} \sum_{i=0}^{N_t-1} \left\| \boldsymbol{u}^{(n,i+1)} - f_\theta(\boldsymbol{u}^{(n,i)}) \right\|_2^2 + R(\theta), \tag{10}$$

where R denotes method-specific regularization, and $N_{\boldsymbol{u}}$ denotes the number of entries in the spatiotemporal field tensor. The loss $\mathcal{L}_\theta$ is also used during inference as the surrogate-based guidance term applied to the denoised field estimate $D_\psi(\boldsymbol{u}_t, t)$. We optimize Equation 10 with minibatch Adam (Kingma & Ba, 2015; Robbins & Monro, 1951).

Our primary model is the continuous-time SINDy surrogate. We represent the PDE as a sparse linear combination of candidate spatial derivative terms

$$F_p(\boldsymbol{u}, \nabla\boldsymbol{u}, \dots) \approx F_{\theta,p}(\boldsymbol{u}) = \boldsymbol{\theta}_p^\top \Phi(\boldsymbol{u}), \tag{11}$$

where $\boldsymbol{\theta}_p \in \mathbb{R}^K$ denotes the learned coefficient vector for the pth PDE component, and $\Phi = (\phi_1, \dots, \phi_K)^\top$ is a library of K candidate functions. We use polynomial libraries comprising

$$\Phi(\boldsymbol{u}) = \left(1, \boldsymbol{u}, \boldsymbol{u}^2 \dots, \boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{u}_{xy}, \dots, \boldsymbol{u}\boldsymbol{u}_x, \boldsymbol{u}\boldsymbol{u}_y, \dots\right)^\top, \tag{12}$$

where subscript denotes the partial derivative. Specifically, the library $\Phi_{q,r}$ is formed by computing all partial derivatives of all vector field components up to order q, and taking all monomials consisting of these features up to total degree r. Explicit construction of the library is given in Appendix C.1.

To obtain a continuous-time surrogate map, we discretize the spatial derivatives in F_Θ using central differences while leaving time continuous. This method-of-lines discretization gives a coupled ODE system on the spatial grid (Schiesser, 1991; Sarmin & Chudov, 1963),

$$\frac{d\boldsymbol{u}}{dt}(t, x_j) = \widetilde{\boldsymbol{F}}_\theta(\boldsymbol{u})_j, \tag{13}$$

where j enumerates all grid points. The one-step surrogate in Equation (9) is then defined by numerical integration from t_i to t_{i+1},

$$f_\theta(\boldsymbol{u}^{(n,i)}) = \mathrm{ODESolve}(\widetilde{\boldsymbol{F}}_\theta, \boldsymbol{u}^{(n,i)}, t_i, t_{i+1}). \tag{14}$$

Direct simulation-based optimization over a large library can be expensive and may produce unstable or ill-posed learned dynamics. We therefore initialize the coefficients using sequentially thresholded least squares (STLSQ) (Brunton et al., 2016). Time derivatives are estimated either by fitting splines to the observed trajectories or by using finite differences, and the sparse coefficients are obtained by regressing these derivative estimates against the library features. After initialization, we start optimizing the coefficients using the continuous-time MoL trajectory loss. Sparsity is encouraged with an ℓ_1 penalty and/or sequential thresholding, with dataset-specific details given in Appendix C.1.

Diffusion model. We learn a diffusion prior D_ψ over complete spatiotemporal solution fields. Each trajectory $\boldsymbol{u} = (\boldsymbol{u}^{(0)}, \ldots, \boldsymbol{u}^{(N_t)})$ is represented as a tensor over physical time, space, and state channels. We reserve t_i for physical time and use σ for the diffusion noise level. Implementation of the diffusion prior follows the EDM formulation. Training is performed using minibatch stochastic optimization with Adam. Further architectural and training details are provided in Appendix C.3.

4.2 Inference

We perform conditional generation using DPS-style guidance with Gaussian approximations to the observation and dynamics-consistency likelihoods (Chung et al., 2023; Shu et al., 2023; Huang et al., 2024; Jacobsen et al., 2025; Wang & Tartakovsky, 2026). By utilizing the Bayes' rule twice, applying Gaussian approximation to both likelihood terms, and plugging in our learned unconditional score s_ψ we obtain an approximation of the conditional score as

$$\nabla_{\boldsymbol{u}_\sigma} \log p(\boldsymbol{u}_\sigma \mid \mathcal{C}, \theta) \approx s_\psi(\boldsymbol{u}_\sigma, \sigma) - \gamma_{\mathcal{C},\sigma} \nabla_{\boldsymbol{u}_\sigma} \mathcal{L}_{\mathcal{C}}(\hat{\boldsymbol{u}}_0) - \gamma_{\theta,\sigma} \nabla_{\boldsymbol{u}_\sigma} \mathcal{L}_\theta(\hat{\boldsymbol{u}}_0). \tag{15}$$

where $\hat{\boldsymbol{u}}_0 = D_\psi(\boldsymbol{u}_\sigma, \sigma)$ is the denoised field estimate, and $\gamma_{\mathcal{C},\sigma}$ and $\gamma_{\theta,\sigma}$ are noise-dependent guidance schedules. The context loss enforces agreement with the sparse observations,

$$\mathcal{L}_{\mathcal{C}}(\boldsymbol{u}) = \|\mathcal{C} - \mathcal{M}_{\mathcal{C}}(\boldsymbol{u})\|_2^2, \tag{16}$$

where $\mathcal{M}_{\mathcal{C}}$ extracts the observed field entries. In our experiments, $\mathcal{M}_{\mathcal{C}}$ is a noiseless sparsity mask. The dynamics loss encourages consistency with the learned one-step map,

$$\mathcal{L}_\theta(\boldsymbol{u}) = \sum_{i=0}^{N_t-1} \left\| \boldsymbol{u}^{(i+1)} - f_\theta(\boldsymbol{u}^{(i)}) \right\|_2^2, \tag{17}$$

where i enumerates the time snapshots of the field $\boldsymbol{u}$. The gradients in equation 15 are taken with respect to the current noisy field $\boldsymbol{u}_\sigma$ by differentiating through the denoiser. For continuous-time SINDy and oracle guidance, this also requires differentiating through the method-of-lines ODE solve. Context guidance is applied throughout the reverse diffusion trajectory, whereas surrogate guidance is activated only for lower noise levels, specifically when $\sigma \leq 10$.

4.3 Optimizing Guidance Weights

The sampler depends on constant guidance strengths $\gamma_{\mathcal{C}}$ and γ_θ, which define the schedules $\gamma_{\mathcal{C},\sigma}$ and $\gamma_{\theta,\sigma}$ in Equation 15. Since overly large values can cause exploding guidance gradients or solver failures, we select them on the validation set using constrained Bayesian optimization (BO) (Garnett, 2023). BO is performed in log-space with $\lambda = (\log \gamma_{\mathcal{C}}, \log \gamma_\theta)$ for surrogate-guided methods, and $\lambda = \log \gamma_{\mathcal{C}}$ for observation-only DPS.

We fit independent Gaussian-process surrogates to both the validation objective $J(\lambda)$ and a feasibility variable $c(\lambda)$, where successful inferences are encoded as $c = -0.5$ and failures as $c = 0.5$. Thus $c(\lambda) < 0$ denotes feasibility. If the constraint posterior at candidate λ_i is $c(\lambda_i) \sim \mathcal{N}(\mu_{c,i}, \sigma_{c,i}^2)$, then

$$p_{\text{feas}}(\lambda_i) = \mathbb{P}[c(\lambda_i) < 0] = \Phi\left(-\frac{\mu_{c,i}}{\sigma_{c,i}}\right). \tag{18}$$

We maximize log expected improvement (LEI) over candidates whose feasibility probability exceeds a threshold:

$$\lambda^\star \in \arg\max_{\lambda:\, p_{\text{feas}}(\lambda) \geq \tau} a_{\text{LEI}}(\lambda). \tag{19}$$

In addition to random sampling, BO is initialized with a grid search in log-space; implementation details and search ranges are given in the Appendix C.3.

Table 2: **Summary of benchmark systems.** Full data-generation details are given in Appendix B.

Dataset	Type	Domain retained	Boundary/IC summary
Lorenz63	ODE	$t \in [1, 2]$	Gaussian initial conditions
Burgers	1D PDE	$(x, t) \in [0, 1]^2$	Dirichlet; two Gaussian pulses
KS	2D PDE	$x \in [0, 64]^2,\ t \in [40, 50]$	chaotic transient discarded
RD	2D PDE	$x \in [0, 1]^2,\ t \in [1, 11]$	periodic; GRF initial condition

5 Experiments

Experimental setting. We evaluate sparse spatiotemporal reconstruction to assess how learned continuous-time guidance affects reconstruction accuracy and physical consistency. The benchmarks comprise Lorenz63, Burgers' equation, the 2D Kuramoto–Sivashinsky equation, and a reaction–diffusion system. For each system, we generate 1000 training trajectories and 100 trajectories for both validation and testing, using randomized initial conditions. Each trajectory contains 16 recorded snapshots. At test time, for each trajectory in the Burgers', 2D Kuramoto-Sivashinsky and the reaction-diffusion datasets we sample 5% of the complete field tensor entries uniformly at random as noiseless context observations $\mathcal{C}$ and reconstruct the complete field. For Lorenz63, the given context is 15%. Table 2 summarizes the systems, and Appendix B specifies their equations, initial and boundary conditions, and time intervals. Evaluation metric specifications can be found in Appendix C.5.

Baselines. We compare DiffSINDy with observation-only DPS (Chung et al., 2023) and discrete-time surrogate guidance using a Fourier Neural Operator (FNO) (Li et al., 2021) or U-Net (Ronneberger et al., 2015). We also introduce two oracle baselines to separate errors in the learned dynamics from differences in the guidance mechanism. Oracle-C integrates the ground-truth dynamics with the same MoL guidance as SINDy. Oracle-D uses a finite-difference residual of the governing equation. All dynamics-guided variants include observation guidance. For Lorenz63, a fully connected network replaces U-Net and FNO is omitted. See Appendix C.2 for surrogate and oracle constructions.

Implementation details. The datasets are generated by using method-of-lines with an adaptive Dormand–Prince 8/7 solver in `diffrax`, with Lorenz63 system reducing to a standard 3-component ODE solve (Prince & Dormand, 1981; Kidger, 2021).

All models are implemented in JAX, supported by its ecosystem of libraries (Bradbury et al., 2018; Gaffney et al., 2026; Heek et al., 2026). We implement the continuous-time MoL-solve using differentiable explicit PID-controlled ODE solvers from `diffrax`, which allows gradients of Equation 10 to be backpropagated through the solver. The initialization of coefficients in the SINDy method is performed with `pysindy` (de Silva et al., 2020; Kaptanoglu et al., 2022).

Diffusion sampling uses the EDM formulation with Heun's method, with one diffusion model trained once per dataset. Each trainable surrogate is fitted with five random seeds, and for each fit, guidance hyperparameters are selected by constrained BO on the validation set, implemented in `BoTorch`, followed by test evaluation with five diffusion noise initializations (Balandat et al., 2020). This gives 25 training–sampling combinations per trainable surrogate, capturing variability from both training and diffusion sampling. Oracle and observation-only baselines omit the surrogate-training step. Solver specifications are given in Appendix B, surrogate architectures and training settings in Appendix C, and diffusion training, sampling, and guidance-hyperparameter selection in Appendix C.3.

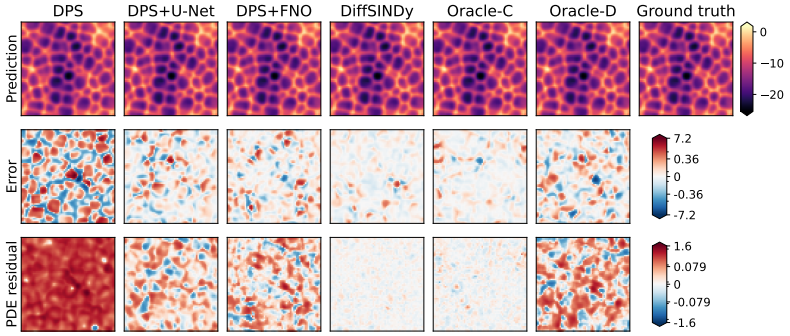


Figure 2: **Qualitative results, Kuramoto-Sivashinsky.** The top row shows model-generated snapshots, the second row their ground-truth residuals, and the bottom PDE residuals with respect to the ground-truth MoL solver. Qualitative results for other datasets are included in Appendix D.

5.1 RESULTS

Reconstruction accuracy. Figure 3 (first row) reports the RRMSE of each method computed against the ground-truth fields across the benchmark datasets. Apart from the continuous-time oracle, SINDy is the strongest method overall. However, the differences in RRMSE are modest across methods, suggesting that this metric is largely determined by the quality of the diffusion prior rather than by the choice of guidance surrogate. This is consistent with the surrogate test errors in Figure 3 (third row): although SINDy reaches near solver-precision prediction errors on several datasets, these surrogate-error differences do not always translate into large RRMSE gains.

Physical consistency. Guidance has a clearer effect on physical consistency. Since the data are temporally sparse, finite-difference residuals are unreliable, so we measure physical consistency using the ground-truth PDE residual: $\boldsymbol{r}_{\mathrm{PDE}}^{(i)} = f_{\mathrm{PDE}}(\hat{\boldsymbol{u}}_0^{(\mathrm{new},i)}) - \hat{\boldsymbol{u}}_0^{(\mathrm{new},i+1)}$, and $\mathcal{L}_{\mathrm{PDE}}(\boldsymbol{u}_0^{(\mathrm{new})}) = \|\boldsymbol{r}\|_2^2$, where f_{PDE} is the MoL-timestep using the ground truth dynamics F, and $\boldsymbol{u}_0^{(\mathrm{new})}$ is the generated sample. DiffSINDy substantially reduces this PDE inconsistency relative to observation-only DPS and neural-surrogate guidance (Figure 3, second row). Thus, in our setting, the main benefit of learned continuous-time guidance is not necessarily a lower pointwise reconstruction error, but improved dynamical consistency of the generated fields. Figure 2 provides a qualitative view of the Kuramoto–Sivashinsky reconstructions, showing visibly reduced PDE residuals under the proposed guidance.

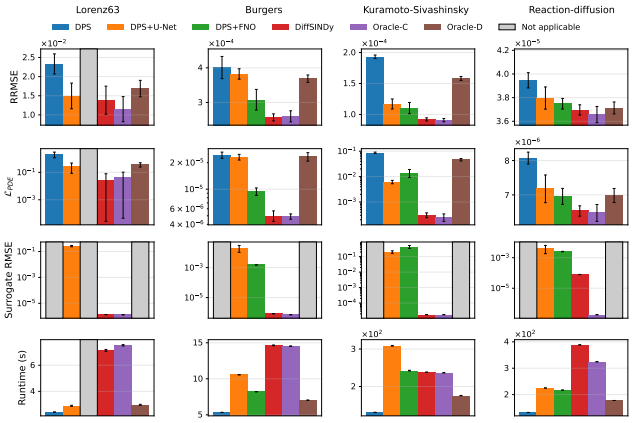


Figure 3: Main results across benchmark systems.

Runtime. Continuous-time guidance is more costly because it requires surrogate simulation and solver differentiation during sampling. DiffSINDy and Oracle-C are therefore generally slower than DPS, Oracle-D, FNO, and U-Net (Figure 3, last row).

5.2 ABLATION STUDIES

Next, we assess the sensitivity of the method to temporal and observation sparsity, as well as to the choice of SINDy feature library.

Temporal resolution. We ablate temporal resolution by reducing the number of saved snapshots per trajectory to 12 and 8 while keeping the physical time interval fixed. As shown in Figure 4, lower temporal resolution increases RRMSE across all methods, including the oracle baselines, and causes

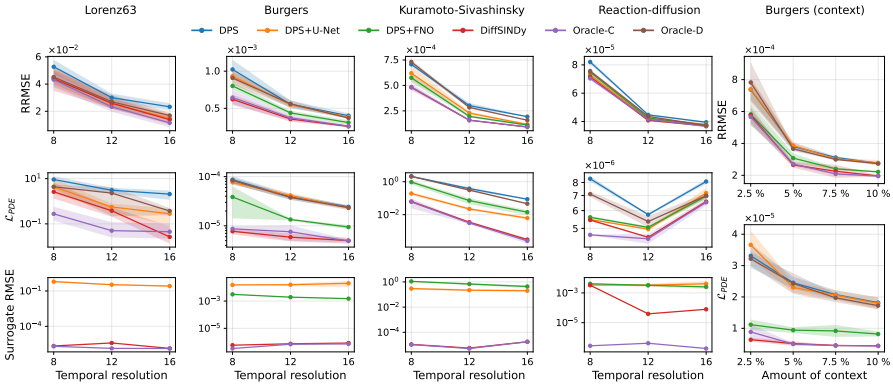


Figure 4: **Ablation studies.** Columns 1–4: temporal-resolution ablation, using fewer saved snapshots over a fixed time interval. Column 5: context-sparsity ablation on Burgers' equation, showing RRMSE and physical consistency as the amount of context available during sampling varies.

the methods to move closer to the observation-only DPS baseline. This suggests that, under sparse temporal sampling, reconstruction error is increasingly dominated by the diffusion prior. In contrast, physical consistency is largely preserved by the continuous-time oracle and SINDy-guided methods. This indicates that continuous-time guidance can still enforce dynamically plausible reconstructions when the observed trajectory is temporally sparse.

Context points. We vary the number of points given as context $\mathcal{C}$ during sampling (Figure 4, column 5). Reducing the amount of observed context consistently increases the reconstruction error across all methods, with broadly similar trends and no qualitative change in their relative behavior. Physical consistency shows more method-dependent behavior: the continuous-time guided methods remain at low $\mathcal{L}_{\mathrm{PDE}}$ across context levels, with FNO also remaining comparatively stable, while observation-only DPS and U-Net deteriorate more as context becomes sparse.

SINDy library. We denote the main SINDy library by $\Phi^{(1)}_{q,r}$ and consider a more restricted alternative, $\Phi^{(2)}_{q,r}$, comprising products of spatial derivatives up to order q with monomials comprising the components of the undifferentiated field up to degree r (Appendix C.1). We vary the library construction and its size across Burgers' dataset. For each configuration, SINDy hyperparameters are selected by BO on a shortened training trajectory. Since this provides only a heuristic proxy for full training, we repeat the BO procedure five times independently and subsequently train the corresponding models on the full trajectories. Figure 5 reports the mean and standard deviation of the Burgers' surrogate RMSE together with the recovered coefficient residuals. Across most library choices, SINDy achieves very low surrogate RMSE, although some configurations of the more restricted $\Phi^{(2)}$ library converge to unstable parameter regions. Recovered coefficients are generally close to the ground truth on the true support, while larger libraries can retain small spurious terms, showing that accuracy in the support of the test data does not imply exact structural recovery.

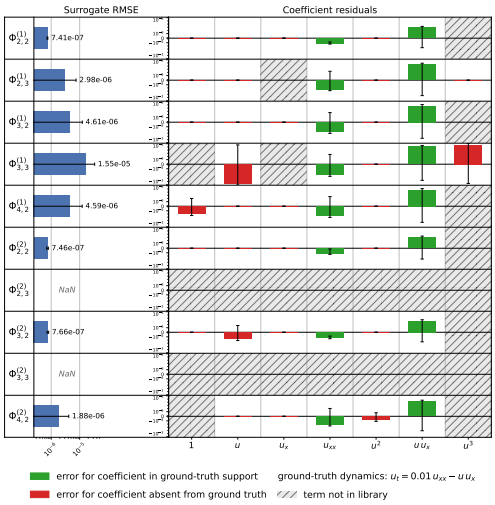


Figure 5: **SINDy library ablation.** Mean surrogate error and coefficient recovery across different SINDy library choices. Error margins are plotted over the configurations identified by five Bayesian optimization searches.

6 Conclusion

Sparse spatiotemporal reconstruction requires recovering complete trajectories from limited observations while preserving consistency with the underlying dynamics. We introduced DiffSINDy, a diffusion-based framework that augments observation guidance with dynamics learned directly from training trajectories. Our main instantiation combines SINDy with differentiable method-of-lines integration, providing interpretable continuous-time guidance without access to the ground-truth governing equations. Across the evaluated ODE and PDE benchmarks, SINDy-guided diffusion achieves competitive reconstruction accuracy while substantially improving physical consistency over observation-only DPS and learned discrete-time surrogates. These gains in dynamical consistency are largely preserved as the temporal resolution is reduced, highlighting the benefit of continuous-time guidance for temporally sparse reconstruction.

Limitations and future work. Several current design choices suggest natural avenues for further work. As shown by our ablation study, scaling SINDy to larger and more general candidate libraries remains challenging, as optimization can become poorly conditioned when candidate terms are strongly correlated. More adaptive libraries or uncertainty-aware dynamics discovery could reduce this dependence on library design. Results can also be sensitive to the hyperparameters selected by BO, both when fitting the dynamics model and when tuning guidance strengths, motivating more robust or less tuning-intensive procedures. Finally, continuous-time guidance requires repeatedly integrating the learned dynamics and differentiating through the numerical solver, increasing inference cost and occasionally causing numerical difficulties. Cheaper and more stable approximations to this guidance are therefore a promising direction.

AI USE STATEMENT

In this work, we used generative AI tools for providing feedback on research methodology. We have not used generative AI tools for generating synthetic datasets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, designing methodology or experiments, proposing or refining hypotheses, implementing methods, supporting qualitative and thematic data analysis and interpreting results. Providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, cleaning and reformatting datasets and assisting with translation are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures or images, editing and creating software code, drafting parts of a research paper and editing a research paper for readability. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the main algorithmic components in Section 4 and the experimental protocol in Section 5. The appendix contains the implementation details needed to reproduce the reported experiments: dataset-generation procedures, equations, initial and boundary conditions, solver tolerances, and retained time intervals are given in Appendix B; surrogate architectures, SINDy libraries, training hyperparameters, and oracle baselines are described in Appendix C; and diffusion-model training, sampling schedules, and guidance-weight optimization are specified in Appendix C.3. The code is published upon acceptance.

REFERENCES

Gabriel Achour, Woong Je Sung, Olivia J Pinon-Fischer, and Dimitri N Mavris. Development of a conditional generative adversarial network for airfoil shape optimization. In *AIAA Scitech 2020 Forum*, Reston, Virginia, January 2020. American Institute of Aeronautics and Astronautics.

Giovanni S. Alberti, Johannes Hertrich, Matteo Santacesaria, and Silvia Sciutto. Manifold learning by mixture models of vaes for inverse problems. *Journal of Machine Learning Research*, 25(202): 1–35, 2024. URL http://jmlr.org/papers/v25/23-0396.html.

Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing Inverse Problems with Invertible Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://dblp.org/rec/conf/iclr/ArdizzoneKRK19.html.

M. Baer. findiff software package, 2018. URL https://github.com/maroba/findiff. https://github.com/maroba/findiff.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

Gang Bao and Yaohua Zang. A unified physics-informed generative operator framework for general inverse problems, 2025. URL https://arxiv.org/abs/2511.03241.

Harry Bateman. Some recent researches on the motion of fluids. *Mon. Weather Rev.*, 43(4):163–170, apr 1915.

Jens Behrmann, Will Grathwohl, Ricky T Q Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible Residual Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pp. 573–582. PMLR, 2019. URL https://proceedings.mlr.press/v97/behrmann19a.html.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Andrew Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1059–1071. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/brock21a.html.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 3 2016. ISSN 1091-6490. doi: 10.1073/pnas.1517384113.

Chaoran Cheng, Boran Han, Danielle C Maddix, Abdul Fatir Ansari, Andrew Stuart, Michael W Mahoney, and Yuyang Wang. Gradient-Free Generation for Hard-Constrained Systems. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f9d7d6c695bc983fcfb5b70a5fbdfd2f-Abstract-Conference.html.

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 (ICLR)*, 2023. URL https://iclr.cc/virtual/2023/poster/11877.

Frank Cole, Yulong Lu, Wuzhe Xu, and Tianhao Zhang. In-Context Learning of Linear Systems: Generalization Theory and Applications to Operator Learning, 2025. URL https://arxiv.org/abs/2409.12293.

Dario Coscia, Nicola Demo, and Gianluigi Rozza. Generative Adversarial Reduced Order Modelling. *Scientific Reports*, 14:3826, 2024. doi: 10.1038/s41598-024-54067-z. URL https://www.nature.com/articles/s41598-024-54067-z.

Giannis Daras, Hyungjin Chung, Chieh-Hsin Lai, Yuki Mitsufuji, Jong Chul Ye, Peyman Milanfar, Alexandros G Dimakis, and Mauricio Delbracio. A survey on diffusion models for inverse problems, 2024. URL https://arxiv.org/abs/2410.00083.

Brian de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J Kutz, and Steven Brunton. PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104, 2020. doi: 10.21105/joss.02104. URL https://doi.org/10.21105/joss.02104.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative Modeling via Drifting, 2026. URL https://arxiv.org/abs/2602.04770.

Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 8780–8794, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

Berthy Feng, Ricardo Baptista, and Katherine Bouman. Neural approximate mirror maps for constrained diffusion models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 55155–55178, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/89b0e466b46292ce0bfe53618aadd3de-Paper-Conference.pdf.

Colin Gaffney, Shutong Li, Daniel Ng, Anastasia Petrushkina, Niket Kumar, Adam Cogdell, Mridul Sahu, Yaning Liang, Nikhil Bansal, Justin Pan, Angel Mau, Abhishek Agrawal, Marco Berlot, Ruoxin Sang, Kiranbir Sodhia, and Rakesh Iyer. Orbax: Distributed Checkpointing with JAX, 2026. URL https://arxiv.org/abs/2605.23066.

Han Gao, Matthew J. Zahr, and Jian-Xun Wang. Physics-informed graph neural galerkin networks: A unified framework for solving pde-governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 390:114502, February 2022. ISSN 0045-7825. doi: 10.1016/j.cma.2021.114502. URL http://dx.doi.org/10.1016/j.cma.2021.114502.

Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems*, 2025.

Pawan Goyal and Peter Benner. Discovery of nonlinear dynamical systems using a Runge–Kutta inspired dictionary-based sparse regression approach. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478(2262):20210883, 2022. doi: 10.1098/rspa.2021.0883.

Jan Hagnberger, Marimuthu Kalimuthu, Daniel Musekamp, and Mathias Niepert. Vectorized Conditional Neural Fields: A Framework for Solving Time-dependent Parametric Partial Differential Equations. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, 2024.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2026. URL http://github.com/google/flax.

Amin Heyrani Nobari, Wei Chen, and Faez Ahmed. Pcdgan: A continuous conditional diverse generative adversarial network for inverse design. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 606–616. ACM, August 2021. doi: 10.1145/3447548.3467414. URL http://dx.doi.org/10.1145/3447548.3467414.

Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. DiffusionPDE: Generative PDE-Solving Under Partial Observation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb3878c1dcbfff9ee95d5d033e5f5942-Abstract-Conference.html.

Christian Jacobsen, Yilin Zhuang, and Karthik Duraisamy. CoCoGen: Physically-Consistent and Conditioned Score-based Generative Models for Forward and Inverse Problems. *SIAM Journal on Scientific Computing*, 47(2):C399–C425, 2025. doi: 10.1137/24M1636071. URL https://epubs.siam.org/doi/10.1137/24M1636071.

Anran Jiao, Qile Yan, John Harlim, and Lu Lu. Solving forward and inverse PDE problems on unknown manifolds via physics-informed neural operators. *SIAM Journal on Scientific Computing*, 48(1):C136–C163, 2026. doi: 10.1137/24M1675254. URL https://doi.org/10.1137/24M1675254.

Marimuthu Kalimuthu, David Holzmüller, and Mathias Niepert. LOGLO-FNO: Efficient learning of local and global features in fourier neural operators. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=MQ1dRdHTpi. J2C Certification.

Alan A Kaptanoglu, Brian M de Silva, Urban Fasel, Kadierdan Kaheman, Andy J Goldschmidt, Jared Callaham, Charles B Delahunt, Zachary G Nicolaou, Kathleen Champion, Jean-Christophe Loiseau, J Nathan Kutz, and Steven L Brunton. PySINDy: A comprehensive Python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994, 2022. doi: 10.21105/joss.03994. URL https://doi.org/10.21105/joss.03994.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/a98846e9d9cc01cfb87eb694d946ce6b-Abstract-Conference.html.

Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Jakob Kruse. Technical report: Training Mixture Density Networks with full covariance matrices, 2020. URL https://arxiv.org/abs/2003.05739.

Jakob Kruse, Lynton Ardizzone, Carsten Rother, and Ullrich Köthe. Benchmarking Invertible Architectures on Inverse Problems, 2021. URL https://arxiv.org/abs/2101.10763.

Yoshiki Kuramoto. Diffusion-Induced chaos in reaction systems. *Progr. Theoret. Phys. Suppl.*, 64: 346–367, 1978.

William Lauga, James Rowbottom, Alexander Denker, Željko Kereta, Moshe Eliasof, and Carola-Bibiane Schönlieb. Graph Neural Regularizers for PDE Inverse Problems, 2025. URL https://arxiv.org/abs/2510.21012.

Zeyang Li, Kaveh Alim, and Navid Azizan. HardFlow: Hard-Constrained Sampling for Flow-Matching Models via Trajectory Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. URL https://ieeexplore.ieee.org/abstract/document/11592684.

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. URL https://iclr.cc/virtual/2021/poster/3281.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023. URL http://jmlr.org/papers/v24/23-0064.html.

Thomas Y. L. Lin, Jiachen Yao, Lufang Chiang, Julius Berner, and Anima Anandkumar. Decoupled Diffusion Sampling for Inverse Problems on Function Spaces, 2026. URL https://arxiv.org/abs/2601.23280.

Yaron Lipman, Ricky T Q Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://iclr.cc/virtual/2023/poster/11309.

Zhaocheng Liu, Dayu Zhu, Sean Rodrigues, Kyutae Lee, and Wenshan Cai. A generative model for the inverse design of metasurfaces. *Nano Letters*, 18, 09 2018. doi: 10.1021/acs.nanolett.8b03171.

Da Long, Zhitong Xu, Qiwei Yuan, Yin Yang, and Shandian Zhe. Invertible Fourier Neural Operators for Tackling Both Forward and Inverse Problems. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 258. PMLR, 2025. URL https://proceedings.mlr.press/v258/long25a.html.

Edward N Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sci.*, 20(2):130–141, mar 1963.

Cooper Lorsung, Zijie Li, and Amir Barati Farimani. Physics informed token transformer for solving partial differential equations. *Machine Learning: Science and Technology*, 5(1):015032, feb 2024. doi: 10.1088/2632-2153/ad27e3. URL https://doi.org/10.1088/2632-2153/ad27e3.

Daniel A. Messenger and David M. Bortz. Weak SINDy for partial differential equations. *Journal of Computational Physics*, 443:110525, 2021. doi: 10.1016/j.jcp.2021.110525.

Roberto Molinaro, Yunan Yang, Björn Engquist, and Siddhartha Mishra. Neural Inverse Operators for Solving PDE Inverse Problems. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 25105–25139. PMLR, 2023. URL https://proceedings.mlr.press/v202/molinaro23a.html.

Daniel Musekamp, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert. Active learning for neural PDE solvers. -, 2025. URL https://openreview.net/forum?id=x4ZmQaumRg.

Oded Ovadia, Adar Kahana, Panos Stinis, Eli Turkel, Dan Givoli, and George Em Karniadakis. Vito: Vision transformer-operator. *Computer Methods in Applied Mechanics and Engineering*, 428:117109, 2024. ISSN 0045-7825. doi: 10.1016/j.cma.2024.117109. URL https://www.sciencedirect.com/science/article/pii/S0045782524003657.

Chicago Y Park, Michael T McCann, Cristina Garcia-Cardona, Brendt Wohlberg, and Ulugbek S Kamilov. Random Walks with Tweedie: A Unified View of Score-Based Diffusion Models. *IEEE Signal Processing Magazine*, 42(3), 2025. doi: 10.1109/MSP.2025.3590608.

Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.*, 120(2):024102, jan 2018.

Niccolò Perrone, Fanny Lehmann, Stefania Fresca, and Filippo Gatti. Correcting Neural Operator Spectral Bias via Diffusion Posterior Sampling with Sparse Observations, 2026. URL https://arxiv.org/abs/2606.03936.

P J Prince and J R Dormand. High order embedded Runge–Kutta formulae. *J. Comp. Appl. Math*, 7 (1):67–75, 1981.

Md Ashiqur Rahman, Manuel A Florez, Anima Anandkumar, Zachary E Ross, and Kamyar Azizzadenesheli. Generative Adversarial Neural Operators. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=X1VzbBU6xR.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, Sept 1951. ISSN 0003-4851. doi: 10.1214/aoms/1177729586. URL http://dx.doi.org/10.1214/aoms/1177729586.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)*, volume 9351 of *Lecture Notes in Computer Science*, pp. 234–241, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28.

Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017. doi: 10.1126/sciadv.1602614. URL https://www.science.org/doi/10.1126/sciadv.1602614.

E.N. Sarmin and L.A. Chudov. On the stability of the numerical integration of systems of ordinary differential equations arising in the use of the straight line method. *USSR Computational Mathematics and Mathematical Physics*, 3(6):1537–1543, 1963. ISSN 0041-5553. doi: 10.1016/0041-5553(63)90256-8. URL https://www.sciencedirect.com/science/article/pii/0041555363902568.

William E. Schiesser. *The Numerical Method of Lines*. Academic Press, 1991.

Shubhanshu Shekhar, Mohammad Javad Khojasteh, Ananya Acharya, Tony Tohme, and Kamal Youcef-Toumi. VINA: Variational Invertible Neural Architectures, 2026. URL https://arxiv.org/abs/2602.20480.

Dule Shu, Zijie Li, and Amir Barati Farimani. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972, 4 2023. ISSN 0021-9991. doi: 10.1016/j.jcp.2023.111972. URL http://dx.doi.org/10.1016/j.jcp.2023.111972.

G I Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations. *Acta Astronaut.*, 4(11-12):1177–1206, nov 1977.

I M Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. ISSN 0041-5553. doi: 10.1016/0041-5553(67)90144-9. URL https://www.sciencedirect.com/science/article/pii/0041555367901449.

Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37, pp. 2256–2265. PMLR, 2015. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An Extensive Benchmark for Scientific Machine Learning. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks*, 2022a. URL https://arxiv.org/abs/2210.07182.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench Datasets, 2022b. URL https://doi.org/10.18419/darus-2986.

Makoto Takamoto, Francesco Alesiani, and Mathias Niepert. Learning Neural PDE Solvers with Parameter-Guided Channel Attention. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 33448–33467. PMLR, 2023. URL https://proceedings.mlr.press/v202/takamoto23a.html.

Zhentao Tan, Yuze Hao, Boyi Zou, Mingsheng Long, Yi Yang, and Gang Bao. Harnessing AI for Inverse Partial Differential Equation Problems: Past, Present, and Prospects, 2026. URL https://arxiv.org/abs/2605.16966.

Yunfei Teng, Anna Choromanska, and Mariusz Bojarski. Invertible Autoencoder for domain adaptation. *Computation*, 7(2):20, 2019. doi: 10.3390/computation7020020. URL https://www.mdpi.com/2079-3197/7/2/20.

Utkarsh Utkarsh, Pengfei Cai, Alan Edelman, Rafael Gomez-Bombarelli, and Christopher Vincent Rackauckas. Physics-Constrained Flow Matching: Sampling Generative Models with Hard Constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://neurips.cc/virtual/2025/poster/117071.

Liwei Wang, Yu-Chin Chan, Faez Ahmed, Zhao Liu, Ping Zhu, and Wei Chen. Deep generative modeling for mechanistic-based learning and design of metamaterial systems. *Computer Methods in Applied Mechanics and Engineering*, 372:113377, December 2020. ISSN 0045-7825. doi: 10.1016/j.cma.2020.113377. URL http://dx.doi.org/10.1016/j.cma.2020.113377.

Sifan Wang, Zehao Dou, Siming Shan, Tong-Rui Liu, and Lu Lu. FunDiff: Diffusion Models over Function Spaces for Physics-Informed Generative Modeling. *Nature Communications*, 17:5749, 2026. doi: 10.1038/s41467-026-72292-0. URL https://doi.org/10.1038/s41467-026-72292-0.

Tian Wang and Chuang Wang. Latent Neural Operator for Solving Forward and Inverse PDE Problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/39f6d5c2e310a5a629dcfc4d517aa0d1-Abstract-Conference.html.

Yuanzhe Wang and Alexandre M. Tartakovsky. Latent Diffusion Posterior Sampling with Surrogate Likelihood Guidance for PDE Inverse Problems, 2026. URL https://arxiv.org/abs/2606.26592.

Long Wei, Peiyan Hu, Ruiqi Feng, Yixuan Du, Tao Zhang, Rui Wang, Yue Wang, Zhi-Ming Ma, and Tailin Wu. Generative PDE Control. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, 2024. URL https://openreview.net/forum?id=vaKnCahjdj.

Jiachen Yao, Abbas Mammadov, Julius Berner, Gavin Kerrigan, Jong Chul Ye, Kamyar Azizzadenesheli, and Anima Anandkumar. Guided Diffusion Sampling on Function Spaces with Applications to PDEs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://neurips.cc/virtual/2025/poster/116070.

Jiacheng Yuan, Lei Zeng, Youan Shi, Yewei Gui, Bin Xia, and Dong Wei. Discretization-informed neural networks for solving inverse heat conduction problems. *International Communications in Heat and Mass Transfer*, 176:111359, 2026. ISSN 0735-1933. doi: 10.1016/j.icheatmasstransfer.

2026.111359. URL https://www.sciencedirect.com/science/article/pii/S0735193326008808.

Qingqing Zhao, David B Lindell, and Gordon Wetzstein. Learning to Solve PDE-constrained Inverse Problems with Graph Networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pp. 26895–26910. PMLR, 2022. URL https://proceedings.mlr.press/v162/zhao22d.html.

Zhiyuan Zhao, Xueying Ding, and B Aditya Prakash. PINNsFormer: A Transformer-Based Framework For Physics-Informed Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/a6f2763089c0bd8f56006c42f09ee24c-Abstract-Conference.html.

Zhenyi Zhu, Yuchen Huang, and Liu Liu. Physicssolver: Transformer-enhanced physics-informed neural networks for forward and forecasting problems in partial differential equations. *Journal of Computational and Applied Mathematics*, 473:116900, 2026. ISSN 0377-0427. doi: 10.1016/j.cam.2025.116900. URL https://www.sciencedirect.com/science/article/pii/S0377042725004145.

The appendix is organized as follows:

- Appendix A expands the related-work discussion on surrogate and generative approaches to PDE inverse problems.
- Appendix B provides the benchmark and data-generation details, including governing equations, initial and boundary conditions, domains, and solver settings.
- Appendix C gives the model and implementation details: Appendix C.1 describes the SINDy libraries and training procedure, Appendix C.2 the learned and oracle baselines, Appendix C.3 the diffusion model, sampling, and guidance optimization, and Appendix C.5 the evaluation metrics.
- Appendix D provides additional qualitative reconstructions and residual visualizations.

A Related Work

This more broad discussion is based on the taxonomy by (Tan et al., 2026) with additional details and citations.

A.1 Surrogate Models for Inverse Problems

Learning-based inverse solvers for PDE-governed systems are often built as surrogates, either by approximating the inverse map directly or by replacing expensive forward PDE evaluations inside an optimization loop. Direct inverse models learn a map from observations to unknown parameters or fields, enabling one-shot inference but often struggling with the non-uniqueness of ill-posed inverse maps. This issue is especially pronounced under sparse measurements, where several physically plausible solutions may explain the same observations. Probabilistic extensions such as invertible neural networks, invertible residual networks, invertible autoencoders, variational invertible models, and mixture-density networks aim to represent this ambiguity more explicitly (Ardizzone et al., 2019; Kruse et al., 2021; Shekhar et al., 2026; Behrmann et al., 2019; Teng et al., 2019; Kruse, 2020). Operator-learning approaches extend direct inversion to function spaces: Neural Inverse Operators combine DeepONet- and FNO-style components, iFNO uses invertible Fourier blocks with latent variables, and latent neural operators perform inversion in low-dimensional representations under sparse observations (Molinaro et al., 2023; Long et al., 2025; Wang & Wang, 2024; Jiao et al., 2026; Bao & Zang, 2025). A complementary line of work uses neural networks as differentiable forward surrogates for inverse optimization. Learned-model-assisted methods replace costly PDE solves with neural operators or other learned simulators, reducing the cost of inversion. Graph neural networks, transformers, and neural-operator architectures have further been used to handle irregular meshes, sparse observations, general geometries, and long-range spatiotemporal dependencies (Zhao et al., 2022; Gao et al., 2022; Yuan et al., 2026; Lauga et al., 2025; Cole et al., 2025; Zhu et al., 2026; Zhao et al., 2024; Ovadia et al., 2024; Lorsung et al., 2024; Li et al., 2023).

A.2 Generative Models for Inverse Problems

Generative models formulate inverse problems as conditional generation. Given observations and, when available, governing physical information, the goal is to sample unknown parameters, states, or fields from a conditional distribution rather than predict a single reconstruction. This is well suited to sparse-measurement inverse problems, where the measurement operator is many-to-one and the posterior may be broad or multimodal. More generally, deterministic point estimates can be insufficient because uncertainty arises from measurement noise, sparse or indirect observations, model-form mismatch, surrogate error, and distribution shift.

Alongside diffusion- and flow-based methods, earlier generative approaches include GAN-based inverse models, which learn a solution manifold and enable fast conditional generation after training (Rahman et al., 2022; Coscia et al., 2024; Liu et al., 2018; Achour et al., 2020; Heyrani Nobari et al., 2021; Wang et al., 2020). Despite their efficiency, however, GANs are often limited by training instability and mode collapse. VAE-based and latent-variable methods instead learn compact latent representations that support conditional sampling, latent-space optimization, and uncertainty-aware search (Alberti et al., 2024). However, their performance relies on the physical validity of the learned latent space.

B DATASET SPECIFICATIONS

Each PDE dataset is generated using the method of lines: spatial derivatives are discretized with second-order accurate central differences, with finite-difference stencils obtained from `findiff` (Baer, 2018). The resulting ODE system is solved with an adaptive Dormand–Prince 8/7 solver with PID-controlled time stepping implemented in `diffrax`. Periodic boundary conditions are implemented with circular padding, whereas Dirichlet boundary conditions are imposed by enforcing zero time derivatives at the boundary. Solver tolerances and implementation details are reported in Table S3.

This setup matches the experimental data-generation protocol: for each system, we generate 1000 training trajectories and 100 trajectories for both validation and testing. The main experiments use 16 recorded snapshots per trajectory, and the temporal-resolution ablations regenerate the data with 12 and 8 snapshots over the same physical time intervals.

Lorenz63 system. Lorenz63 is used as an ODE benchmark. Since the dynamics are finite-dimensional, no method-of-lines discretization is required. We use the standard Lorenz63 system

$$\dot{x} = \sigma_L(y - x), \tag{S1}$$
$$\dot{y} = x(\rho - z) - y, \tag{S2}$$
$$\dot{z} = xy - \beta z, \tag{S3}$$

with $\sigma_L = 10$, $\rho = 28$, and $\beta = 8/3$ (Lorenz, 1963). Initial conditions are sampled as

$$x(0), y(0), z(0) \sim \mathcal{N}(0, 10^2). \tag{S4}$$

Trajectories are simulated on $t \in [0, 2]$, and the first time unit is discarded; the retained data therefore corresponds to $t \in [1, 2]$.

Burgers' equation. We use the one-dimensional viscous Burgers' equation (Bateman, 1915),

$$u_t = -uu_x + \nu u_{xx}, \tag{S5}$$

with viscosity $\nu = 0.01$. The equation is simulated on

$$x \in [0, 1], \quad t \in [0, 1], \tag{S6}$$

with Dirichlet boundary conditions. Initial conditions are sampled as a superposition of two Gaussian pulses,

$$u_0(x) = a_0 \exp\left[-\frac{(x - \mu_0)^2}{2\sigma_0^2}\right] - a_1 \exp\left[-\frac{(x - (\mu_1 + 0.4))^2}{2\sigma_1^2}\right], \qquad x \in \Omega, \tag{S7}$$
$$a_0, a_1 \sim \mathcal{U}(0, 2), \tag{S8}$$
$$\mu_0, \mu_1 \sim \mathcal{U}(0.2, 0.4), \tag{S9}$$
$$\sigma_0, \sigma_1 \sim \mathcal{U}(0.05, 0.15), \tag{S10}$$

where $\mathcal{U}$ denotes the uniform distribution (Wei et al., 2024). The spatial resolution of the data is 128.

Kuramoto–Sivashinsky equation. We use the two-dimensional Kuramoto–Sivashinsky (KS) equation, a standard model for flame-front propagation and spatiotemporal chaos (Kuramoto, 1978; Sivashinsky, 1977; Pathak et al., 2018). The dynamics are

$$u_t = -\nabla^2 u - \nabla^4 u - \frac{1}{2}\|\nabla u\|^2, \tag{S11}$$

where ∇^2 is the Laplacian, and ∇^4 is the biharmonic operator. The system is simulated on

$$x \in [0, 64] \times [0, 64], \qquad t \in [0, 50] \tag{S12}$$

with periodic boundary conditions. We discard the first 40 time units and retain $t \in [40, 50]$ as data. The spatial resolution of the retained trajectories is 64x64. The retained trajectories exhibit spatiotemporal chaos.

Reaction–diffusion system. We use a two-component reaction–diffusion system following the PDEBench benchmark (Takamoto et al., 2022a;b; 2023; Hagnberger et al., 2024; Musekamp et al., 2025; Kalimuthu et al., 2025). The dynamics are

$$u_t = D_u \nabla^2 u + u(1 - u^2) - v - k, \tag{S13}$$

$$v_t = D_v \nabla^2 v + u - v, \tag{S14}$$

with coefficients

$$D_u = 0.001, \qquad D_v = 0.005, \qquad k = 0.005. \tag{S15}$$

We use periodic boundary conditions and initialize the system with a Gaussian random field, $u(x, 0), v(x, 0) \sim \mathcal{N}(0, 1)$. Contrary to the original PDEBench setting, we simulate on

$$x \in [0, 1] \times [0, 1], \qquad t \in [0, 11]. \tag{S16}$$

The first time unit is discarded, and $t \in [1, 11]$ is retained as data. The spatial resolution of the data is 64x64.

C Model specifications

All trainable models are optimized with Adam and use a cosine-decay learning-rate schedule with linear warmup and adaptive gradient clipping (Brock et al., 2021). During training, we employ validation-based early stopping. Model-specific learning rates, warmup lengths, decay steps, batch sizes, stopping criteria, and gradient clipping values are reported in the corresponding hyperparameter tables. All models are implemented in `JAX` using `Flax`, `optax`, `orbax`, `diffrax`, and `jax-dataloader`.

C.1 SINDy Model

Libraries. The library $\Phi_{q,r}$ used in the experiments is constructed as

$$E_q(\boldsymbol{u}) = \{\partial^\alpha u_p \mid \alpha \in \mathbb{N}^D,\ |\alpha| \le q,\ p = 1, \dots, P\}, \tag{S17}$$

$$\Phi_{q,r}(\boldsymbol{u}) = \{\hat{E}_q(\boldsymbol{u})^\beta \mid \beta \in \mathbb{N}^{K_q},\ |\beta| \le r\}, \tag{S18}$$

where $\hat{E}_q(\boldsymbol{u}) = (e_1(\boldsymbol{u}), \dots, e_{K_q}(\boldsymbol{u}))$ is a fixed ordering of the distinct elements of $E_q(\boldsymbol{u})$, and α and β are multi-indices for partial derivatives and monomials, respectively:

$$\partial^\alpha u_p = \frac{\partial^{|\alpha|} u_p}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \dots \partial x_D^{\alpha_D}}, \qquad \hat{E}_q(\boldsymbol{u})^\beta = \prod_{i=1}^{K_q} e_i(\boldsymbol{u})^{\beta_i}, \tag{S19}$$

$$|\alpha| = \sum_{i=1}^{D} \alpha_i, \qquad |\beta| = \sum_{i=1}^{K_q} \beta_i. \tag{S20}$$

The library $\Phi^{(2)}_{q,r}$ used in the ablation is constructed as

$$P_r(\boldsymbol{u}) = \{\boldsymbol{u}^\beta \mid \beta \in \mathbb{N}^P,\ |\beta| \le r\}, \tag{S21}$$

$$\Phi^{(2)}_{q,r}(\boldsymbol{u}) = \{e \cdot p \mid e \in E_q(\boldsymbol{u}),\ p \in P_r(\boldsymbol{u})\}. \tag{S22}$$

Training. During training, we apply a thresholding step after each epoch. In this step, the coefficient with the smallest absolute value is set to zero if its magnitude is below a prescribed threshold. This thresholding is activated only after a dataset-specific number of training epochs. For some datasets, thresholding alone was sufficient to obtain sparse and stable models; in these cases, we set the LASSO regularization strength to zero. For periodic domains, spatial derivatives are evaluated using circular padding, whereas for Dirichlet boundary conditions we enforce zero time derivatives at the boundary. For backpropagation through ODE solves, we use `RecursiveCheckpointAdjoint` implemented in `diffrax`. This adjoint stores a logarithmic number of solver states during the forward pass and recursively recomputes intermediate states during the backward pass, reducing

memory usage compared to backpropagating through all solver steps directly while avoiding the approximation error of a separately integrated continuous adjoint.

The SINDy hyperparameters for Lorenz63, Burgers' equation, and the Kuramoto–Sivashinsky equation were selected manually. For the reaction–diffusion system, the SINDy training hyperparameters were selected using Bayesian optimization. The resolution-16 hyperparameters are reported in Table S1, and the hyperparameters changed in the temporal resolution ablation are listed in Table S2.

Table S1: SINDy surrogate feature-library and training hyperparameters across datasets. The maximum number of solver steps given to the solver is max_steps divided by the number of intervals between time points in data.

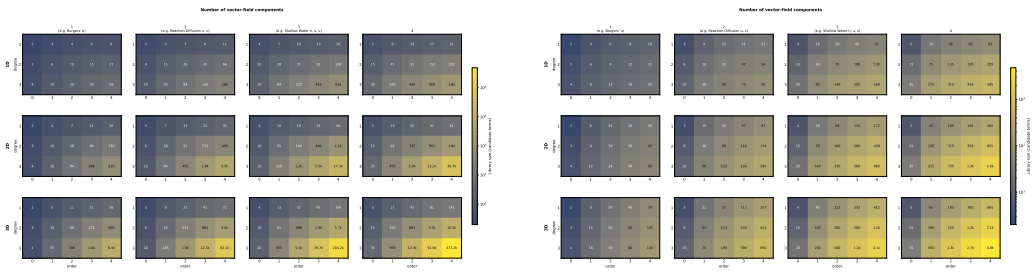
Feature library				
Hyperparameter	**Lorenz63**	**Burgers**	**KS**	**RD**
derivative order q	0	2	4	2
polynomial degree r	2	2	2	3
PySINDy initialization				
Hyperparameter	**Lorenz63**	**Burgers**	**KS**	**RD**
samples	1000	1000	20	20
polynomial degree	2	2	2	3
differentiation method	Spline	Spline	Spline	Finite Differences
smoothing parameter	0.1	0.01	1.0	--
spline order	3	3	3	--
derivative order	--	--	--	2
optimizer	STLSQ	STLSQ	STLSQ	STLSQ
elimination threshold	0.1	0.001	0.15	2.0e-04
L2 regularization α	0.05	0.05	0.05	4.2e-07
Training hyperparameters				
Hyperparameter	**Lorenz63**	**Burgers**	**KS**	**RD**
batch size	100	100	50	50
max epochs	1000	100	100	100
patience	1000	100	60	100
tolerance	1.0e-06	1.0e-05	1.0e-12	1.0e-08
learning rate schedule				
initial value	1.0e-05	1.0e-06	1.0e-05	1.0e-05
peak value	0.01	0.01	0.05	0.0032
end value	1.0e-05	0.0	1.0e-07	1.0e-05
warmup steps	1000	100	100	100
decay steps	9000	900	1900	1900
optimizer	adam	adam	adam	adam
β_1	0.9	0.9	0.9	0.9
β_2	0.999	0.999	0.999	0.999
ε	1.0e-08	1.0e-05	1.0e-08	1.0e-08
solver	Dopri8	Dopri8	Dopri8	Dopri8
absolute tolerance	1.0e-06	1.0e-05	1.0e-05	1.0e-05
relative tolerance	1.0e-05	1.0e-04	1.0e-04	1.0e-04
initial step size	1.0e-05	1.0e-05	1.0e-05	1.0e-05
max steps	400	1600	400	800
PID-controller				
proportional	0	0	0	0
integral	1	1	1	1
derivative	0	0	0	0
gradient clip max norm	0.5	10.0	0.5	1.0
elimination threshold	0.1	1.0e-06	0.1	1.0e-08
elimination start epoch	100	30	10	8
regularization weight	0	0	0	1.0e-08

Table S2: SINDy hyperparameters that differ at lower temporal resolutions from their resolution-16 values in Table S1.

PySINDy initialization				
Hyperparameter	**Lorenz63/12**	**Lorenz63/8**	**RD/12**	**RD/8**
elimination threshold	0.09	0.3	5.5e-04	4.5e-04
L2 regularization α	0.05	0.05	0.21	0.061
Training hyperparameters				
Hyperparameter	**Lorenz63/12**	**Lorenz63/8**	**RD/12**	**RD/8**
elimination threshold	0.1	0.1	9.1e-05	2.1e-04
elimination start epoch	100	100	20	7
regularization weight	0	0	3.3e-07	0.0022
lr peak value	0.01	0.1	0.0014	0.0013

Table S3: Diffrax ODE solver specifications for data generation, and inference with SINDy and Oracle-C. Inference with Oracle-C and SINDy use `RecursiveCheckpointAdjoint` for back-propagation through the solver.

ODE solver		
Parameter	**Data**	**Inference**
solver	Dopri8	Dopri8
absolute tolerance	1.0e-12	1.0e-6
relative tolerance	1.0e-10	1.0e-5
initial step size	1.0e-5	1.0e-5
max steps	10000	1600
PID-controller		
proportional	0	0
integral	1	1
derivative	0	0



(a) Library $\Phi^{(1)}$. (b) Library $\Phi^{(2)}$.

Figure S1: Parameter count of the SINDy model with varying library sizes.

C.2 Baselines

Neural surrogates. Both the U-Net and FNO surrogates instantiate f_θ as a discrete-time one-step predictor,

$$\boldsymbol{u}^{(n,i+1)} \approx f_\theta(\boldsymbol{u}^{(n,i)}), \tag{S23}$$

and are trained using the surrogate loss in equation 10. The U-Net uses an encoder–decoder architecture with convolutional ResNet blocks and skip connections, while the FNO uses spectral convolution layers with learned updates on truncated Fourier modes. These models serve as stable, basis-agnostic neural baselines for surrogate guidance.

The U-Net includes self-attention with four attention heads when attention is enabled. For datasets with periodic domains, convolutions use circular padding; otherwise, we use `SAME` padding. The remaining architecture and training hyperparameters for the U-Net and FNO surrogates are given in Tables S5 and S4, respectively.

For Lorenz63, the neural surrogate is a fully connected network with two hidden layers of width 128 and `gelu` activations after each hidden layer.

Oracle guidance. We include oracle baselines to separate the effect of the guidance mechanism from errors due to surrogate learning. The continuous-time oracle baseline, denoted Oracle-C, uses the same method-of-lines guidance as the SINDy surrogate, but replaces the learned vector field F_θ with the ground-truth dynamics F. Its guidance loss is therefore the trajectory prediction loss in equation 17, with the one-step map obtained by integrating the true semi-discrete PDE from t_i to t_{i+1}.

We also compare against a finite-difference residual baseline, denoted Oracle-D. Here, guidance is applied through a pointwise PDE residual rather than through continuous-time simulation:

$$\mathcal{L}_{\mathrm{D}}(\boldsymbol{u}) = \sum_{i=1}^{N_t-1} \left\| \partial_t^{\mathrm{FD}} \boldsymbol{u}^{(i)} - \widetilde{\boldsymbol{F}}(\boldsymbol{u}^{(i)}) \right\|_2^2, \tag{S24}$$

where $N_{\boldsymbol{u}}$ is the number of entries in the field tensor $\boldsymbol{u}$, $\partial_t^{\mathrm{FD}} \boldsymbol{u}^{(i)}$ is a finite-difference approximation of the time derivative, and $\widetilde{\boldsymbol{F}}$ is the ground-truth PDE operator with spatial derivatives discretized

by finite differences. All finite differences are second-order accurate central differences. Oracle-D therefore provides oracle PDE guidance without differentiating through a continuous-time solver.

Table S4: FNO surrogate architecture and training hyperparameters across PDE datasets.

Architecture hyperparameters			
Hyperparameter	**Burgers**	**KS**	**RD**
width	96	128	96
depth	4	5	4
padding (circular)	0	1	1
activation	gelu	gelu	gelu
modes per dim	24	28	24
Training hyperparameters			
Hyperparameter	**Burgers**	**KS**	**RD**
batch size	100	25	25
max epochs	400	200	150
patience	400	200	100
tolerance	1.0e-06	1.0e-06	1.0e-05
gradient clipping norm	0.2	0.05	0.2
learning rate schedule			
decay steps	3600	7600	5600
end value	5.0e-05	1.0e-06	5.0e-05
initial value	0.0	0.0	0.0
peak value	0.001	0.001	0.001
warmup steps	400	400	400
optimizer	adam	adam	adam
β_1	0.9	0.9	0.9
β_2	0.999	0.999	0.999
ε	1e-08	1e-08	1e-08

C.3 Diffusion Model

Parametrization. We use the EDM denoiser parametrization of Karras et al. (2022). For a noisy field $\boldsymbol{u}_\sigma$, the denoiser is written as

$$D_\psi(\boldsymbol{u}_\sigma, \sigma) = c_{\text{skip}}(\sigma)\boldsymbol{u}_\sigma + c_{\text{out}}(\sigma)F_\psi(c_{\text{in}}(\sigma)\boldsymbol{u}_\sigma; c_{\text{noise}}(\sigma)), \tag{S25}$$

where F_ψ is a U-Net. The EDM preconditioning coefficients are

$$c_{\text{skip}}(\sigma) = \frac{\sigma_{\text{data}}^2}{\sigma^2 + \sigma_{\text{data}}^2}, \qquad c_{\text{out}}(\sigma) = \frac{\sigma\,\sigma_{\text{data}}}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}, \tag{S26}$$

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}, \qquad c_{\text{noise}}(\sigma) = \frac{1}{4}\log\sigma. \tag{S27}$$

The U-Net architecture follows the ADM implementation of Dhariwal & Nichol (2021), with modifications for spatiotemporal field data. After the first encoder layer, we keep the number of channels fixed at twice the base-channel count. Attention blocks use four attention heads, and attention is omitted at the two largest resolution levels. For spatially two-dimensional datasets, attention is disabled. For these datasets, we also avoid downsampling along the temporal axis and use $3 \times 3 \times 3$ convolution kernels. For ODE data, no downsampling is used. Architecture and training hyperparameters are reported in Table S6.

Training. For each dataset, we train one unconditional EDM diffusion prior on the fully observed training trajectories. The denoiser is optimized with the weighted EDM denoising objective in equation 3. The diffusion model is trained once per dataset and reused across all guidance methods; only the surrogate models and guidance hyperparameters vary between guided variants. Dataset-specific architecture parameters, batch sizes, learning-rate schedules, gradient clipping values, and training horizons are reported in Table S6.

Table S5: U-Net surrogate architecture and training hyperparameters across PDE datasets.

Architecture hyperparameters			
Hyperparameter	**Burgers**	**KS**	**RD**
dimension	1	2	2
attention	1	0	1
base channels	48	96	48
groupnorm num groups	16	16	16
encoder res blocks / level	1	2	1
depth	3	3	3
convolution padding	SAME	CIRCULAR	CIRCULAR
Training hyperparameters			
Hyperparameter	**Burgers**	**KS**	**RD**
batch size	100	10	25
max epochs	400	300	100
patience	400	300	100
tolerance	1.0e-06	1.0e-06	1.0e-06
learning rate schedule			
decay steps	3600	29600	3600
end value	1.0e-07	1.0e-07	1.0e-06
initial value	0.0	0.0	0.0
peak value	1.0e-04	2.0e-05	3.0e-04
warmup steps	400	400	400
gradient clip max norm	0.2	0.2	0.2
optimizer	adam	adam	adam
β_1	0.9	0.9	0.9
β_2	0.999	0.999	0.999
ε	1e-08	1e-08	1e-08

Inference. For sampling, we use the Heun's method, which yields a deterministic sampler (Karras et al., 2022). The initial sample is drawn from $\mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ with $\sigma_{\max} = 80$. We use $\sigma_{\min} = 0.002$ and $\rho = 7$, and discretize the noise levels as

$$\sigma_i = \left(\sigma_{\max}^{1/\rho} + \frac{i}{N-1}\left(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}\right)\right)^{\rho}, \qquad i = 0, \ldots, N-1, \tag{S28}$$

followed by the final noise level $\sigma_N = 0$. Here N denotes the number of nonzero diffusion steps.

We use normalized guidance schedules

$$\gamma_{\mathcal{C},\sigma} = \frac{\gamma_{\mathcal{C}}\sigma}{|\mathcal{C}|\|\nabla_{\boldsymbol{u}_\sigma}\mathcal{L}_{\mathcal{C}}(\hat{\boldsymbol{u}}_0)\|_2}, \tag{S29}$$

$$\gamma_{\theta,\sigma} = \begin{cases} 0, & \sigma > 10, \\ \dfrac{\gamma_\theta \sigma}{\frac{1}{N_{\hat{\boldsymbol{u}}_0}}\|\nabla_{\boldsymbol{u}_\sigma}\mathcal{L}_\theta(\hat{\boldsymbol{u}}_0)\|_2}, & \sigma \le 10, \end{cases} \tag{S30}$$

where $\hat{\boldsymbol{u}}_0 = D_\psi(\boldsymbol{u}_\sigma, \sigma)$ is the denoised estimate and $|\mathcal{C}|$ is the number of context points, and $N_{\hat{\boldsymbol{u}}_0}$ is the number of entries in the spatiotemporal field tensor $\hat{\boldsymbol{u}}_0$. Context guidance is applied throughout the reverse diffusion trajectory, while surrogate guidance is activated only below $\sigma_{\theta,\max} = 10$. For continuous-time SINDy and Oracle-C guidance, the method-of-lines solve performed during inference uses the ODE solver settings reported in Table S3.

Continuous-time surrogate/oracle guidance can occasionally destabilize the reverse SDE: the guidance term transiently dominates the unconditional score, driving samples off-manifold. We mitigate this with a joint guidance clip. Let

$$g(\boldsymbol{u}_\sigma, \sigma) = \gamma_{\mathcal{C},\sigma}\nabla_{\boldsymbol{u}_\sigma}\mathcal{L}_{\mathcal{C}}(\hat{\boldsymbol{u}}_0) + \gamma_{\theta,\sigma}\nabla_{\boldsymbol{u}_\sigma}\mathcal{L}_\theta(\hat{\boldsymbol{u}}_0) \tag{S31}$$

denote the combined guidance term entering the EDM probability-flow ODE. We rescale g to

$$\tilde{g} = g\,\min\left\{1,\ \frac{k\,\|(\boldsymbol{u}_\sigma - D_\psi(\boldsymbol{u}_\sigma, \sigma))/\sigma\|_2}{\|g\|_2}\right\}, \tag{S32}$$

Table S6: Diffusion model (U-Net) architecture and training hyperparameters across datasets. The same hyperparameters are used at all temporal resolution levels, except at resolution 8, where no attention is used for Lorenz63 to improve training stability.

Architecture hyperparameters				
Hyperparameter	**Lorenz63**	**Burgers**	**KS**	**RD**
dimension	1	2	3	3
attention	1	1	0	0
base channels	128	48	48	48
groupnorm num groups	16	16	16	16
encoder res blocks / level	1	1	1	1
depth	3	3	3	3
noise embed dim	64	64	64	64
downsample temporal dim	False	True	False	False
convolution padding	SAME	SAME	CIRCULAR	CIRCULAR
skip connection scale	0.707	1	1	1
Training hyperparameters				
Hyperparameter	**Lorenz63**	**Burgers**	**KS**	**RD**
batch size	100	100	25	25
max epochs	300	500	200	200
patience	300	500	200	200
tolerance	1.0e-05	1.0e-05	1.0e-05	1.0e-05
learning rate schedule				
decay steps	5000	4500	7600	7800
end value	0.0	1.0e-07	1.0e-05	1.0e-05
initial value	0.0	0.0	0.0	0.0
peak value	2.0e-04	0.001	2.0e-04	0.001
warmup steps	200	500	400	200
gradient clip max norm	0.1	0.2	0.1	0.1
optimizer	adam	adam	adam	adam
β_1	0.9	0.9	0.9	0.9
β_2	0.999	0.999	0.999	0.999
ε	1e-08	1e-08	1e-08	1e-08

i.e., g is left unchanged whenever its norm is already at most k times the probability-flow drift norm $\|(\boldsymbol{u}_\sigma - D_\psi(\boldsymbol{u}_\sigma, \sigma))/\sigma\|_2$, and otherwise rescaled down to that bound. We tune k per dataset from a small set of guided sampling runs.

For Lorenz63 and Burgers' equation, we apply this clip to both Oracle-C and DiffSINDy, with $k = 15$ and $k = 3$, respectively. For reaction–diffusion, clipping is only needed for DiffSINDy, at $k = 1$. For Kuramoto–Sivashinsky, occasional bad samples appeared across all guidance methods, so we apply the clip uniformly with $k = 8$.

C.4 Bayesian optimization

We optimize the guidance strengths $\gamma_{\mathcal{C}}$ and γ_θ using Bayesian optimization implemented in `botorch` (Balandat et al., 2020). For inference-time guidance-weight selection, the BO objective is reconstruction RRMSE over the validation set, computed against the ground-truth fields as defined in equation S36. For surrogate-guided methods, we initialize BO with a grid search over the Cartesian product

$$\gamma_{\mathcal{C}} \in \{10^n \mid n \in \mathbb{Z}\} \cap [\gamma_{\mathcal{C},\min}, \gamma_{\mathcal{C},\max}], \tag{S33}$$

$$\gamma_\theta \in \{10^n \mid n \in \mathbb{Z}\} \cap [\gamma_{\theta,\min}, \gamma_{\theta,\max}], \tag{S34}$$

with the search over γ_θ omitted for observation-only guidance. The feasible grid evaluations are used to initialize the BO search. For each guidance strength, the upper BO bound is set to ten times the largest feasible value found during grid search, and the lower bound is set to 1. Grid-search ranges and sampling details are reported in Table S7.

We use independent Gaussian-process surrogates with RBF kernels for the validation objective and the feasibility constraint. The acquisition function is Log Expected Improvement. Candidate maximization uses 1024 raw samples and 30 restarts, with Monte Carlo estimates computed using a randomized quasi-Monte Carlo sampler, `SobolQMCNormalSampler`, with 256 samples (Sobol, 1967). We apply a feasibility gate to the raw samples using the constraint GP posterior. If the posterior at candidate λ_i is $c(\lambda_i) \sim \mathcal{N}(\mu_{c,i}, \sigma^2_{c,i})$, then the probability of feasibility is

$$p_{\text{feas}}(\lambda_i) = \Phi\left(-\frac{\mu_{c,i}}{\sigma_{c,i}}\right), \tag{S35}$$

where Φ is the standard normal CDF. We discard candidates with $p_{\text{feas}} < 0.6$. If fewer than five candidates remain, we evaluate the candidate with the largest feasibility probability; otherwise, we evaluate the feasible candidate with the largest acquisition value.

For Lorenz63, Burgers' equation, and reaction–diffusion, we run 100 BO iterations for surrogate-guided methods and 50 BO iterations for observation-only guidance. For Kuramoto–Sivashinsky, after grid initialization, we additionally draw 30 uniformly random initial candidates in log-transformed parameter space and then run 50 BO iterations.

Note that we generally do not apply guidance clipping during Bayesian optimization or its initialization. The exception is the continuous-time guidance methods, DiffSINDy and Oracle-C, for which Bayesian Optimization produced particularly unstable results without clipping.

Table S7: Sampling-time guidance hyperparameters across datasets, with $\gamma_{\mathcal{C}}$ and γ_θ given as their Bayesian-optimization initialization range. For plain observation-only DPS guidance, we use the bounds $\gamma_{\mathcal{C}} \in [10, 10^5]$.

	Test configuration			
Hyperparameter	**Lorenz63**	**Burgers**	**KS**	**RD**
$\gamma_{\mathcal{C}}$	$[100, 1000]$	$[100, 1000]$	$[100, 10^5]$	$[100, 10^4]$
γ_θ	$[10, 10^4]$	$[100, 1000]$	$[1000, 10^5]$	$[10, 10^5]$
sampling steps	200	200	100	100

SINDy library ablation. For the experiment carried in Section 5.2, we modify the constrained BO procedure as follows:

- optimize the validation surrogate loss instead of reconstruction RRMSE;
- initialize with 20 uniformly sampled candidates in log-parameter space, omitting the log-grid initialization;
- run 200 BO iterations;
- use Matérn kernels for both the objective and feasibility GP surrogates;
- optimize the six SINDy hyperparameters listed in Table S8;
- keep all remaining hyperparameters fixed to the Burgers' equation settings in Table S1, except for a maximum of 30 training epochs.

These modifications were necessary to stabilize BO under the highly variable optimization landscape induced by large and heterogeneous SINDy libraries.

C.5 Evaluation metrics

We evaluate generated trajectories against the corresponding ground-truth test trajectories. Let $\hat{\boldsymbol{u}}_0^{(\text{new})} = (\hat{\boldsymbol{u}}_0^{(\text{new},0)}, \ldots, \hat{\boldsymbol{u}}_0^{(\text{new},N_t)})$ denote a generated clean trajectory for a new test instance, and let $\boldsymbol{u}^{(\text{new})}$ denote the corresponding ground-truth trajectory. All norms below are computed over the full spatiotemporal field tensor, including time, space, and state channels.

Table S8: Bayesian-optimization search ranges for the SINDy library ablation. The BO objective is the validation surrogate loss.

Hyperparameter	Lower bound	Upper bound
PySINDy initialization		
elimination threshold	10^{-8}	10^{-2}
L2 regularization α	10^{-8}	10
SINDy training		
elimination threshold	10^{-8}	10^{-3}
elimination start epoch	1	20
regularization weight	10^{-8}	0.1
learning rate peak value	10^{-3}	0.1

Relative root mean squared error. For each test trajectory, we report

$$\mathrm{RRMSE}\left(\hat{\boldsymbol{u}}_0^{(\mathrm{new})}, \boldsymbol{u}^{(\mathrm{new})}\right) = \left(\frac{\mathrm{mean}\left[\left(\hat{\boldsymbol{u}}_0^{(\mathrm{new})} - \boldsymbol{u}^{(\mathrm{new})}\right)^2\right]}{\left\|\boldsymbol{u}^{(\mathrm{new})}\right\|_2^2} \right)^{1/2} . \tag{S36}$$

The reported values are the mean of this quantity over the test trajectories and diffusion noise initializations.

Surrogate prediction error. For a learned or oracle one-step map f, we measure one-step prediction error on test trajectories using

$$\mathcal{L}_f\left(\boldsymbol{u}^{(\mathrm{new})}\right) = \mathrm{mean}_i\left[\left(\boldsymbol{u}^{(\mathrm{new},i+1)} - f\left(\boldsymbol{u}^{(\mathrm{new},i)}\right)\right)^2\right], \tag{S37}$$

where the mean is taken over the entries in the field snapshot tensors and time indices i. For discrete-time neural surrogates, $f = f_\theta$ is the learned one-step predictor. For continuous-time SINDy and Oracle-C, f is obtained by integrating the corresponding method-of-lines system from t_{i-1} to t_i. In the experiments, we take the square root of equation S37 and report the mean taken over the test set as the surrogate RMSE.

Physical consistency. Physical consistency is measured by replacing the learned surrogate with the ground-truth one-step dynamics. Let f_{PDE} denote the ground-truth method-of-lines time-stepper from t_i to t_{i+1}. For a generated trajectory, the one-step physical-consistency residual is $\mathbf{r}_{\mathrm{PDE}}^{(i)} = f_{\mathrm{PDE}}(\hat{\boldsymbol{u}}_0^{(\mathrm{new},i)}) - \hat{\boldsymbol{u}}_0^{(\mathrm{new},i+1)}$, and the corresponding loss is

$$\mathcal{L}_{\mathrm{PDE}}\left(\hat{\boldsymbol{u}}_0^{(\mathrm{new})}\right) = \mathrm{mean}\left[\boldsymbol{r}_{\mathrm{PDE}}^2\right] . \tag{S38}$$

Runtime. Runtime is measured as the wall-clock inference time required to generate on the entire test set of 100 trajectories, reported in seconds on NVIDIA GH200 nodes. For Kuramoto–Sivashinsky and reaction–diffusion, inference uses a batch size of 25, so sampling is run sequentially four times to cover the full test set.

D Qualitative Results

Samples generated by the model along with reconstruction residuals, PDE residuals $\boldsymbol{r}$, and the ground truth are given in Figure S2.

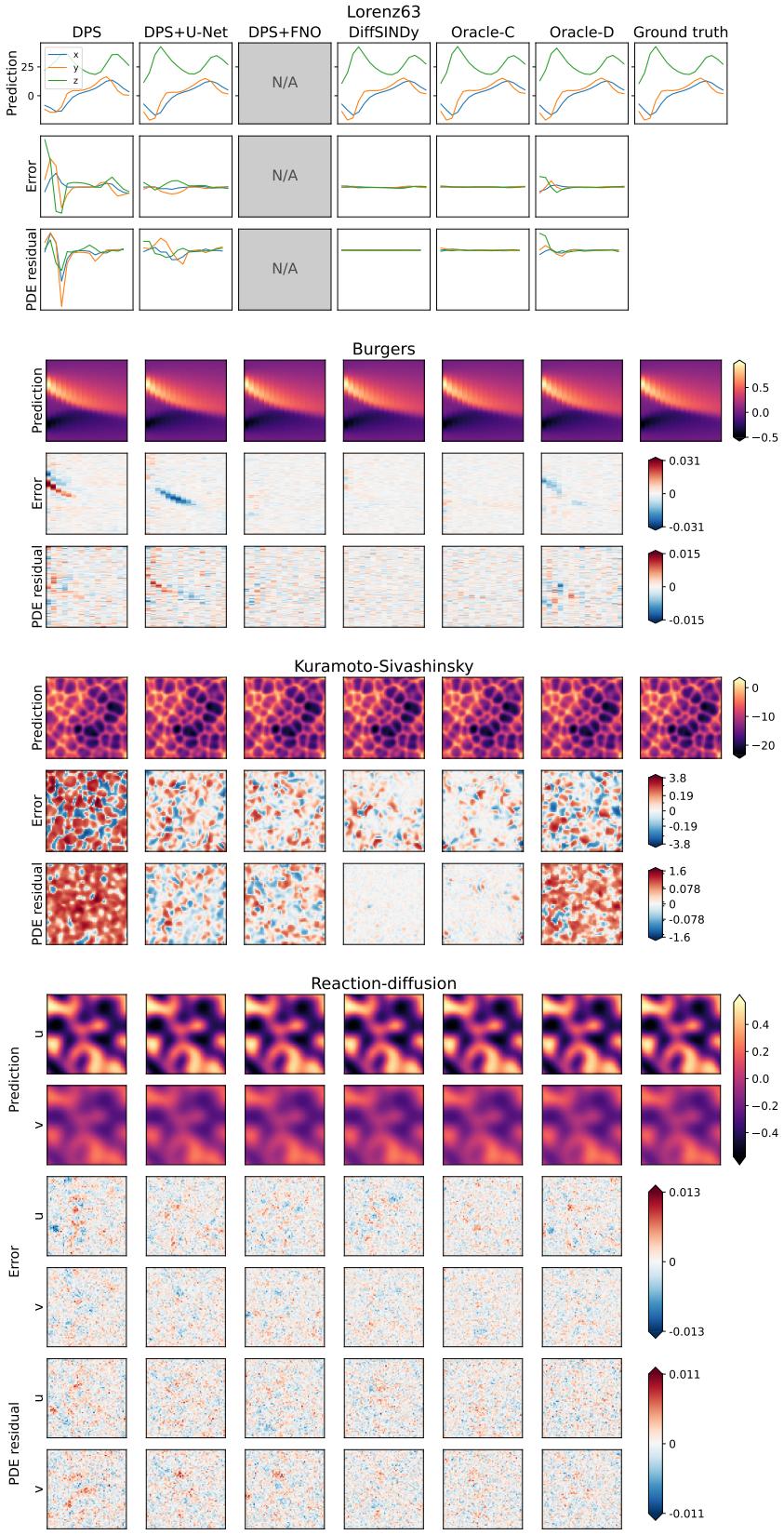


Figure S2: Generated samples and the corresponding residuals.