Learning Spatiotemporal Diffusion Models with Continuous-time PDE Guidance

Juho Latukka **Markus Heinonen** **Harri Lähdesmäki**

Department of Computer Science
Aalto University

Abstract

Spatiotemporal Partial Differential Equations (PDEs) are fundamental to modeling complex physical systems. While significant progress has been made in discovering these dynamics from observed data (the ”discovery task”), reconstructing a trajectory of entire fields from sparse, arbitrary observations – the ”imputation task” – remains a substantial challenge. We propose a novel framework that first learns the unknown PDE dynamics using a data-driven approach, specifically by adapting the SINDy method. Subsequently, we employ a generative denoising diffusion model to impute complete spatiotemporal fields. Dual-guidance of the generative process by sparse observations and learned PDE dynamics enables accurate and physically consistent reconstruction of spatiotemporal phenomena even when both the dynamics and the initial conditions are unknown.

1 Introduction

We consider spatiotemporal PDE systems $\mathbf{u}(t,\mathbf{x}) \in \mathbb{R}^P$ over continuous space $\mathbf{x} \in \Omega \subset \mathbb{R}^D$ and time $t \in \mathbb{R}$, governed by a time derivative

$$\dot{\mathbf{u}}(t,\mathbf{x}) = f_\theta(t,\mathbf{x},\mathbf{u}) \tag{1}$$

under some initial and boundary conditions. We assume N observed trajectories $\mathcal{D} = \{\mathbf{u}^{(n)}\}_{n=1}^N$ from the same dynamics f_θ but with varying initial conditions $\mathbf{u}^{(n,0)}$. Each trajectory $\mathbf{u}^{(n)} = (\mathbf{u}^{(n,0)}, \mathbf{u}^{(n,1)}, \ldots, \mathbf{u}^{(n,N_t)})$ is observed at time points $t_0, t_1, \ldots, t_{N_t}$ on a fixed spatial grid. We assume that all trajectories in $\mathcal{D}$ are observed on the same spatiotemporal grid.

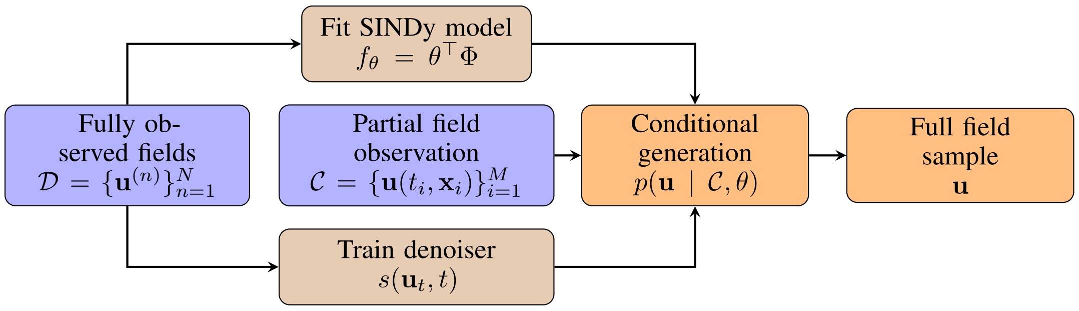
In the *discovery* task the vector field f_θ is learned from data $\mathcal{D}$, and at test time we can simulate solutions $\mathbf{u}^{(\text{new},0)} \mapsto \mathbf{u}^{(\text{new})}$ from given initial conditions. In a more challenging *imputation* task we need to solve $\mathbf{u}^{(\text{new})}$ given only some arbitrary context points $\mathcal{C} = \{\mathbf{u}(t_i,\mathbf{x}_i)\}_{i=1}^M$. We cannot use a forward solver since $\mathbf{u}^{(\text{new},0)}$ is not given.

We propose to use a generative diffusion model to impute the field $\mathbf{u}^{(\text{new})}$ by denoising under two kinds of guidance to ensure that the sampled field satisfies both the context points $\mathcal{C}$ and the PDE equation (1). While previous works have considered guidance assuming known spatio-temporal dynamics, we consider a more general case of unknown dynamics. This requires first learning the PDE, for which we follow the SINDy method (Brunton et al., 2016).

2 PDE Discovery

Following SINDy, we parameterize the PDE as a linear combination of spatial derivative terms

$$f_\theta(t,\mathbf{x},\mathbf{u}) = \sum_k \theta_k \phi_k(t,\mathbf{x},\mathbf{u}) = \theta^\top \Phi(t,\mathbf{x},\mathbf{u}), \tag{2}$$



Fit SINDy model $f_\theta = \theta^\top \Phi$
Fully observed fields $\mathcal{D} = \{\mathbf{u}^{(n)}\}_{n=1}^{N}$
Partial field observation $\mathcal{C} = \{\mathbf{u}(t_i, \mathbf{x}_i)\}_{i=1}^{M}$
Conditional generation $p(\mathbf{u} \mid \mathcal{C}, \theta)$
Full field sample $\mathbf{u}$
Train denoiser $s(\mathbf{u}_t, t)$

Figure 1: Illustration of the workflow.

where $\theta = (\theta_1, \dots, \theta_K)^\top \in \mathbb{R}^K$ are the learned weights, and $\Phi = (\phi_1, \dots, \phi_K)^\top$ is a predefined basis function library of size K. We use function libraries of polynomial terms

$$\Phi = \Big(1, \mathbf{u}, \mathbf{u}^2, \dots, \mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_{xx}, \mathbf{u}_{xy}, \mathbf{u}_{yy}, ..., \mathbf{u}\mathbf{u}_x, \mathbf{u}\mathbf{u}_y, \mathbf{u}_x^2, \mathbf{u}_y^2, \mathbf{u}_x\mathbf{u}_y \dots\Big)^\top,$$

where subscript denotes the partial derivative.

Dynamics learning In standard SINDy learning we optimize the PDE to match the gradients of the data interpolations. We opt for simulation-based optimization, which gives accurate estimates also in the case of temporally sparse observations. We learn sparse dynamics using LASSO regularization with a look-ahead loss

$$\mathcal{L}(\theta) = \sum_{n,i=1}^{N,N_t} \left\| \mathbf{u}^{(n,i)} - \mathbf{u}_\theta\big(t_i \mid \mathbf{u}^{(n,i-1)}\big) \right\|^2 + \lambda \|\theta\|_1, \tag{3}$$

where $\mathbf{u}_\theta\big(t_i \mid \mathbf{u}^{(n,i-1)}\big)$ is the forward-simulated solution. As the simulation method, we use the method of lines, where the system is discretized spatially into a grid of coupled ODEs that are solved in tandem (Schiesser, 1991). As ODE solvers, we use explicit PID controlled adaptive numerical solvers implemented in the `diffrax` library (Kidger, 2021).

In non-convex optimization problems, the initial guess requires careful consideration. We use random initialization on a small shell around the origin of the weight space. However, during the simulation process, the PDE defined by the SINDy model may become ill-posed and difficult to simulate numerically. Thus, we also experiment on first learning the SINDy model with finite differences approximations on all dimensions, and using this coarse approximation of the dynamics as the initial guess. This is implemented using the `PySINDy` package (de Silva et al., 2020; Kaptanoglu et al., 2022).

3 Diffusion Model

A diffusion model gradually noises the data distribution, and learns to remove the noise (Sohl-Dickstein et al., 2015). We apply standard score-based diffusion models on the solution fields $\mathbf{u}$, where we interpret the solution as a tensor with time and space axes as dimensions. Note that we refer to t (without a subscript) as diffusion time, while the solution $\mathbf{u}_t$ is the full spatiotemporal field at diffusion time t.

Variance-exploding diffusion defines dual processes (Song et al., 2021; Karras et al., 2022)

$$\text{forward:} \quad d\mathbf{u}_t = \sqrt{2\dot{\sigma}_t \sigma_t} d\mathbf{w}_t, \qquad \mathbf{u}_0 \sim p_0 \tag{4}$$

$$\text{reverse:} \quad d\mathbf{u}_t = -\dot{\sigma}_t \sigma_t \nabla \log p_t(\mathbf{u}_t) dt, \qquad \mathbf{u}_T \sim p_T, \tag{5}$$

where the noise schedule is σ_t, $\mathbf{w}$ is a Wiener process, and the processes share marginal p_t. To learn the score function $s(\mathbf{u}_t, t) = \nabla \log p_t(\mathbf{u}_t)$, we use denoising score matching (Hyvärinen and Dayan, 2005; Vincent, 2011; Song et al., 2021),

$$\mathcal{L}(\mathbf{s}) = \mathbb{E}_{t, \mathbf{u}_0, \mathbf{u}_t \mid \mathbf{u}_0} \left[\left\| s(\mathbf{u}_t, t) - \nabla_{\mathbf{u}_t} \log p(\mathbf{u}_t \mid \mathbf{u}_0) \right\|^2 \right], \tag{6}$$

where the score is over the Gaussian noise distribution $p(\mathbf{u}_t \mid \mathbf{u}_0) = \mathcal{N}(\mathbf{u}_t \mid \mathbf{u}_0, \sigma_t^2 I)$. We parameterize the score function s with a U-Net, and train the diffusion model using the data $\mathcal{D}$.

4 PDE Guidance

Our goal is to sample trajectories $\mathbf{u} \sim p(\mathbf{u} \mid \mathcal{C}, \theta)$ using the trained diffusion model such that the observed context points $\mathcal{C}$ and the learned PDE equation $\dot{\mathbf{u}} = f_\theta$ are satisfied. In prior work similar to ours, adherence to a known PDE equation is often measured by finite difference methods (Shu et al., 2023; Huang et al., 2024; Jacobsen et al., 2024). However, if the data is sparse in one dimension, e.g., time, this does not lead to a satisfactory approximation. Instead, we use a simulation-based approach to obtain a more accurate score

$$\mathcal{L}_{\mathrm{PDE}}(\mathbf{u}, \theta) = \sum_{n,i=1}^{N,N_t} \left\| \mathbf{u}^{(n,i)} - \mathbf{u}_\theta\big(t_i \mid \mathbf{u}^{(n,i-1)}\big) \right\|^2, \tag{7}$$

where $\mathbf{u}_\theta\big(t_i \mid \mathbf{u}_{i-1}^{(n)}\big)$ is the simulated solution using the learned SINDy weights θ. The solver is similar to the one described in Section 2.

By applying the Bayes' rule twice, the score function can be broken into individual parts

$$s(\mathbf{u}_t \mid \mathcal{C}, \theta) = \nabla_{\mathbf{u}_t} \log p(\theta \mid \mathbf{u}_t, \mathcal{C}) + \nabla_{\mathbf{u}_t} \log p(\mathcal{C} \mid \mathbf{u}_t) + \underbrace{\nabla_{\mathbf{u}_t} \log p(\mathbf{u}_t)}_{\approx s(\mathbf{u}_t)}, \tag{8}$$

where the conditional score directs towards regions with high compatibility with the PDE equation and context points. We follow the Diffusion Posterior Sampling (DPS) framework. We approximate the error in the context points as a Gaussian, which gives

$$\nabla_{\mathbf{u}_t} \log p(\mathcal{C} \mid \mathbf{u}_t) \approx -\gamma_{\mathrm{OBS,t}} \nabla_{\mathbf{u}_t} \mathcal{L}_{\mathrm{OBS}}(\hat{\mathbf{u}}_0(\mathbf{u}_t, t), \mathcal{C}), \tag{9}$$

where $\gamma_{\mathrm{OBS},t}$ is the guidance schedule, $\hat{\mathbf{u}}_0(\mathbf{u}_t, t) = \mathbb{E}[\mathbf{u}_0 \mid \mathbf{u}_t] = \mathbf{u}_t + \sigma_t^2 s(\mathbf{u}_t, t)$ is a single-step denoising mean following Tweedie's formula and

$$\mathcal{L}_{\mathrm{OBS}}(\mathbf{u}, \mathcal{C}) = \left\| \mathcal{C} - \mathcal{M}_{\mathcal{C}}(\mathbf{u}) \right\|^2, \tag{10}$$

where $\mathcal{M}_{\mathcal{C}}(\mathbf{u})$ is a measurement map (Park et al., 2024; Chung et al., 2024; Huang et al., 2024). In our experiments, the measurement map $\mathcal{M}_{\mathcal{C}}$ is a simple sparsity mask without added noise. However, DPS is also capable of solving inverse problems with nonlinear and noisy maps $\mathcal{M}_{\mathcal{C}}$.

Similarly, we approximate the error in θ given $\mathbf{u}_0$ as a Gaussian such that

$$\nabla_{\mathbf{u}_t} \log p(\theta \mid \mathbf{u}_t, \mathcal{C}) \approx -\gamma_{\mathrm{PDE},t} \nabla_{\mathbf{u}_t} \mathcal{L}_{\mathrm{PDE}}(\hat{\mathbf{u}}_0(\mathbf{u}_t, t), \theta), \tag{11}$$

where $\gamma_{\mathrm{PDE},t}$ is the guidance schedule.

The complete workflow is depicted in Figure 1. Further model specifications can be found in Appendix B.

5 Experiments

The training, validation and testing data are generated by solving Burgers' equation, Shallow Water Equations and Navier-Stokes with randomized initial conditions. The solver is based on the method of lines coupled with Runge-Kutta solvers implemented in the `diffrax` library. Further specifications can be found in Appendix A.

For context, we randomly select approximately 5 % of the grid points. Hyperparameters for generation are chosen using Bayesian optimization as implemented in the `bayesian-optimization` and `Weights&Biases` packages (Mockus et al., 2014; Fernando Nogueira, 2014; Biewald, 2020).

The results shown in Table 1 show the improvement in the results compared to a basic diffusion model with no PDE guidance. We also show the results obtained by using guidance in which the time dimension is also discretized in the same resolution as the data. This is given by

$$\mathcal{L}_{\mathrm{DISCRETE}}(\mathbf{u}, \theta) = \left\| \partial_t \mathbf{u} - f_\theta(\mathbf{u}) \right\|^2, \tag{12}$$

Table 1: Mean and standard deviation of Relative Root Mean Squared Error (RRMSE) of the model on test data across 30 distinct draws $\mathbf{u}_T \sim p_T(\mathbf{u}_T)$ for each test trajectory. We also report the mean and standard deviation of the spread across the test trajectories, where spread is the expected standard deviation of distinct predictions for the same data. D+C stands for plain diffusion model with only context guidance. D+C+P stands for diffusion model with guidance based on context and the true underlying (oracle) PDE, with D+C+Pd using the guidance given by (12). D+C+S is the model described in this work.

RRMSE (10^{-4})	D+C	D+C+Pd	D+C+S	D+C+P
Burgers' Equation	4.7±0.3	4.6±0.3	2.5±0.1	2.4±0.1
SWE	0.91±0.62	0.56±0.27	0.46±0.10	0.40±0.06
Navier-Stokes Equations	0.32±0.24	0.25±0.09	0.18±0.01	0.16±0.01
Spread (10^{-3})				
Burgers' Equation	3.9±2.4	4.1±3.3	2.8±1.0	4.5±0.9
SWE	14±25	7.8±14.2	5.9±3.9	4.9±2.2
Navier-Stokes Equations	5.9±13.4	2.6±3.0	3.0±0.2	2.6±0.2

where ∂_t is approximated by the central difference of the solution field $\mathbf{u}$ operating in the temporal dimension, and $f_\theta(\mathbf{u})$ are its assumed true time-derivatives learned by the SINDy model. As a comparison, we also provide results obtained through guidance using the ground-truth PDE. This oracle guidance leads to smaller error in prediction, as expected.

Additionally, we show qualitative results comprising generated trajectories along with the corresponding context and residuals. These are presented in Figure 2.

Burgers' equation The 1-dimensional Burgers' equation describes the propagation of a 1-dimensional wave. For this experiment, we define the function library as the monomial terms up to degree 3 with the largest derivative order of 3 (with intersect):

$$\Phi = \left(1, u, u^2, \dots, u_x, u_y, u_{xx}, u_{xy}, u_{yy}, ..., uu_x, uu_y, u_x^2, u_y^2, u_x u_y, \dots\right)^\top. \tag{13}$$

The initial guess for the weights is obtained using `PySINDy`.

Shallow Water Equations Shallow Water Equations (SWE) model the propagation of surface perturbations in shallow bodies of water. They can be derived from the Navier-Stokes equations. For each component of the vector field, we define the function library as

$$\Phi = \left(u, v, h_x, h_y, u_x, v_y, uu_x, vu_y, uv_x, vv_y, hu_x, uh_x, hv_y, vh_y, u_{xx}, u_{yy}, v_{xx}, v_{yy}\right)^\top. \tag{14}$$

We use random initialization for the weights.

Navier-Stokes Navier-Stokes equation models the movement of fluids. We use the 2-dimensional form of the equation. For each component of the vector field, we define the function library as

$$\Phi = \left(1, u_x, v_x, u_y, v_y, uu_x, uu_y, uv_x, uv_y, vv_x, vv_y, vu_x, vu_y, u_{xx}, v_{xx}, u_{xy}\right)^\top. \tag{15}$$

We use random initialization for the weights.

6 Conclusion

We introduced a novel framework for spatiotemporal field imputation given unknown PDE dynamics. Our method tackles the challenging problem of reconstructing complete fields from sparse observations, where initial conditions are often unavailable. The core contribution is the enhancement of predictions generated by a denoising diffusion model, conditioned on a sparse set of context points, with guidance based on a *learned* PDE. This PDE vector field is effectively learned from data using

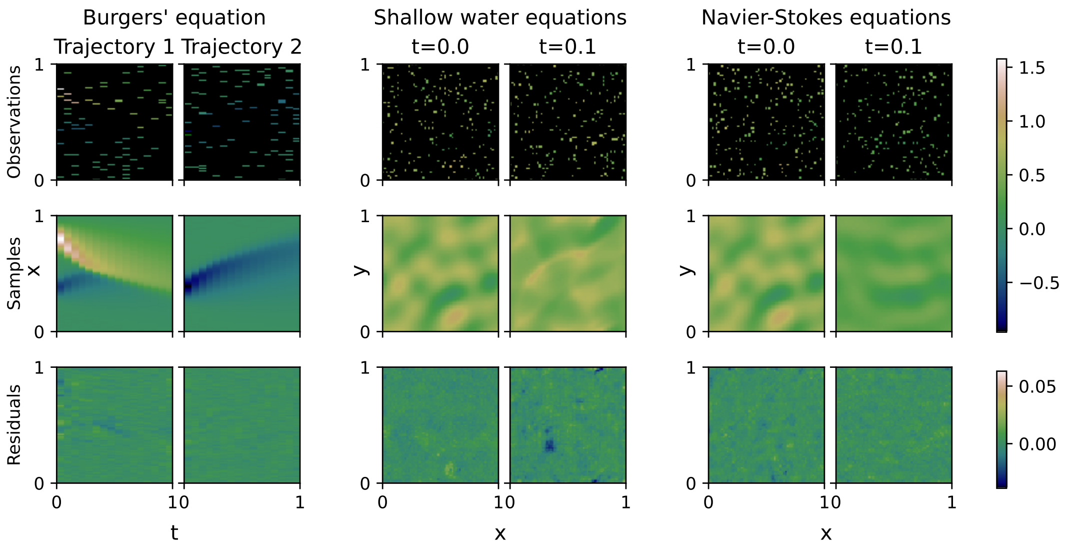


Figure 2: Generated samples with the corresponding context and residuals.

Table 2: Mean and standard deviation of the actual runtime and proportion of failed predictions on test data across 30 iterations with distinct $\mathbf{u}_T \sim p_T(\mathbf{u}_T)$, measured in seconds. All tests are run on Nvidia H200 SXM GPUs.

Runtime (s)	D+C	D+C+Pd	D+C+S	D+C+P
Burgers' Equation	130±3	187±3	554±44	496±4
SWE	333±2	495±2	1444±96	768±17
Navier-Stokes Equations	333±2	454±4	870±15	737±9
Failure in prediction (%)				
Burgers' Equation	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
SWE	1.0±0.9	0.0±0.0	0.3±0.5	0.2±0.4
Navier-Stokes Equations	0.0±0.0	0.3±0.5	0.2±0.4	0.2±0.4

an adapted SINDy method. Our approach offers a solution for inferring complete system states from limited data, integrating data-driven PDE learning with generative modeling. Additionally, since the guidance is applied through continuous-time simulation of the PDE, our model is able to operate even on temporally sparse data. Our method paves the way for more accurate analysis and prediction in complex physical systems.

Future work and limitations The main limitation is the failure of convergence of the proposed SINDy algorithm for more general and large function libraries. This might be due to non-linear nature of the optimization problem, which leads to convergence to a local minimum. Another contributing factor might be multicollinearity and correlations between the functions in the function library given the data (North et al., 2022; Kiser et al., 2023).

Another problem is the possibility of attempting to forward-integrate an ill-posed PDE during SINDy training. This leads to the PID-controlled solver requiring arbitrarily many steps. Overall, forward-integration, and calculating gradients through the solver is computationally intensive. Runtime comparison on test data is given in Table 2.

During inference, the solver sometimes obtains difficult initial values and exceeds the pre-determined number of maximum steps. This issue is exacerbated by large guidance strengths in schedules $\gamma_{\text{PDE},t}, \gamma_{\text{OBS},t}$. With high enough guidance strengths, numerical issues can lead to failure even without a solver. The proportions of failed predictions on the test data can be found in Table 2.

References

Biewald, L. (2020). Experiment Tracking with Weights and Biases.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2024). Diffusion Posterior Sampling for General Noisy Inverse Problems.

de Silva, B., Champion, K., Quade, M., Loiseau, J.-C., Kutz, J., and Brunton, S. (2020). PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104.

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., Kapturowski, S., Keck, T., Kemaev, I., King, M., Kunesch, M., Martens, L., Merzic, H., Mikulik, V., Norman, T., Papamakarios, G., Quan, J., Ring, R., Ruiz, F., Sanchez, A., Sartran, L., Schneider, R., Sezener, E., Spencer, S., Srinivasan, S., Stanojević, M., Stokowiec, W., Wang, L., Zhou, G., and Viola, F. (2020). The DeepMind JAX Ecosystem.

Fernando Nogueira (2014). Bayesian Optimization: Open source constrained global optimization tool for Python.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising Diffusion Probabilistic Models.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. (2022). Video diffusion models. *arXiv:2204.03458*.

Huang, J., Yang, G., Wang, Z., and Park, J. J. (2024). DiffusionPDE: Generative PDE-Solving Under Partial Observation.

Hyvärinen, A. and Dayan, P. (2005). Estimation of non-normalized statistical models by score matching. *JMLR*.

Iakovlev, V., Heinonen, M., and Lähdesmäki, H. (2023). Learning Space-Time Continuous Neural PDEs from Partially Observed States.

Jacobsen, C., Zhuang, Y., and Duraisamy, K. (2024). CoCoGen: Physically-Consistent and Conditioned Score-based Generative Models for Forward and Inverse Problems.

Kaptanoglu, A. A., de Silva, B. M., Fasel, U., Kaheman, K., Goldschmidt, A. J., Callaham, J., Delahunt, C. B., Nicolaou, Z. G., Champion, K., Loiseau, J.-C., Kutz, J. N., and Brunton, S. L. (2022). PySINDy: A comprehensive Python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994.

Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the Design Space of Diffusion-Based Generative Models.

Kidger, P. (2021). *On Neural Differential Equations*. PhD thesis, University of Oxford.

Kiser, S. L., Guskov, M., Rébillat, M., and Ranc, N. (2023). Exact identification of nonlinear dynamical systems by Trimmed Lasso.

Kværnø, A. (2004). Singly diagonally implicit Runge–Kutta methods with an explicit first }stage. *BIT Numerical Mathematics*, 44(3):489–502.

Mockus, J., Tiesis, V., and Zilinskas, A. (2014). The application of Bayesian methods for seeking the extremum. In *Towards Global Optimization*, volume 2, pages 117–129.

North, J. S., Wikle, C. K., and Schliep, E. M. (2022). A Bayesian Approach for Spatio-Temporal Data-Driven Dynamic Equation Discovery.

Park, C. Y., McCann, M. T., Garcia-Cardona, C., Wohlberg, B., and Kamilov, U. S. (2024). Random Walks with Tweedie: A Unified Framework for Diffusion Models.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library.

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

Schiesser, W. E. (1991). *The Numerical Method of Lines*. Academic Press.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-Attention with Relative Position Representations.

Shu, D., Li, Z., and Barati Farimani, A. (2023). A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-Based Generative Modeling through Stochastic Differential Equations.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Computation*.

Wei, L., Hu, P., Feng, R., Du, Y., Zhang, T., Wang, R., Wang, Y., Ma, Z.-M., and Wu, T. (2024). Generative PDE Control. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*.

A Dataset specifications

Here we detail the exact specifications used in generating the datasets used in experiments.

Each dataset is generated using the method of lines. The PDEs are discretized spatially using second-order accurate central differences. The adaptive PID step size controller parameters of the ODE solvers used for data generation are detailed in Table 3. We use a simple I-controller, setting the coefficients of the proportional and derivative terms to zero.

A.1 Burgers' equation

With a simple diffusion and advection terms, the 1-dimensional Burgers' equation models the diffusion and propagation of disturbances across space. It is written as

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + v\frac{\partial^2 u}{\partial x^2}. \tag{16}$$

We use the diffusivity constant $v = 0.01$ and generate the data using initial data defined as a superposition of two random Gaussian functions. Specifically, the initial data is generated as

$$\begin{aligned} u_0(x) &= a_0 \exp\left[-\frac{1}{2\sigma_0}(x-\mu_0)^2\right] - a_1 \exp\left[-\frac{1}{2\sigma_1}(x-(\mu_1+0.4))^2\right], \, x \in \Omega && (17)\\ a_0, a_1 &\sim \mathcal{U}(0,2) && (18)\\ \mu_0, \mu_1 &\sim \mathcal{U}(0.2, 0.4) && (19)\\ \sigma_0, \sigma_1 &\sim \mathcal{U}(0.05, 0.15), && (20)\end{aligned}$$

where $\mathcal{U}$ denotes the uniform distribution (Wei et al., 2024). Finally, we fix the boundary values to the initial values.

We use a spatial resolution of 128 points and a temporal resolution of 16 time points. Our spatiotemporal domain is the unit square $[0, 1] \times [0, 1]$. We simulate 1000 training samples for training the model, 100 samples for validation and 500 samples for testing. As the simulation method, we use the method of lines with Kværnø (2004) 5/4 method as the ODE solver. It is an implicit Runge-Kutta method with an explicit first stage.

A.2 Shallow Water Equations

The non-conservative Shallow Water Equations are defined through the evolution of three scalar fields: the water surface h, and the velocity components u and v in dimensions x and y, respectively. Omitting the linear terms relating to the Coriolis force and friction, the SWE are written as

$$\begin{aligned} \frac{\partial h}{\partial t} &= -\frac{\partial}{\partial x}\left((H+h)u\right) - \frac{\partial}{\partial y}\left((H+h)u\right) && (21)\\ \frac{\partial u}{\partial t} &= -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial x} - g\frac{\partial h}{\partial x} + v(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) && (22)\\ \frac{\partial v}{\partial t} &= -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial x} - g\frac{\partial h}{\partial y} + v(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}), && (23)\end{aligned}$$

where h, u, v are functions of $(x, y) \in \Omega$ and $t \in (0, T]$, and $u(x, y, 0) = v(x, y, 0) = 0$, $h(x, y, 0) = h_0(x, y)$. We let the gravitational constant $g = 9.81$, the diffusivity constant $v = 0.0002$ and the mean surface height $H = 2.0$. The initial data used for data generation is a superposition of sinusoidal waves at different amplitudes and frequencies. Specifically, following the work of Iakovlev et al. (2023), we set

$$\begin{aligned} \tilde{h}_0(x,y) &= \sum_{k,l} \lambda_{k,l}\cos(2\pi kx + 2\pi ly) + \gamma_{k,l}\sin(2\pi kx + 2\pi ly) && (24)\\ h_0(x,y) &= \frac{\tilde{h}_0(x,y) - \min \tilde{h}_0}{\max \tilde{h}_0 - \min \tilde{h}_0} && (25)\\ \lambda_{k,l}, \gamma_{k,l} &\sim \mathcal{N}(0,1), \; k,l = -N, -N+1, ..., N, && (26)\end{aligned}$$

Table 3: ODE solver configurations for each dataset.

	Stage	Initial step	Rel. tol.	Abs. tol.	Max steps
Burgers' Equation	Data generation	10^{-8}	10^{-10}	10^{-12}	10000
	SINDy training	10^{-5}	10^{-6}	10^{-5}	1600
	SINDy validation	10^{-5}	10^{-7}	10^{-6}	1600
	PDE guidance	10^{-5}	10^{-5}	10^{-6}	1600
SWE	Data generation	10^{-8}	10^{-8}	10^{-10}	100000
	SINDy training	10^{-5}	10^{-6}	10^{-5}	1120
	SINDy validation	10^{-5}	10^{-7}	10^{-6}	1600
	PDE guidance	10^{-5}	10^{-5}	10^{-6}	480
Navier-Stokes	Data generation	10^{-8}	10^{-9}	10^{-10}	100000
	SINDy training	10^{-5}	10^{-6}	10^{-5}	1120
	SINDy validation	10^{-5}	10^{-7}	10^{-6}	1600
	PDE guidance	10^{-5}	10^{-5}	10^{-6}	480

where $x, y \in \Omega$. We set $N = 3$. The boundary conditions are periodic.

We use a spatial resolution of 64 points per dimension and a temporal resolution of 16 time points. The spatial domain is the unit square $\Omega = [0,1] \times [0,1]$, and the temporal domain is the interval $[0, 0.1]$. We generate 1000 samples for training the model with 100 samples in validation and test sets. As the simulation method, we use the method of lines with the Dormand-Prince 8/7 method as the ODE solver (Prince and Dormand, 1981). It is an explicit Runge-Kutta method.

A.3 Navier-Stokes equations

The Navier-Stokes equations are defined as the dynamical evolution of the velocity of a fluid. We simulate data according to the Navier–Stokes momentum equation with uniform shear and bulk viscosities given as

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial v}{\partial y} + \left(\frac{4}{3}\nu + \xi\right)\frac{\partial^2 u}{\partial x^2} + \nu\frac{\partial^2 u}{\partial x^2} + \left(\frac{1}{3}\nu + \xi\right)\frac{\partial^2 u}{\partial x \partial y} + \frac{1}{\rho} + a \tag{27}$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial v}{\partial y} + \nu\frac{\partial^2 u}{\partial x^2} + \left(\frac{4}{3}\nu + \xi\right)\frac{\partial^2 u}{\partial x^2} + \left(\frac{1}{3}\nu + \xi\right)\frac{\partial^2 u}{\partial x \partial y} + \frac{1}{\rho} + b, \tag{28}$$

where u, v are functions of $(x, y) \in \Omega$ and $t \in (0, T]$. We set the shear kinematic viscosity $\nu = 0.01$, the bulk kinematic viscosity $\xi = 0.1$, density $\rho = 2.0$ and acceleration components $a = -2.5$ and $b = 2.5$. Additionally, we have set the pressure field $p(x, y) = x + y$, which gives $\nabla p = 1$. The initial data are given as $u(x, y, 0) = v(x, y, 0) = h_0(x, y)$, where h_0 is defined in (24).

We use a spatial resolution of 64 points per dimension and a temporal resolution of 16 time points. The spatial domain is the unit square $\Omega = [0,1] \times [0,1]$ and the temporal domain is the interval $T = [0, 0.1]$. We generate 1000 samples for training, and 100 samples in both validation and test sets. As in the generation of SWE data, we use the Dormand-Prince 8/7 method as the solver of the spatially discretized PDE.

B Model specifications

B.1 SINDy Model

Training Alongside LASSO regression, we find that applying thresholding steps similar to the Sequentially Thresholded Least Squares (STLSQ) algorithm is beneficial for convergence (Brunton et al., 2016). We apply thresholding after each epoch. In each thresholding step, we clamp the absolutely smallest weight, that is smaller than the threshold, to zero. We start applying thresholding only after a specified number of epochs. We find that this strategy is sufficient for convergence on Burgers' data and set the LASSO regularization strength to zero.

We optimize (3) using the Adam optimizer with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, as implemented in the `optax` library (DeepMind et al., 2020). For Burgers' equation and Navier-Stokes, we apply a

Table 4: Learning rate schedules for SINDy model training.

Data	Schedule	Steps	Initial LR	Final LR
Burgers' Equation	Linear	200	0.00001	0.00005
	Constant	200	0.00005	0.00005
	Linear	500	0.00005	0.005
SWE	Linear	100	0.00001	0.01
	Constant	100	0.01	0.01
	Linear	1000	0.01	0.1
Navier-Stokes	Linear	100	0.00001	0.01
	Constant	100	0.01	0.01
	Linear	300	0.01	0.1
	Cosine decay	500	0.1	0

Table 5: Hyperparameters for SINDy model training.

	Burgers' Equation	SWE	Navier-Stokes
LASSO regularization strength	0.0	0.06	0.06
Batch size	100	100	100
Maximum epochs	800	200	100
Min success ratio	0.1	0.1	0.1
Threshold cutoff	0.001	0.0005	10^{-5}
Thresholding activation epoch	10	30	30
2-norm of initial weights	-	0.1	0.1
Maximum relative gradient norm	10.0	0.1	0.01

piecewise linear increasing learning rate schedule. For SWE, we additionally apply cosine learning rate decay. Learning rate schedules are specified in Table 4.

For initializing the SINDy weights θ for training on Burgers' data, we discretize the PDE also on the temporal dimension and minimize the PDE residual. This is implemented using the `PySINDy` library, where we use the STLSQ optimizer with thresholding cutoff 10^{-5}.

We implement adaptive gradient clipping. Training is performed using the `JAX` library (Bradbury et al., 2018). Training hyperparameters for each dataset are specified in Table 5.

B.2 Diffusion Model

Parametrization The parametrization of the score network follows the work of Karras et al. (2022). They set $\sigma(t) = t$ and parametrize the score network as

$$s_\psi(\mathbf{u}_\sigma, \sigma) = \frac{D_\psi(\mathbf{u}_\sigma, \sigma) - \mathbf{u}_\sigma}{\sigma^2}, \tag{29}$$

$$D_\psi(\mathbf{u}_\sigma, \sigma) = c_{\text{skip}}(\sigma)\mathbf{u}_\sigma + c_{\text{out}}(\sigma)F_\psi(c_{\text{in}}(\sigma)\mathbf{u}_\sigma; c_{\text{noise}}(\sigma)), \tag{30}$$

where F_ψ is a U-Net with self-attention (Ho et al., 2020). Due to the large memory footprint of self-attention, we do not use self-attention on the two largest resolution levels in the architecture. The scale parameters c are fixed to the values in the original work on the EDM.

In the experiments on Shallow Water Equations and Navier-Stokes, we do not downsample on the temporal axis, and following the work of Ho et al. (2022), we factorize attention into temporal and spatial components. Additionally, we use relative positional encoding in the temporal attention block (Shaw et al., 2018). Similarly to the work of Wei et al. (2024), we use 3x3x3 kernels for convolutional layers.

Training The objective function is

$$\mathcal{L} = \mathbb{E}_{\sigma\sim p_{\text{train}}, \mathbf{u}\sim p_{\text{data}}, \epsilon\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})}\left[\lambda(\sigma)\|D_\psi(\mathbf{u}+\epsilon, \sigma)\|_2^2\right], \tag{31}$$

where the loss weighting $\lambda(\sigma)$ and noise scale distibution p_{train} are fixed to the ones in the original work on the EDM. The expectation is approximated with Monte Carlo. We minimize (31) using the

Table 6: Hyperparameters for the architecture and training of the diffusion model.

	Burgers' Equation	SWE	Navier-Stokes
Nof resolution levels	3	3	3
Time embed dimension	64	48	48
Nof residual blocks per encoder layer	1	2	2
Base channels	64	64	64
Batch size	24	24	24
Initial earning rate	0.002	0.002	0.001
Final learning rate	0.0004	10^{-5}	10^{-5}
Max epochs and decay steps	2000	300	300
Early stopping patience	100	40	20
Early stopping tolerance	10^{-5}	10^{-6}	10^{-5}
Maximum relative gradient norm	0.2	0.2	0.2

Adam optimizer with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Training is performed in the `PyTorch` framework (Paszke et al., 2019). To enforce training stability, we also use adaptive gradient clipping. Additionally, we employ early stopping. The hyperparameters of architecture and training of the diffusion model are specified in Table 6.

Inference The reference distribution $p_T = \mathcal{N}(0, T)$. Following the work of Karras et al. (2022), we set $T = \sigma_0 = 80$, $\sigma_{N-1} = 0.002$ and $\rho = 7$, and use Heun's method to solve the reverse process using the discretization given by

$$\sigma_i = \begin{cases} (\sigma_{N-1}^{\frac{1}{\rho}} + \frac{i}{N-1}(\sigma_{N-1}^{\frac{1}{\rho}} - \sigma_0^{\frac{1}{\rho}}))^{\rho}, & i = 1, ..., N-2 \\ 0, & i = N, \end{cases} \tag{32}$$

where N is the amount of diffusion steps taken. We use the guidance schedules

$$\gamma_{\text{OBS},t} = \frac{\gamma_{\text{OBS}}}{|\mathcal{C}|t}$$
$$\gamma_{\text{PDE},t} = \frac{\gamma_{\text{PDE}}}{|\mathbf{u}_t|t},$$

where $|\mathcal{C}|$ is the number of context points and $|\mathbf{u}_t|$ is the number of points in the spatiotemporal field tensor. Context guidance is applied along the whole backwards diffusion trajectory. However, PDE guidance is only be applied to a part of the trajectory. We optimize the guidance strengths γ_{OBS} and γ_{PDE}, and the interval of application of PDE guidance via Bayesian optimization implemented in `Weights&Biases` and `bayesian-optimization`.

We use $N = 1000$ sampling steps for Burgers' equation and $N = 200$ sampling steps for other datasets. Other hyperparameters for generation are given in Table 7.

Table 7: Hyperparameters for generation on each dataset.

Data	Model	γ_{OBS}	γ_{PDE}	PDE guid. start	PDE guid. end
Burgers' Equation	D+C	9988921	0	-	-
	D+C+Pd	8760530	10	65.897	0.076
	D+C+S	19402400	18490400	76.989	0.022
	D+C+P	14890700	67482000	56.527	0.009
SWE	D+C	92146980	0	-	-
	D+C+Pd	19626960	58	75.712	0.024
	D+C+S	47934140	8032200	69.651	0.014
	D+C+P	40208310	16327160	67.725	0.002
Navier-Stokes	D+C	48836900	0	-	-
	D+C+Pd	86436120	20	40.066	0.086
	D+C+S	86932290	86742110	49.372	0.003
	D+C+P	80064020	142059660	75.712	0.002