Solving Inverse Physics Problems with Score Matching

Benjamin Holzschuh [1 2] **Simona Vegetti** [2] **Nils Thuerey** [1]

Abstract

We propose to solve inverse problems involving the temporal evolution of physics systems by leveraging recent advances from diffusion models. Our method moves the system's current state backward in time step by step by combining an approximate inverse physics simulator and a learned correction function. Training the learned correction with a single-step loss is equivalent to a score matching objective, while recursively predicting longer parts of the trajectory during training relates to maximum likelihood training of a corresponding probability flow. Our resulting inverse solver has excellent accuracy and temporal stability and, in contrast to other learned inverse solvers, allows for sampling the posterior of the solutions.

1. Introduction

We target inverse problems to reconstruct the distribution of initial states for a given end state of a physics system. This problem is genuinely tough (Zhou et al., 1996; Gómez-Bombarelli et al., 2018), and existing methods lack tractable approaches to represent and sample the distribution of states. Our method builds on recent advances from the field of diffusion-based approaches (Sohl-Dickstein et al., 2015; Ho et al., 2020): Based on a given end state of the system, we predict a previous state by taking a small time step backward in time and repeating this multiple times for a single inference. The prediction of the previous state depends on an approximate inverse of the dynamics, a learned update s_θ, and a small Gaussian perturbation.

In contrast to previous diffusion methods, we include domain knowledge about the physics process in the form of an approximate inverse simulator, which replaces the drift term of diffusion models (Song et al., 2021; Zhang & Chen, 2021). The learned component s_θ corrects any errors that occur due to numerical issues, e.g., the time discretization, and breaks ambiguities due to a loss of information in the simulation over time. Compared to other strategies for conditional sampling and diffusion models, such as classifier and classifier-free guidance, our method directly embeds the diffusion into the physics process. All conditioning information that we consider here is thus already contained in the state of the process.

The training of s_θ is similar to learned correction approaches for numerical simulations (Tompson et al., 2017; Um et al., 2020; Kochkov et al., 2021); however, in our method, we target to learn corrections for the "reverse" simulation. Training can either be based on single simulation steps or be extended to rollouts for multiple steps. The latter requires the differentiability of the inverse physics step. While the training with single steps directly minimizes a score matching objective, we show that the extension to multiple steps corresponds to maximum likelihood training of a related neural ordinary differential equation (ODE). Considering multiple steps is important for the stability of the produced trajectories. Feedback from physics and neural network interactions at training time leads to more robust results.

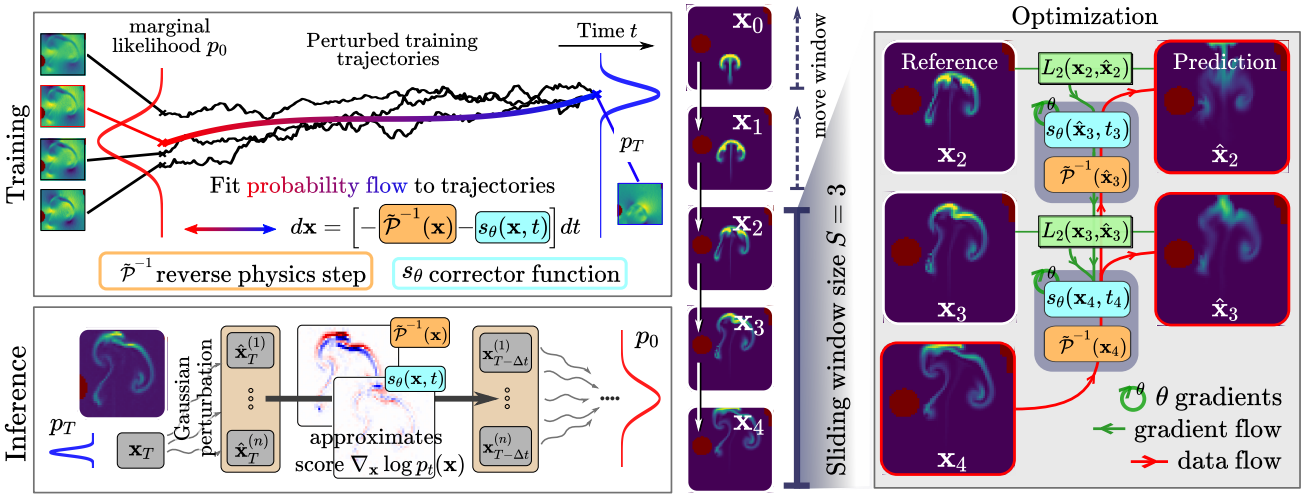
Figure 1 gives an overview of our method. Our aim is not to develop a new, generic algorithm for diffusion models but rather to provide a first demonstration that the combination of diffusion-based techniques and differentiable simulations has merit for solving inverse problems. In the following, we refer to methods using this combination as *score matching via differentiable physics* (SMDP). Our main contributions are: (1) We introduce a reverse physics simulation step into diffusion models to develop a probabilistic framework for solving inverse problems. (2) We provide the theoretical foundation that this combination yields learned corrections representing the score of the underlying data distribution. (3) We highlight the effectiveness of SMDP with two challenging inverse problems.

2. Method Overview

Problem formulation Consider a dataset of trajectories $(\mathbf{x}_0, ..., \mathbf{x}_M)$, where each $\mathbf{x}_j \in \mathbb{R}^D$ corresponds to a discretized state of a numerical simulation at time t_j. $t_j - t_k := \Delta t$ denotes the timestep, and $t_0 = 0$ and

[1]Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Germany [2]Max Planck Institute for Astrophysics, Karl-Schwarzschild-Straße 1, 85748 Garching, Germany. Correspondence to: Benjamin Holzschuh <benjamin.holzschuh@tum.de>.

Accepted to ICML workshop on Structured Probabilistic Inference & Generative Modeling.



(a) Training and inference overview.

(b) Sliding window and optimizing s_θ.

Figure 1: Overview of our method. For training, we fit a neural ODE, the probability flow, to the set of perturbed training trajectories (a, top). The probability flow is comprised of an approximate reverse physics simulator $\tilde{\mathcal{P}}^{-1}$ that contrary to $\mathcal{P}$ moves the state of the system backward in time as well as a correction function s_θ. For inference, we simulate the system backward in time from $\mathbf{x}_T$ to $\mathbf{x}_0$ by combining $\tilde{\mathcal{P}}^{-1}$, the trained s_θ and Gaussian noise in each step (a, bottom). For optimizing s_θ, our approach moves a sliding window of size S along the training trajectories and reconstructs the current window (b). Gradients for θ are accumulated and backpropagated through all prediction steps.

$t_M = T$. Moreover, we assume that the temporal evolution of our system can be approximated recursively by $\mathbf{x}_{m+1} = \mathbf{x}_m + \Delta t\, \mathcal{P}(\mathbf{x}_m) + \sqrt{\Delta t}\, F_{t_m}$, where $\mathcal{P} : \mathbb{R}^D \to \mathbb{R}^D$ is a physics simulator and F_{t_m} a stochastic forcing that accounts for uncertainties and perturbations. We draw the initial state $\mathbf{x}_0$ from a distribution p_0. This then induces a distribution of states p_{t_i} for all time steps $0 < i < M$. Our goal is to infer an initial state $\mathbf{x}_0$ given a simulation end state $\mathbf{x}_M$, i.e. we want to sample from the distribution $p_0(\cdot \,|\, \mathbf{x}_M)$. In the first experiment, we consider a Gaussian stochastic forcing, i.e., the system is described by the SDE $d\mathbf{x} = \mathcal{P}(\mathbf{x})dt + g(t)dW$ for a function $g : \mathbb{R} \to \mathbb{R}$, whereas in the second experiment, there is no stochastic forcing but small noise added to the end state.

2.1. Learned Corrections for Reverse Simulation

In the following, we furthermore assume that we have access to an approximate reverse physics simulator $\tilde{\mathcal{P}}^{-1} : \mathbb{R}^D \to \mathbb{R}^D$, which moves the simulation state backward in time (Holl et al., 2022). We train a neural network $s_\theta(\mathbf{x}, t)$ parameterized by θ such that

$$\mathbf{x}_m \approx \mathbf{x}_{m+1} + \Delta t \left[\tilde{\mathcal{P}}^{-1}(\mathbf{x}_{m+1}) + s_\theta(\mathbf{x}_{m+1}, t_{m+1})\right]. \quad (1)$$

Multi-step loss We define a hyperparameter S, called sliding window size, and write $\mathbf{x}_{i:i+S} \in \mathbb{R}^{S \times D}$ to denote the trajectory starting at $\mathbf{x}_i$ that is comprised of $\mathbf{x}_i$ and the following $S - 1$ states. Then, we define the **multi-step loss**

$$\mathcal{L}(\theta) = \frac{1}{M}\, \mathbb{E}\left[\sum_{m=0}^{M-S+1} ||\mathbf{x}_{m:m+S-1} - \hat{\mathbf{x}}_{m:m+S-1}||_2^2\right],$$

where the expectation is computed by drawing a trajectory $\mathbf{x}_{0:M}$ from the training data set. The predicted trajectories $\hat{\mathbf{x}}_{m:m+S-1}$ are defined recursively by $\hat{\mathbf{x}}_{i+S} = \mathbf{x}_{i+S}$ and computing $\hat{\mathbf{x}}_{i+S-1-j}$ from $\hat{\mathbf{x}}_{i+S-j}$ based on equation (1). For $S = 2$, we denote the corresponding loss **1-step loss**, since the prediction only comprises a single previous simulation state instead of a longer trajectory of states. Conceptually, the 1-step loss is similar to the training of standard diffusion models. However, in practice, approaches that consider a loss based on predicting longer parts of the trajectories are more successful for training learned corrections (Bar-Sinai et al., 2019; Um et al., 2020; Kochkov et al., 2021). In appendix A we show how this loss relates to matching the score $\nabla_\mathbf{x} \log p_t(\mathbf{x})$ and maximum likelihood training. In our experiments, we leverage the forward physics simulator $\mathcal{P}$ to obtain an approximation for the reverse step $\tilde{\mathcal{P}}^{-1}$. It is also possible replace $\tilde{\mathcal{P}}^{-1}$ by a learned surrogate model.

2.2. Training and Inference

We start training s_θ with the multi-step loss and window size $S = 2$ and gradually increase the window size S until a maximum $S_{\max}$. For $S > 2$, the unrolling of the predicted trajectory includes interactions between s_θ and the reverse

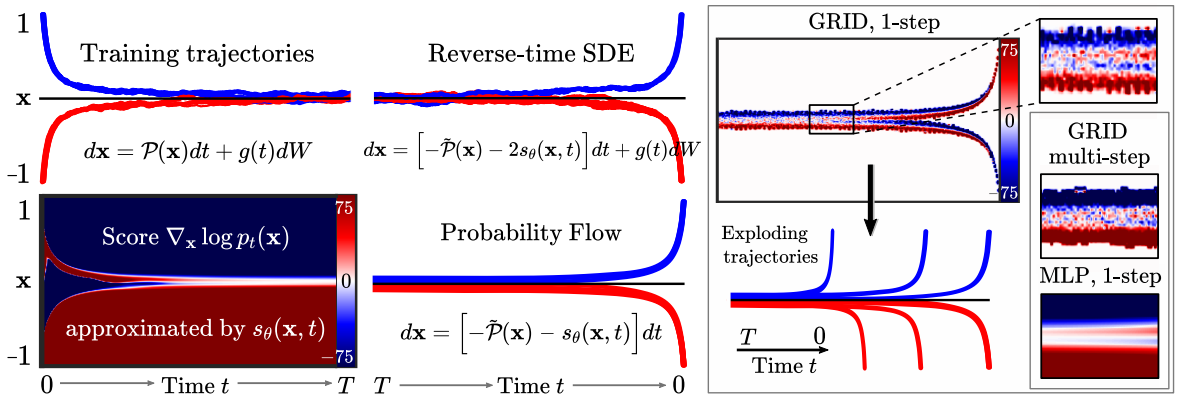


Figure 2: Overview of our 1D toy SDE. (a) Training with a data set of trajectories and known temporal dynamics given by $\mathcal{P}(\mathbf{x}) := -\text{sign}(\mathbf{x})\mathbf{x}^2$ and $g \equiv 0.1$. We estimate the score $\nabla_\mathbf{x} \log p_t(\mathbf{x})$ with our proposed method using an MLP network for $s_\theta(\mathbf{x}, t)$. Negative values (blue) push down the trajectories, and positive ones (red) push them up. Together with the dynamics, this can be used to reverse the system as shown in (b) either with the reverse-time SDE or the probability flow ODE. A successful inversion of the dynamics requires the network s_θ to be robust and extrapolate well (c). Inference using GRID trained with the 1-step loss causes trajectories to explode. Training GRID with the multi-step loss solves this issue.

physics step $\tilde{\mathcal{P}}^{-1}$. For inference, we consider the neural SDE

$$d\mathbf{x} = \left[-\tilde{\mathcal{P}}^{-1}(\mathbf{x}) - C\, s_\theta(\mathbf{x}, t)\right] dt + g(t)dW,$$

which we solve via the Euler-Maruyama method. Sampling from this SDE yields a posterior distribution. For $C = 2$ and if the system is described by an SDE, this corresponds to its reverse-time SDE (Anderson, 1982). Setting $C = 1$ and excluding the noise gives the probability flow ODE, which constitutes a unique solution. We denote the ODE variant by *SMDP ODE* and the SDE variant by *SMDP SDE*.

2.3. 1D Toy SDE

As the first experiment we consider a simple quadratic SDE of the form: $dx = -\left[\lambda_1 \cdot \text{sign}(x)x^2\right] dt + \lambda_2 dW$, with $\lambda_1 = 7$ and $\lambda_2 = 0.03$. Throughout this experiment, p_0 is a categorical distribution, where we draw either 1 or -1 with the same probability. The reverse-time SDE that transforms the distribution p_T of values at $T = 10$ to p_0 is given by

$$dx = -\left[\lambda_1 \cdot \text{sign}(x)x^2 - \lambda_2^2 \cdot \nabla_x \log p_t(x)\right] dt + \lambda_2 dw.$$

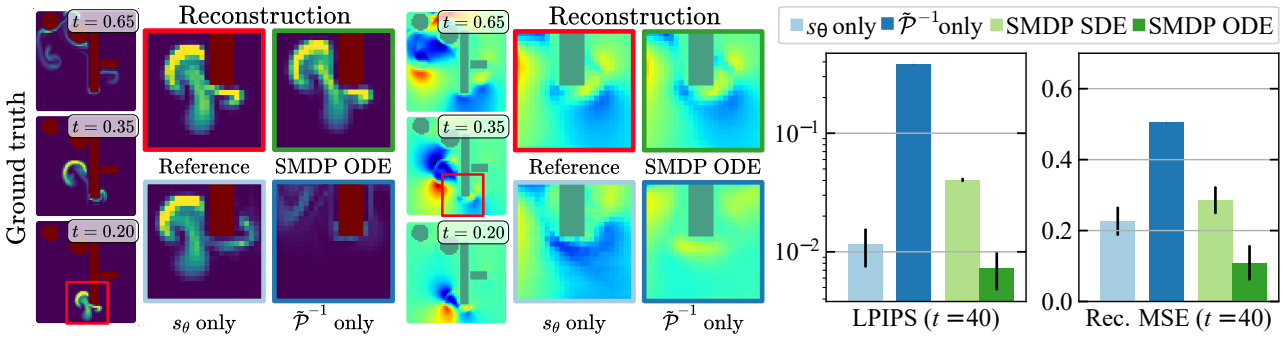
In figure 2a, we show paths from this SDE simulated with the Euler-Maruyama method. The trajectories approach 0 as t increases. Given the trajectory value at $t = 10$, it is no longer possible to infer the origin of the trajectory at $t = 0$.

This experiment allows us to use an analytic reverse simulator: $\tilde{\mathcal{P}}^{-1}(x) = \lambda_1 \cdot \text{sign}(x)x^2$. This is a challenging problem because the reverse physics step increases quadratically with x, and s_θ has to control the reverse process accurately to stay within the training domain, or paths will explode to infinity. We evaluate each model based on how well the predicted trajectories $\hat{x}_{0:T}$ match the posterior distribution. When drawing $\hat{x}_T$ randomly from $[-0.1, 0.1]$, we should obtain trajectories with $\hat{x}_0$ being either -1 or 1 with the same likelihood. We assign the label -1 or 1 if the relative distance of an endpoint is $< 10\%$ and denote the percentage in each class by p_{-1} and p_1. As some trajectories miss the target, typically $p_{-1} + p_1 < 1$. Hence, we define the posterior metric Q as twice the minimum of p_{-1} and p_1, i.e., $Q = 2 \cdot \min(p_{-1}, p_1)$ so that values closer to one indicate a better match of the correct posterior distribution.

Training The training data set consists of 2.500 simulated trajectories from 0 to T and $\Delta t = 0.02$. For the network $s_\theta(x, t)$, we consider a multilayer perceptron (MLP) and, as a special case, a grid-based discretization (GRID). For GRID, we discretize the domain $[0, T] \times [-1.25, 1.25]$ and linearly interpolate the solution. The cell centers are initialized with 0. We evaluate s_θ trained via the 1-step and multi-step losses with $S_{\max} = 10$. Details of hyperparameters and model architectures are given in appendix B.

Method	Probability flow ODE			Reverse-time SDE		
	Dataset size			Dataset size		
	100%	10%	1%	100%	10%	1%
multi-step	**0.97**	**0.91**	**0.81**	**0.99**	0.94	**0.85**
1-step	0.78	0.44	0.41	0.93	0.71	0.75
ISM	0.19	0.15	0.01	0.92	0.94	0.52

Table 1: Posterior metric Q for 1.000 predicted trajectories averaged over three runs.



(a) Marker density (b) Velocity field (x) (c) Evaluation

Figure 3: Buoyancy flow case. Ground truth shows the marker density and velocity field in the x-direction at different points of the simulation trajectory from the test set (a, b). We show reconstructions given the simulation end state at $t = 0.65$ and provide an evaluation of the reconstructed trajectories based on perceptual similarity (LPIPS) and the reconstruction MSE for three runs (c).

Better extrapolation and robustness from multi-step loss See figure 2c for an overview of the differences between the learned score from MLP and GRID and the effects of the multi-step loss. Training via GRID shows that most cells do not get any gradient updates and remain 0. This is caused by a need for more training data in these regions. In addition, the boundary of the trained region is jagged and diffuse. Trajectories traversing these regions can quickly explode. In contrast, the multi-step loss leads to a consistent signal around the center line at $x = 0$, effectively preventing exploding trajectories.

Evaluation and comparison with baselines As a baseline for learning the score, we consider implicit score matching (Hyvärinen, 2005, ISM). We train all methods with the same network architecture. As can be seen in table 1, our proposed multi-step training performs best or is on par with the baselines for all dataset sizes and inference types. For standard deviations, see appendix B.

2.4. Buoyancy-driven Flow with Obstacles

Next, we test our methodology on a more challenging problem. For this purpose, we consider deterministic simulations of buoyancy-driven flow within a fixed domain $\Omega \subset [0, 1] \times [0, 1]$ and randomly placed obstacles. We use semi-Lagrangian advection for the velocity and MacCormack advection for the hot marker density. The temperature dynamics of the marker field are modeled with a Boussinesq approximation. Each simulation runs from time $t = 0.0$ to $t = 0.65$ with a step size of $\Delta t = 0.01$. Our method is trained with the objective of reconstructing a plausible initial state given an end state of the marker density and velocity fields at time $t = 0.65$, as shown in figure 3a and figure 3b. We place spheres and boxes with varying sizes at different positions within the simulation domain that do not overlap with the marker inflow. For each simulation, we place one to two objects of each category.

Score matching for deterministic systems During training, we add Gaussian noise to each simulation state $\mathbf{x}_t$ with $\sigma_t = \sqrt{\Delta t}$. In this experiment, no stochastic forcing is used to create the data set, i.e., $g \equiv 0$. By adding noise to the simulation states, the 1-step loss still minimizes a score matching objective in this situation, similar to denoising score matching; see appendix A.3 for a derivation. In the situation without stochastic forcing, during inference, our method effectively alternates between the reverse physics step, a small perturbation, and the correction by $s_\theta(\mathbf{x}, t)$, which projects the perturbed simulation state back to the distribution p_t. For the SDE trajectories, $C = 2$ slightly overshoots, and $C = 1$ gives an improved performance.

Training and ablation study Our training data set consists of 250 simulations with corresponding trajectories generated with *phiflow* (Holl et al., 2020). Our neural network architecture for $s_\theta(\mathbf{x}, t)$ uses dilated convolutions (Stachenfeld et al., 2021). See appendix C for additional training details. The reverse physics step $\tilde{\mathcal{P}}^{-1}$ is implemented directly in the solver by using a negative step size $-\Delta t$ for time integration. For training, we consider the multi-step formulation with $S_{\max} = 20$. For the evaluation, we consider a reconstruction MSE and the perceptual similarity metric LPIPS. The test set contains five simulations. The SDE version yields good results for this experiment but is most likely constrained in performance by the approximate reverse physics step and large step sizes. However, the ODE version significantly outperforms directly inverting the simulation numerically ($\tilde{\mathcal{P}}^{-1}$ only), and when training without the reverse physics step (s_θ only), as shown in 3c.

3. Conclusion

We presented a combination of learned corrections training and diffusion models in the context of physical simulations and differentiable physics for solving inverse physics problems. We showed its competitiveness, accuracy, and long-term stability in two challenging and versatile experiments and motivated our design choices. We considered two variants with complementary benefits for inference: while the ODE variants achieve the best MSE, the SDE variants allow for sampling the posterior and yield an improved coverage of the target data manifold.

References

Anderson, B. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Bar-Sinai, Y., Hoyer, S., Hickey, J., and Brenner, M. P. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Holl, P., Koltun, V., Um, K., and Thuerey, N. phiflow: A differentiable pde solving framework for deep learning via physical simulations. In *NeurIPS Workshop*, volume 2, 2020.

Holl, P., Koltun, V., and Thuerey, N. Scale-invariant learning by physics inversion. *Advances in Neural Information Processing Systems*, 35:5390–5403, 2022.

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res.*, 6:695–709, 2005. URL http://jmlr.org/papers/v6/hyvarinen05a.html.

Johnson, N. L., Kotz, S., and Balakrishnan, N. *Continuous univariate distributions, volume 2*, volume 289. John wiley & sons, 1995.

Kersting, H., Krämer, N., Schiegg, M., Daniel, C., Tiemann, M., and Hennig, P. Differentiable likelihoods for fast inversion of 'likelihood-free' dynamical systems. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5198–5208. PMLR, 2020. URL http://proceedings.mlr.press/v119/kersting20a.html.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Kloeden, P. E., Platen, E., Kloeden, P. E., and Platen, E. *Stochastic differential equations*. Springer, 1992.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning accelerated computational fluid dynamics. *CoRR*, abs/2102.01010, 2021. URL https://arxiv.org/abs/2102.01010.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2256–2265. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/sohl-dickstein15.html.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Stachenfeld, K., Fielding, D. B., Kochkov, D., Cranmer, M., Pfaff, T., Godwin, J., Cui, C., Ho, S., Battaglia, P., and Sanchez-Gonzalez, A. Learned simulators for turbulence. In *International Conference on Learning Representations*, 2021.

Tompson, J., Schlachter, K., Sprechmann, P., and Perlin, K. Accelerating eulerian fluid simulation with convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=ByH2gxrKl.

Um, K., Brand, R., Fei, Y. R., Holl, P., and Thuerey, N. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/43e4e6a6f341e00671e123714de019a8-Abstract.html.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural Comput.*, 23(7):1661–1674, 2011. doi: 10.1162/NECO_a_00142. URL https://doi.org/10.1162/NECO_a_00142.

Zhang, Q. and Chen, Y. Diffusion normalizing flow. *Advances in Neural Information Processing Systems*, 34: 16280–16291, 2021.

Zhou, K., Doyle, J., and Glover, K. Robust and optimal control. *Control Engineering Practice*, 4(8):1189–1190, 1996.

A. Proofs and Training Methodology

Below we summarize the problem formulation from the main paper and provide details about the training procedure and the derivation of our methodology.

Problem setting Let $(\Omega, \mathcal{F}, P)$ be a probability space and $W(t) = (W_1(t), ..., W_D(t))^T$ be a D-dimensional Brownian motion. Moreover, let $\mathbf{x}_0$ be a $\mathcal{F}_0$-measurable $\mathbb{R}^D$-valued random variable that is distributed as p_0 and represents the initial simulation state. We consider the time evolution of the physical system for $0 \leq t \leq T$ modeled by the stochastic differential equation (SDE)

$$d\mathbf{x} = \mathcal{P}(\mathbf{x})dt + g(t)dW \tag{2}$$

with initial value $\mathbf{x}_0$ and Borel measurable drift $\mathcal{P} : \mathbb{R}^D \to \mathbb{R}^D$ and diffusion $g : [0, T] \to \mathbb{R}^+$. This SDE transforms the marginal distribution p_0 of initial states at time 0 to the marginal distribution p_T of end states at time T.

Moreover, we assume that we have sampled N trajectories of length M from the above SDE with a fixed time discretization $0 \leq t_0 < t_1 < ... < t_M \leq T$ for the interval $[0, T]$ and collected them in a training data set $\{(\mathbf{x}_{t_i}^{(n)})_{i=0}^M\}_{n=0}^N$. For simplicity, we assume that all time steps are equally spaced, i.e., $t_{i+1} - t_i := \Delta t$. Moreover, in the following we use the notation $\mathbf{x}_{i:j}$ for $0 \leq i < j \leq M$ to refer to the trajectory $(\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+1}}, ..., \mathbf{x}_{t_j})$.

Assumptions Throughout this paper, we make some additional assumptions to ensure the existence of a unique solution to the SDE (2) and the strong convergence of the Euler-Maruyama method. In particular:

- Finite variance of samples from p_0: $\mathbb{E}_{\mathbf{x}_0 \sim p_0}[||\mathbf{x}_0||_2^2] < \infty$
- Lipschitz continuity of $\mathcal{P}$: $\exists K_1 > 0 \, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^D : ||\mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{y})||_2 \leq K_1 ||\mathbf{x} - \mathbf{y}||_2$
- Lipschitz continuity of g: $\exists K_2 > 0 \, \forall t, s \in [0, T] : |g(t) - g(s)| \leq K_3 |t - s|$
- Linear growth condition: $\exists K_3 > 0 \, \forall \mathbf{x} \in \mathbb{R}^D : ||\mathcal{P}(\mathbf{x})||_2 \leq K_3(1 + ||\mathbf{x}||_2)$
- g is bounded: $\exists K_4 > 0 \, \forall t \in [0, T] : |g(t)| \leq K_4$

Euler-Maruyama Method Using Euler-Maruyama steps, we can simulate paths from SDE (2) similar to ordinary differential equations (ODE). Given an initial state $\mathbf{X}_{t_0}$, we let $\hat{\mathbf{X}}_{t_0}^{\Delta t} = \mathbf{X}_{t_0}$ and define recursively

$$\hat{\mathbf{X}}_{t_{m+1}}^{\Delta t} \leftarrow \hat{\mathbf{X}}_{t_m}^{\Delta t} + \Delta t \, \mathcal{P}(\hat{\mathbf{X}}_{t_m}^{\Delta t}) + \sqrt{\Delta t} \, g(t_m) \, z_{t_m}, \tag{3}$$

where z_{t_m} are i.i.d. with $z_{t_m} \sim \mathcal{N}(0, I)$. For $t_i \leq t < t_{i+1}$, we define the piecewise constant solution of the Euler-Maruyama Method as $\hat{\mathbf{X}}_t^{\Delta t} := \hat{\mathbf{X}}_{t_i}^{\Delta t}$. Let $\mathbf{X}_t$ denote the solution of the SDE (2). Then the Euler-Maruyama solution $\hat{\mathbf{X}}_t^{\Delta t}$ converges strongly to $\mathbf{X}_t$.

Lemma A.1. *[Strong convergence of Euler-Maruyama method] Consider the piecewise constant solution $\hat{X}_t^{\Delta t}$ of the Euler-Maruyama method. There is a constant C such that*

$$\sup_{0 \leq t \leq T} \mathbb{E}[||X_t - \hat{X}_t^{\Delta t}||_2] \leq C\sqrt{\Delta t}. \tag{4}$$

Proof. See Kloeden et al. (1992, 10.2.2) □

A.1. 1-step Loss and Score Matching Objective

Theorem A.2. *Consider a data set $\{\mathbf{x}_{0:m}^{(n)}\}_{n=1}^N$ with trajectories sampled from SDE (2). Then the 1-step loss*

$$\mathcal{L}_{\text{single}}(\theta) := \frac{1}{M} \mathbb{E}_{\mathbf{x}_{0:M}} \left[\sum_{m=0}^{M-1} \left[||\mathbf{x}_m + \Delta t \left[-\mathcal{P}(\mathbf{x}_{m+1}) + s_\theta(\mathbf{x}_{m+1}, t_{m+1}) \right]||^2 \right] \right] \tag{5}$$

is equivalent to minimizing the score matching objective

$$\mathcal{J}(\theta) := \int_0^T \mathbb{E}_{\mathbf{x}_t}[||\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \tilde{s}_\theta(\mathbf{x}, t)||_2^2]dt, \tag{6}$$

where $\tilde{s}_\theta(\mathbf{x}, t) = s_\theta(\mathbf{x}, t)/g^2(t)$ *as* $\Delta t \to 0$.

Proof.

"⇐": Consider θ^* such that $\tilde{s}_{\theta^*}(\mathbf{x}, t) \equiv \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, which minimizes the score matching objective $\mathcal{J}(\theta)$. Then fix a time step t and sample $\mathbf{x}_t$ and $\mathbf{x}_{t+\Delta t}$ from the data set. The probability flow solution $\mathbf{x}_t^p$ based on equation (5) is

$$\mathbf{x}_t^p := \mathbf{x}_{t+\Delta t} + \Delta t \left[-\mathcal{P}(\mathbf{x}_{t+\Delta t}) + s_\theta(\mathbf{x}_{t+\Delta t}, t + \Delta t)\right]. \tag{7}$$

At the same time, we know that the transformation of marginal likelihoods from $p_{t+\Delta t}$ to p_t follows the reverse-time SDE (Anderson, 1982)

$$d\mathbf{x} = \left[\mathcal{P}(\mathbf{x}) + g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt + g(t)dW, \tag{8}$$

which runs backward in time from T to 0. Denote by $\hat{\mathbf{x}}_t^{\Delta t}$ the solution of the Euler-Maruyama method at time t initialized with $\mathbf{x}_{t+\Delta t}$ at time $t + \Delta t$.

Using the triangle inequality for squared norms, we can write

$$\lim_{\Delta t\to 0} \mathbb{E}\left[||\mathbf{x}_t - \mathbf{x}_t^p||_2^2\right] \leq 2 \lim_{\Delta t\to 0} \mathbb{E}\left[||\mathbf{x}_t - \hat{\mathbf{x}}_t^{\Delta t}||_2^2\right] + 2 \lim_{\Delta t\to 0} \mathbb{E}\left[||\hat{\mathbf{x}}_t^{\Delta t} - \mathbf{x}_t^p||_2^2\right]. \tag{9}$$

Because of the strong convergence of the Euler-Maruyama method, we have that for the first term of the bound in equation (9)

$$\lim_{\Delta t\to 0} \mathbb{E}\left[||\mathbf{x}_t - \hat{\mathbf{x}}_t^{\Delta t}||_2^2\right] = 0 \tag{10}$$

independent of θ. At the same time, for the Euler-Maruyama discretization, we can write

$$\hat{\mathbf{x}}_t^{\Delta t} = \mathbf{x}_{t+\Delta t} + \Delta t \left[-\mathcal{P}(\mathbf{x}_{t+\Delta t}) + g^2(t + \Delta t)\nabla_{\mathbf{x}} \log p_{t+\Delta t}(\mathbf{x}_{t+\Delta t})\right] + g(t + \Delta t)\sqrt{\Delta t} z_{t+\Delta t}, \tag{11}$$

where $z_{t+\Delta t}$ is a standard Gaussian distribution, i.e., $z_{t+\Delta t} \sim \mathcal{N}(0, I)$. Therefore, we can simplify the second term of the bound in equation (9)

$$\lim_{\Delta t\to 0} \mathbb{E}\left[\left|\left|\hat{\mathbf{x}}_t^{\Delta t} - \mathbf{x}_t^p\right|\right|_2^2\right] \tag{12}$$

$$= \lim_{\Delta t\to 0} \mathbb{E}_{\mathbf{x}_{t+\Delta t}\sim p_{t+\Delta t}, z\sim\mathcal{N}(0,I)}\left[\left|\left|\Delta t\, g(t + \Delta t)^2 \left[\nabla_{\mathbf{x}} \log p_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - \tilde{s}_\theta(\mathbf{x}_{t+\Delta t}, t + \Delta t)\right]\right.\right. \tag{13}$$

$$\left.\left. + g(t + \Delta t)\sqrt{\Delta t}\, z\right|\right|_2^2\Bigg]. \tag{14}$$

If θ^* minimizes the score matching objective, then $\tilde{s}_{\theta^*}(\mathbf{x}, t) \equiv \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, and therefore the above is the same as

$$\lim_{\Delta t\to 0} \mathbb{E}_z[||\sqrt{\Delta t}\, g(t + \Delta t)\, z||_2^2] = 0. \tag{15}$$

Combining equations (10) and (15) yields

$$\lim_{\Delta t\to 0} \mathbb{E}\left[||\mathbf{x}_t - \mathbf{x}_t^p||_2^2\right] = 0. \tag{16}$$

Additionally, since g is bounded, we even have

$$\mathbb{E}[||\sqrt{\Delta t}\, g(t + \Delta t)\, z||_2^2] \leq \mathbb{E}[||\sqrt{\Delta t}\, K_4\, z||_2^2] = \mathbb{E}[||K_4\, z||_2^2]\Delta t. \tag{17}$$

For $\mathcal{L}_{\text{single}}(\theta)$, using the above bound (17) and strong convergence of the Euler-Maruyama method, we can therefore derive the following bound

$$\mathcal{L}_{\text{single}}(\theta) = \frac{1}{M}\mathbb{E}\left[\sum_{m=0}^{M-1}\left[||\mathbf{x}_m + \Delta t\left[-\mathcal{P}(\mathbf{x}_{m+1}) + s_\theta(\mathbf{x}_{m+1}, t_{m+1})\right]||^2\right]\right] \tag{18}$$

$$\leq \frac{1}{M}\, M\, 2\, \left[C\sqrt{\Delta t} + \mathbb{E}_z[||K_4\, z||_2^2]\Delta t\right], \tag{19}$$

which implies that $\mathcal{L}_{\text{single}}(\theta) \to 0$ as $\Delta t \to 0$.

"⇒": With the definitions from "⇐", let θ^* denote a minimizer such that $\mathcal{L}_{\text{single}}(\theta) \to 0$ as $\Delta t \to 0$. This implies that each summand of $\mathcal{L}_{\text{single}}(\theta)$ also goes to zero as $\Delta t \to 0$, i.e., $\lim_{\Delta t\to 0}\mathbb{E}\left[||\mathbf{x}_t - \mathbf{x}_t^p||_2^2\right] = 0$. Note that at least one minimizer exists as we can choose $s_{\theta^*}(\mathbf{x}, t) \equiv \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. Again, with the triangle inequality for squared norms, we have that

$$\lim_{\Delta t\to 0}\mathbb{E}\left[||\hat{\mathbf{x}}_t^{\Delta t} - \mathbf{x}_t^p||_2^2\right] \leq 2\lim_{\Delta t\to 0}\mathbb{E}\left[||\mathbf{x}_t - \hat{\mathbf{x}}_t^{\Delta t}||_2^2\right] + 2\lim_{\Delta t\to 0}\mathbb{E}\left[||\mathbf{x}_t - \mathbf{x}_t^p||_2^2\right]. \tag{20}$$

By the strong convergence of the Euler-Maruyama method and $\theta = \theta^*$, we obtain

$$\lim_{\Delta t\to 0}\mathbb{E}\left[||\hat{\mathbf{x}}_t^{\Delta t} - \mathbf{x}_t^p||_2^2\right] = 0. \tag{21}$$

At the same time, for fixed $\Delta t > 0$, we can compute

$$\mathbb{E}\left[||\hat{\mathbf{x}}_t^{\Delta t} - \mathbf{x}_t^p||_2^2\right] \tag{22}$$

$$= \mathbb{E}_{\mathbf{x}_{t+\Delta t}, z\sim\mathcal{N}(0,I)}[||\Delta t\, g(t+\Delta t)^2\left[\nabla_{\mathbf{x}}\log p_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - s_\theta(\mathbf{x}_{t+\Delta t}, t+\Delta t)\right] + \sqrt{\Delta t}\, g(t+\Delta t)\, z||_2^2] \tag{23}$$

$$= \Delta t\, g(t+\Delta t)\,\mathbb{E}_{\mathbf{x}_{t+\Delta t}, z\sim\mathcal{N}(0,I)}[||\Delta t^{3/2}\, g(t+\Delta t)^{5/2}\left[\nabla_{\mathbf{x}}\log p_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - s_\theta(\mathbf{x}_{t+\Delta t}, t+\Delta t)\right] + z||_2^2]. \tag{24}$$

For fixed $\mathbf{x}_{t+\Delta t}$, the distribution over $z \sim \mathcal{N}(0, I)$ in equation (24) correspond to a noncentral chi-squared distribution (Johnson et al., 1995, Chapter 13.4), whose mean can be calculated as

$$\mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\left|\left|\Delta t^{3/2}\, g(t+\Delta t)^{5/2}\left[\nabla_{\mathbf{x}}\log p_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - s_\theta(\mathbf{x}_{t+\Delta t}, t+\Delta t)\right] + z\right|\right|_2^2\right] \tag{25}$$

$$= \left|\left|\Delta t^{3/2}\, g(t+\Delta t)^{5/2}\left[\nabla_{\mathbf{x}}\log p_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - s_\theta(\mathbf{x}_{t+\Delta t}, t+\Delta t)\right]\right|\right|_2^2 + D. \tag{26}$$

For each $\Delta t > 0$, the above is minimized if and only if $\nabla_{\mathbf{x}}\log p_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) = s_\theta(\mathbf{x}_{t+\Delta t}, t+\Delta t)$.

□

A.2. Multi-step Loss and Maximum Likelihood Training

We now extend the 1-step formulation from above to multiple steps and discuss its relation to maximum likelihood training. For this, we consider our proposed probability flow ODE defined by

$$d\mathbf{x} = \left[\mathcal{P}(\mathbf{x}) - s_\theta(\mathbf{x}, t)\right] dt \tag{27}$$

and for $t_i < t_j$ define $p_{t_i}^{t_j,\text{ODE}}$ as the distribution obtained by sampling $\mathbf{x} \sim p_{t_j}$ and integrating the probability flow with network $s_\theta(\mathbf{x}, t)$ equation (27) backward in time until t_i. We can choose two arbitrary time points t_i and t_j with $t_i < t_j$ because we do not require fixed start and end times of the simulation.

The maximum likelihood training objective of the probability flow ODE (27) can be written as maximizing

$$\mathbb{E}_{\mathbf{x}_{t_i}\sim p_{t_i}}[\log p_{t_i}^{p_{t_j},\text{ODE}}(\mathbf{x}_{t_i})]. \tag{28}$$

Our proposed multi-step loss is based on the sliding window size S, which is the length of the sub-trajectory that we aim to reconstruct with the probability flow ODE (27). The multi-step loss is defined as

$$\mathcal{L}_{\text{multi}}(\theta) := \frac{1}{M}\,\mathbb{E}_{\mathbf{x}_{0:M}}\left[\sum_{m=0}^{M-S+1}\left[||\mathbf{x}_{m:m+S-1} - \hat{\mathbf{x}}_{m:m+S-1}||_2^2\right]\right], \tag{29}$$

where we compute the expectation by sampling $\mathbf{x}_{0:m}$ from the training data set and $\hat{\mathbf{x}}_{i:i+S-1}$ is the predicted sub-trajectory that is defined recursively by

$$\hat{\mathbf{x}}_{i+S} = \mathbf{x}_{i+S} \quad \text{and} \quad \hat{\mathbf{x}}_{i+S-1-j} = \hat{\mathbf{x}}_{i+S-j} + \Delta t\left[-\mathcal{P}(\hat{\mathbf{x}}_{i+S-j}) + s_\theta(\hat{\mathbf{x}}_{i+S-j}, t_{i+S-j})\right]. \tag{30}$$

In the following, we show that the multi-step loss (29) maximizes a variational lower bound for the maximum likelihood training objective (28).

Theorem A.3. *Consider a data set $\{\mathbf{x}_{0:m}^{(n)}\}_{n=1}^N$ with trajectories sampled from SDE (2). Then the multi-step loss (29) maximizes a variational lower bound for the maximum likelihood training objective of the probability flow ODE (28) as $\Delta t \to 0$.*

Let $\mu_{t_i}^{\text{ODE}}(\mathbf{x}_{t_j})$ denote the solution of the probability flow ODE (27) integrated backward from time t_j to t_i with initial value $\mathbf{x}_{t_j}$.

For the maximum likelihood objective, we can derive a variational lower bound

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}_{t_i}}\left[\log p_{t_i}^{t_j,\,\text{ODE}}(\mathbf{x}_{t_i})\right] &= \mathbb{E}_{\mathbf{x}_{t_i}}\left[\log\left(\mathbb{E}_{\mathbf{x}_{t_j}}\left[p_{t_i}^{t_j,\,\text{ODE}}(\mathbf{x}_{t_i}|\mathbf{x}_{t_j})\right]\right)\right] && (31)\\
&= \mathbb{E}_{\mathbf{x}_{t_i}}\left[\log\left(\mathbb{E}_{\mathbf{x}_{t_j}|\mathbf{x}_{t_i}}\left[\frac{p_{t_i}(\mathbf{x}_{t_i})}{p_{t_j}(\mathbf{x}_{t_j}|\mathbf{x}_{\mathbf{t_i}})}p_{t_i}^{t_j,\,\text{ODE}}(\mathbf{x}_{t_i}|\mathbf{x}_{t_j})\right]\right)\right] && (32)\\
&\geq \mathbb{E}_{\mathbf{x}_{t_i}}\mathbb{E}_{\mathbf{x}_{t_j}|\mathbf{x}_{t_i}}\left[\log\left(\frac{p_{t_j}(\mathbf{x}_{t_j})}{p_{t_j}(\mathbf{x}_{t_j}|\mathbf{x}_{\mathbf{t_i}})}p_{t_i}^{t_j,\,\text{ODE}}(\mathbf{x}_{t_i}|\mathbf{x}_{t_j})\right)\right] && (33)\\
&= \mathbb{E}_{\mathbf{x}_{t_i}}\mathbb{E}_{\mathbf{x}_{t_j}|\mathbf{x}_{t_i}}\left[\log\left(\frac{p_{t_j}(\mathbf{x}_{t_j})}{p_{t_j}(\mathbf{x}_{t_j}|\mathbf{x}_{\mathbf{t_i}})}\right) + \log\left(p_{t_i}^{t_j,\,\text{ODE}}(\mathbf{x}_{t_i}|\mathbf{x}_{t_j})\right)\right], && (34)
\end{aligned}$$

where the inequality is due to Jensen's inequality. Since only $p_{t_i}^{t_j,\,\text{ODE}}(\mathbf{x}_{t_i}|\mathbf{x}_{t_j})$ depends on θ, this is the same as maximizing

$$\mathbb{E}_{\mathbf{x}_{t_i}}\mathbb{E}_{\mathbf{x}_{t_j}|\mathbf{x}_{t_i}}\left[\log\left(p_{t_i}^{t_j,\,\text{ODE}}(\mathbf{x}_{t_i}|\mathbf{x}_{t_j})\right)\right]. \tag{35}$$

The probability flow ODE is likelihood-free, which makes it challenging to optimize. Therefore, we relax the objective by perturbing the ODE distributions by convolving them with a Gaussian kernel G_ϵ with small $\epsilon > 0$, see, e.g., Kersting et al. (2020, Gaussian ODE filtering). This allows us to model the conditional distribution $p_{t_i}^{t_j,\,\text{ODE}}|\,\mathbf{x}_{t_j}$ as a Gaussian distribution with mean $\mu_{t_i}^{t_j,\,\text{ODE}}(\mathbf{x}_{t_j})$ and variance $\sigma^2 = \epsilon$. Then maximizing (35) reduces to matching the mean of the distribution, i.e., minimizing

$$\mathbb{E}_{\mathbf{x}_{t_i}}\mathbb{E}_{\mathbf{x}_{t_j}|\mathbf{x}_{t_i}}\left[||\mathbf{x}_{t_i} - \mu_{t_i}^{\text{ODE}}(\mathbf{x}_{t_j})||_2^2\right] \tag{36}$$

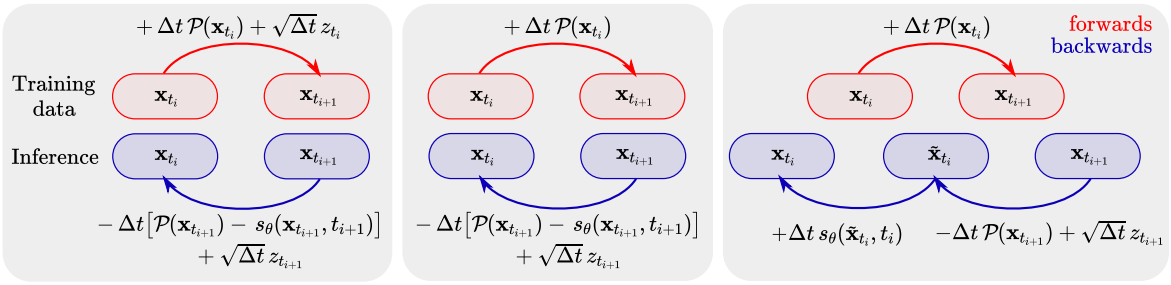
independent of $\epsilon > 0$. Since this is true for any time step $t_j > t_i$ and corresponding simulation state $\mathbf{x}_{t_j}$ given $\mathbf{x}_{t_i}$, we can pick the pairs $(\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+1}})$, $(\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+2}})$, $(\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+3}})$ and so on. Then, we can optimize them jointly by considering the sum of the individual objectives up to a maximum sliding window size

$$\mathbb{E}_{\mathbf{x}_{i:j}}\left[\sum_{k=i}^{j-1}||\mathbf{x}_{t_k} - \mu_{t_k}^{\text{ODE}}(\mathbf{x}_{t_j})||_2^2\right]. \tag{37}$$

As $\Delta t \to 0$, we compute the terms $\mu_{t_k}^{\text{ODE}}(\mathbf{x}_{t_j})$ on a single trajectory starting at $\mathbf{x}_{t_i}$ with sliding window S covering the trajectory until $\mathbf{x}_{t_j}$ via the Euler method, i.e., we can define recursively

$$\hat{\mathbf{x}}_{i+S} = \mathbf{x}_{i+S} \quad \text{and} \quad \hat{\mathbf{x}}_{i+S-1-j} = \hat{\mathbf{x}}_{i+S-j} + \Delta t\left[-\mathcal{P}(\hat{\mathbf{x}}_{i+S-j}) + s_\theta(\hat{\mathbf{x}}_{i+S-j}, t_{i+S-j})\right]. \tag{38}$$

By varying the starting points of the sliding window $\mathbf{x}_{t_i}$, this yields our proposed multi-step loss $\mathcal{L}_{\text{multi}}(\theta)$.



(a) SDE for training data and reverse-time SDE for inference (b) ODE for training data and SDE for inference (c) ODE for training data and physics step + denoising step for inference

Figure 4: Variants of training and inference for different physical systems. (a) shows the SDE and reverse-time SDE setup with the Euler-Maruyama discretization when the system is modeled by an SDE. The diffusion term $g(t)$ is absorbed in the Gaussian random variable $z_t \sim \mathcal{N}(0, g(t)^2 I)$ and network $s_\theta(\mathbf{x}, t)$. In (b), we assume that the temporal evolution of the training data is deterministic, i.e., we model the physical system without the diffusion term. However, for inference, we consider the reverse-time SDE of the same form as in (a), where the diffusion coefficient $g(t)$ is chosen as a hyperparameter that depends on the noise scale added to the data. Then, in (c), we split the Euler step for the backward direction into a physics-only update, adding the Gaussian noise z and a denoising step by $s_\theta(\mathbf{x}, t)$.

A.3. Denoising Score Matching for Deterministic Simulations

So far, we have considered physical systems that can be modeled by an SDE, i.e., equation (2). While this problem setup is suitable for many scenarios, we would also like to apply a similar methodology when the system is deterministic, i.e., when we can write the problem as an ordinary stochastic equation

$$d\mathbf{x} = \mathcal{P}(\mathbf{x})dt. \tag{39}$$

In the case of chaotic dynamical systems, this still represents a hard inverse problem, especially when information is lost due to noise added to the trajectories after their generation.

The training setup based on modeling the physics system with an SDE is shown in figure 4a. Figure 4b and 4c illustrate two additional data setup and inference variants for deterministic physical systems modeled by the ODE (39). While for the toy experiment in section 2.3 in the main paper, our setup resembles (a), for the buoyancy-driven flow in section 2.4 we consider (c) as the system is deterministic.

For this variant, the update by $-\mathcal{P}(\mathbf{x})$ and $s_\theta(\mathbf{x}, t)$ is separated into two steps. The temporal evolution from t_{i+1} to t_i is then defined entirely by physics. We apply an additive noise to the system and the update step by $s_\theta(\mathbf{x}, t)$, which can be interpreted as denoising for a now slightly perturbed state $\tilde{\mathbf{x}}_{t_i}$. In this case, we show that the network $s_\theta(\mathbf{x}, t)$ still learns the correct score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ during training using denoising score matching. We compare the performance of variants (b) and (c) for the buoyancy-drive flow in appendix C.

When separating physics and score updates, we calculate the updates as

$$\hat{\mathbf{x}}_{t_i} = \mathbf{x}_{t_{i+1}} - \Delta t\,\mathcal{P}(\mathbf{x}_{t_{i+1}}) \tag{40}$$

$$\hat{\mathbf{x}}^{\text{noise}}_{t_i} = \hat{\mathbf{x}}_{t_i} + \sqrt{\Delta t}\,g(t_i)\,z_{t_i} \tag{41}$$

$$\mathbf{x}_{t_i} = \hat{\mathbf{x}}^{\text{noise}}_{t_i} + \Delta t\,g^2(t_i)\,s_\theta(\hat{\mathbf{x}}^{\text{noise}}_{t_i}, t_i), \tag{42}$$

where $z_{t_i} \sim \mathcal{N}(0, I)$. If the physics system is deterministic and Δt is small enough, then we can approximate $\mathbf{x}_{t_i} \approx \hat{\mathbf{x}}_{t_i}$ and for the moment, we assume that we can write

$$\hat{\mathbf{x}}^{\text{noise}}_{t_i} = \mathbf{x}_{t_i} + \sqrt{\Delta t}\,g(t_i)\,z_{t_i}. \tag{43}$$

In this case, we can rewrite the 1-step loss $\mathcal{L}_{\text{single}}(\theta)$ from (5) to obtain the denoising score matching loss

$$\mathcal{L}_{\text{DSM}}(\theta) := \mathbb{E}_{(\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+1}})}\left[||\mathbf{x}_{t_i} - \hat{\mathbf{x}}^{\text{noise}}_{t_i} - \Delta t\,g^2(t_i)\,s_\theta(\hat{\mathbf{x}}^{\text{noise}}_{t_i}, t_i)||_2^2\right], \tag{44}$$

which is the same as minimizing

$$\mathbb{E}_{(\mathbf{x}_{t_i},\mathbf{x}_{t_{i+1}})}\left[||s_\theta(\hat{\mathbf{x}}_{t_i}^{\text{noise}},t_i)-\frac{1}{\Delta t\, g^2(t_i)}(\mathbf{x}_{t_i}-\hat{\mathbf{x}}_{t_i}^{\text{noise}})||_2^2\right]. \tag{45}$$

Now, the idea presented in Vincent (2011) is that for score matching, we can consider a joint distribution $p_{t_i}(\mathbf{x}_{t_i},\tilde{\mathbf{x}}_{t_i})$ of sample $\mathbf{x}_{t_i}$ and corrupted sample $\tilde{\mathbf{x}}_{t_i}$. Using Bayes' rule, we can write $p_{t_i}(\mathbf{x}_{t_i},\tilde{\mathbf{x}}_{t_i})=p_\sigma(\tilde{\mathbf{x}}_{t_i}|\,\mathbf{x}_{t_i})p_{t_i}(\mathbf{x}_{t_i})$. The conditional distribution $p_\sigma(\cdot|\,\mathbf{x}_{t_i})$ for the corrupted sample is then modeled by a Gaussian with standard deviation $\sigma=\sqrt{\Delta t}\,g(t_i)$, i.e., we can write $\tilde{\mathbf{x}}=\mathbf{x}+\sqrt{\Delta t}\,g(t_i)\,z$ for $z\sim\mathcal{N}(0,I)$ similar to equation (43). Moreover, we can define the distribution of corrupted data q_σ as

$$q_\sigma(\tilde{\mathbf{x}})=\int p_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_{t_i}(\mathbf{x})d\mathbf{x}. \tag{46}$$

If σ is small, then $q_\sigma\approx p_{t_i}$ and $\mathrm{KL}(q_\sigma||\,p_{t_i})\to 0$ as $\sigma\to 0$. Importantly, in this case, we can directly compute the score for $p_\sigma(\cdot|\,\mathbf{x})$ as

$$\nabla_{\tilde{\mathbf{x}}}\log p_\sigma(\tilde{\mathbf{x}}|\,\mathbf{x})=\frac{1}{\sigma^2}(\mathbf{x}-\tilde{\mathbf{x}}). \tag{47}$$

Moreover, the theorem proven by Vincent (2011) means that we can use the score of the conditional distribution $p_\sigma(\cdot|\mathbf{x})$ to train $s_\theta(\mathbf{x},t)$ to learn the score of $q_\sigma(\mathbf{x})$, i.e.

$$\arg\min_\theta \mathbb{E}_{\tilde{\mathbf{x}}\sim q_\theta}\left[||s_\theta(\mathbf{x},t_i)-\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})||_2^2\right] \tag{48}$$

$$=\quad \arg\min_\theta \mathbb{E}_{\mathbf{x}\sim p_{t_i},\tilde{\mathbf{x}}\sim p_\sigma(\cdot|\,\mathbf{x})}\left[||s_\theta(\mathbf{x},t_i)-\nabla_{\tilde{\mathbf{x}}}\log p_\sigma(\tilde{\mathbf{x}}|\,\mathbf{x})||_2^2\right]. \tag{49}$$

By combining (49) and (47), this exactly equals the denoising loss $\mathcal{L}_{\text{DSM}}(\theta)$ in (45). As $q_\sigma\approx p_{t_i}$, we also obtain that $\nabla_{\mathbf{x}}\log q_\sigma(\mathbf{x})\approx\nabla_{\mathbf{x}}\log p_{t_i}(\mathbf{x})$, so the network $s_\theta(\mathbf{x},t_i)$ approximately learns the correct score for p_{t_i}.

We have assumed (43) that the only corruption for $\hat{\mathbf{x}}_{t_i}^{\text{noise}}$ is the Gaussian noise. This is not true, as we have

$$\hat{\mathbf{x}}_{t_i}^{\text{noise}}=\mathbf{x}_{t_i}+\sqrt{\Delta t}\,g(t_i)\,z_{t_i}+(\mathbf{x}_{t_{i+1}}-\Delta t\,\mathcal{P}(\mathbf{x}_{t_{i+1}})-\mathbf{x}_{t_i}), \tag{50}$$

so there is an additional source of corruption, which comes from the numerical errors due to the term $\mathbf{x}_{t_{i+1}}-\Delta t\,\mathcal{P}(\mathbf{x}_{t_{i+1}})-\mathbf{x}_{t_i}$. The conditional distribution $p_\sigma(\cdot|\,\mathbf{x})$ is only approximately Gaussian. Ideally, the effects of numerical errors are dominated by the Gaussian random noise. However, even small errors may accumulate for longer sequences of inference steps. To account for this, we argue that the multi-step loss $\mathcal{L}_{\text{multi}}(\theta)$ should be used. During training, with the separation of physics update and denoising, the simulation state is first progressed from time t_{i+1} to time t_i using the reverse physics solver. This only yields a perturbed version of the simulation at time t_i due to numerical inaccuracies. Then a small Gaussian noise is added and, via the denoising network $s_\theta(\mathbf{x},t)$, the simulation state is projected back to the distribution p_{t_i}, which should also resolve the numerical errors. This is iterated, as discussed in section 2 in the main paper, depending on the sliding window size and location.

B. 1D Toy SDE

For the 1D toy SDE discussed in section 3.1, we consider the SDE given by

$$dx = -\left[\lambda_1 \cdot \text{sign}(x)x^2\right] dt + \lambda_2 dw, \tag{51}$$

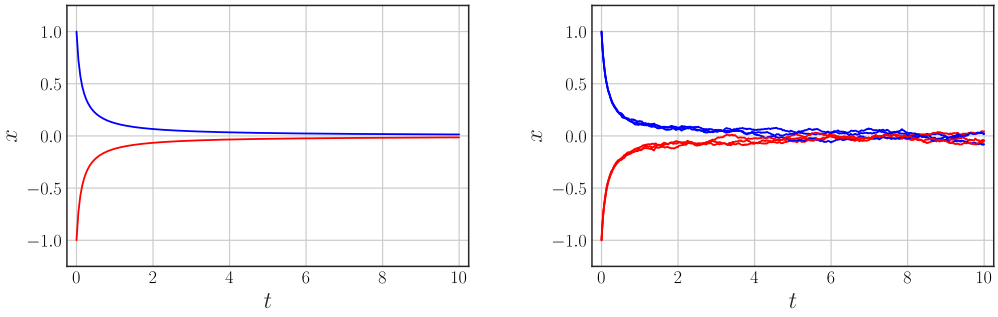
with $\lambda_1 = 7$ and $\lambda_2 = 0.03$. The corresponding reverse-time SDE is

$$dx = -\left[\lambda_1 \cdot \text{sign}(x)x^2 - \lambda_2^2 \cdot \nabla_x \log p_t(x)\right] dt + \lambda_2 dw. \tag{52}$$

Throughout this experiment, p_0 is a categorical distribution, where we draw either 1 or -1 with the same probability. In figure 5, we show trajectories from this SDE simulated with the Euler-Maruyama method. Trajectories start at 1 or -1 and approach 0 as t increases.

Neural network architecture We employ a neural network $s_\theta(x, t)$ parameterized by θ to approximate the score via the 1-step loss, the multi-step loss and implicit score matching (Hyvärinen, 2005, ISM). In all cases, the neural network is a simple multilayer perceptron with elu activations and five hidden layers with 30, 30, 25, 20, and then 10 neurons for the last hidden layer.

We use the Adam optimizer with standard hyperparameters as described in the original paper (Kingma & Ba, 2015). The learning rate, batch size, and the number of epochs depend on the data set size (100% with 2.500 trajectories, 10%, or 1%) and are chosen to ensure convergence of the training loss.



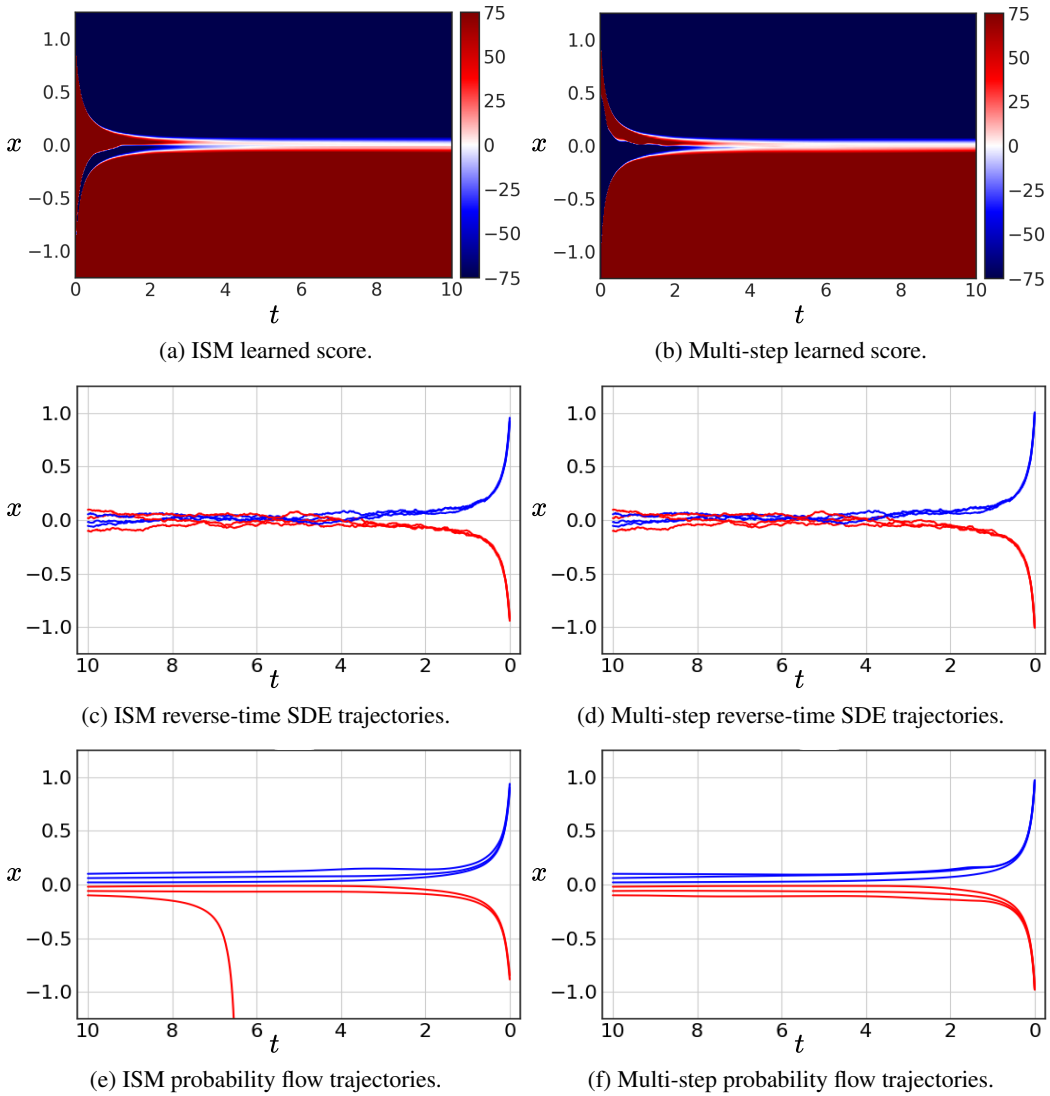
(a) $\lambda_2 = 0$ (b) $\lambda_2 = 0.03$

Figure 5: Trajectories from SDE (51) with $\lambda_2 = 0$ (a) and $\lambda_2 = 0.03$ (b).

Training - 1-step loss For the 1-step loss and all data set sizes, we train for 250 epochs with a learning rate of 10e-3 and batch size of 256. In the first phase, we only keep every 5th point of a trajectory and discard the rest. Then, we again train for 250 epochs with the same batch size and a learning rate of 10e-4 but keep all points. Finally, we finetune the network with 750 training epochs and a learning rate of 10e-5.

Training - multi-step loss For the multi-step loss and 100%, we first train with the 1-step loss, which resembles a sliding window size of 2. We initially train for 1.000 epochs with a batch size of 512 and a learning rate of 10e-3, where we keep only every 5th point on a trajectory and discard the rest. Then, with a decreased learning rate of 10e-4, we begin training with a sliding window size of $S = 2$ and increment it every 1.000 epochs by one until $S_{\max} = 10$. In this phase, we train on all points without any removals.

Training - ISM For ISM, we compute the partial derivative $\partial s_\theta(\mathbf{x})_i / \partial \mathbf{x}_i$ using reverse-mode automatic differentiation in JAX (jax.jacrev). For 100% and 10% of the data set, we train for 2.000 epochs with a learning rate of 10e-3 and batch size of 10.000. Then we train for an additional 2.000 epochs with a learning rate 10e-4. For 1%, increase the number of epochs to 20.000.



(a) ISM learned score.

(b) Multi-step learned score.

(c) ISM reverse-time SDE trajectories.

(d) Multi-step reverse-time SDE trajectories.

(e) ISM probability flow trajectories.

(f) Multi-step probability flow trajectories.

Figure 6: Comparison of Implicit Score Matching (ISM, left) and our proposed training with the multi-step loss (Multi-step, right). Colormap in (a) and (b) truncated to [-75, 75].

Comparison We directly compare the learned score for the reverse-time SDE trajectories and the probability flow trajectories between ISM and the multi-step loss in figure 6 trained on the full dataset. The learned score of ISM and the multi-step loss in figure 6a and figure 6b are visually very similar, showing that our method and loss learn the correct score. Overall, after finetuning both ISM and the multi-step loss, the trajectories of the multi-step loss are more accurate compared to ISM. For example, in figure 6e, a trajectory explodes to negative infinity. Also, trajectories from the multi-step loss end in either -1 or 1, while ISM trajectories are attenuated and do not fully reach -1 or 1 exactly, particularly for the probability flow ODE.

Results of Table 1 in Main Paper We include the standard deviations of table 1 from the main paper in table 2 above. The posterior metric Q is very sensitive to the learned score $s_\theta(\mathbf{x}, t)$. Overall, our proposed multi-step loss gives the most consistent and reliable results.

Method	Probability flow ODE			Reverse-time SDE		
	Dataset size			Dataset size		
	100%	10%	1%	100%	10%	1%
multi-step	**0.97**±0.04	**0.91**±0.05	**0.81**±0.01	**0.99**±0.01	0.94±0.02	**0.85**±0.06
1-step	0.78±0.16	0.44±0.13	0.41±0.13	0.93±0.05	0.71±0.10	0.75±0.10
ISM	0.19±0.05	0.15±0.15	0.01±0.01	0.92±0.05	0.94±0.01	0.52±0.22

Table 2: Posterior metric Q for 1.000 predicted trajectories averaged over three runs.

C. Buoyancy-driven Flow with Obstacles

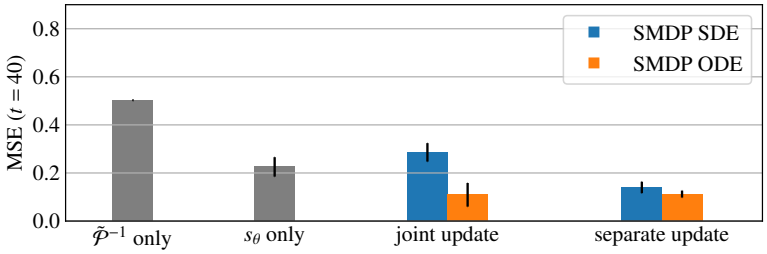
Training We train all networks with Adam and learning rate 10^{-4} with batch size 16. We begin training with a sliding window size of $S = 2$, which we increase every 30 epochs by 2 until $S_{\max} = 20$.

Dil-ResNet The Dil-ResNet architecture is described in (Stachenfeld et al., 2021), Appendix A. We concatenate a constant time channel to the input. Additionally, positional information is added to the network input by encoding the x-position and y-position inside the domain in two separate channels.

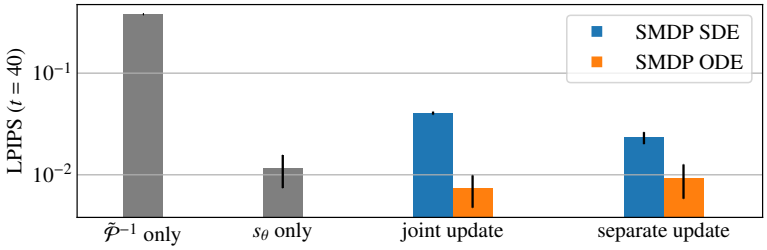
Separate vs. joint updates We compare a joint update of the reverse physics step and corrector function s_θ, see figure 4b, and a separate update of reverse physics step and corrector function, see figure 4c. An evaluation regarding the reconstruction MSE and perceptual distance is shown in figure 7. Both training and inference variants achieve advantages over "$\tilde{\mathcal{P}}^{-1}$ only" and "s_θ only" approaches. Overall, there are no apparent differences for the ODE inference performance but slight benefits for the SDE inference when separating physics and corrector update.

Additional results We provide more detailed visualizations for the buoyancy-driven flow case in figure 9 and figure 10. These again highlight the difficulties of the reverse physics simulator to recover the initial states by itself. Including the learned corrections significantly improves this behavior.

In figure 8, we also show an example of the posterior sampling for the SDE. It becomes apparent that the inferred small-scale structures of the different samples change. However, in contrast to cases like the heat diffusion example, the physics simulation in this scenario leaves only little room for substantial changes in the states.

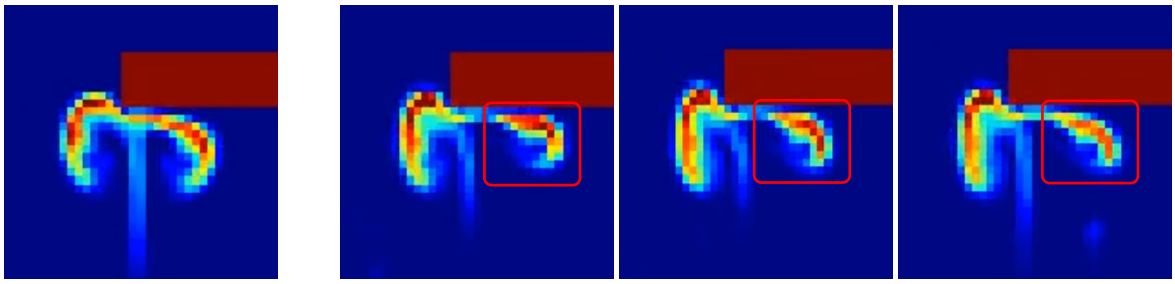


(a) Reconstruction MSE ($t = 40$).



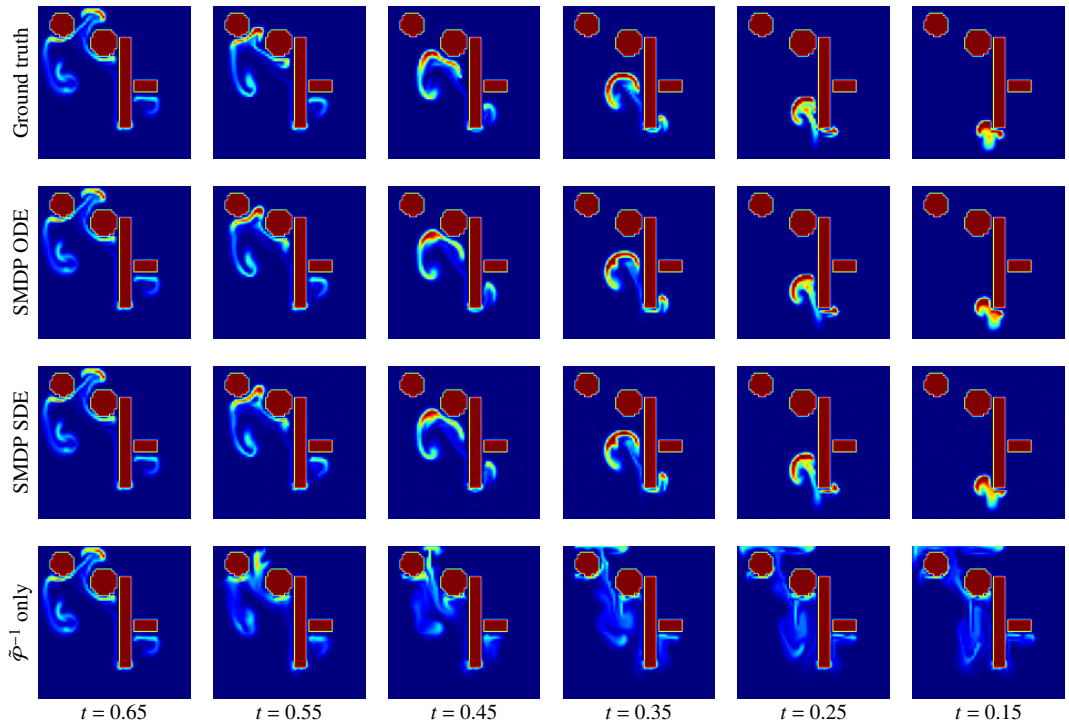
(b) Perceptual distance LPIPS ($t = 40$).

Figure 7: Comparison of separate and joint updates averaged over three runs.

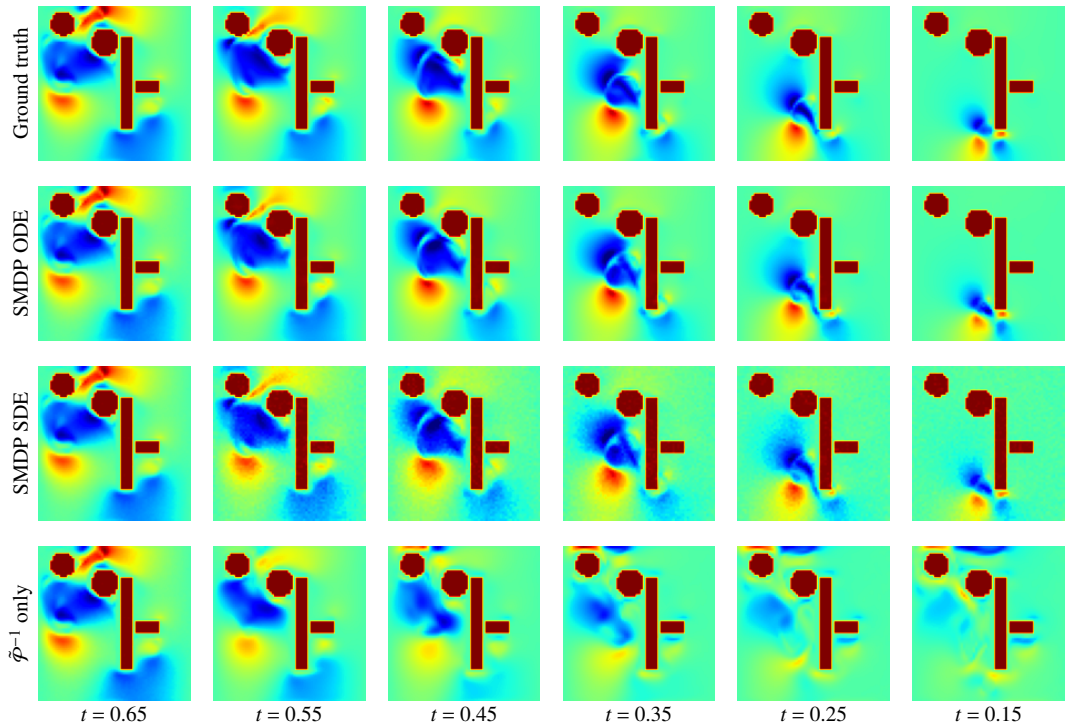


Ground truth SMDP-SDE

Figure 8: Comparison of SMDP-SDE predictions and ground truth for buoyancy-driven flow at $t = 0.36$.

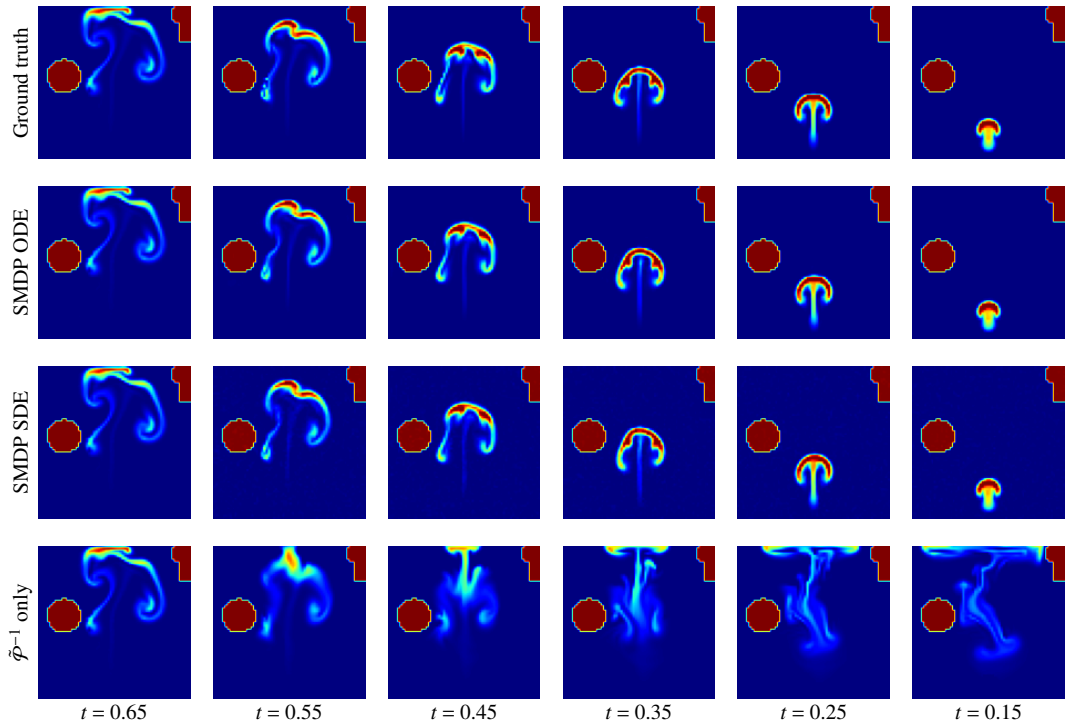


(a) Marker field

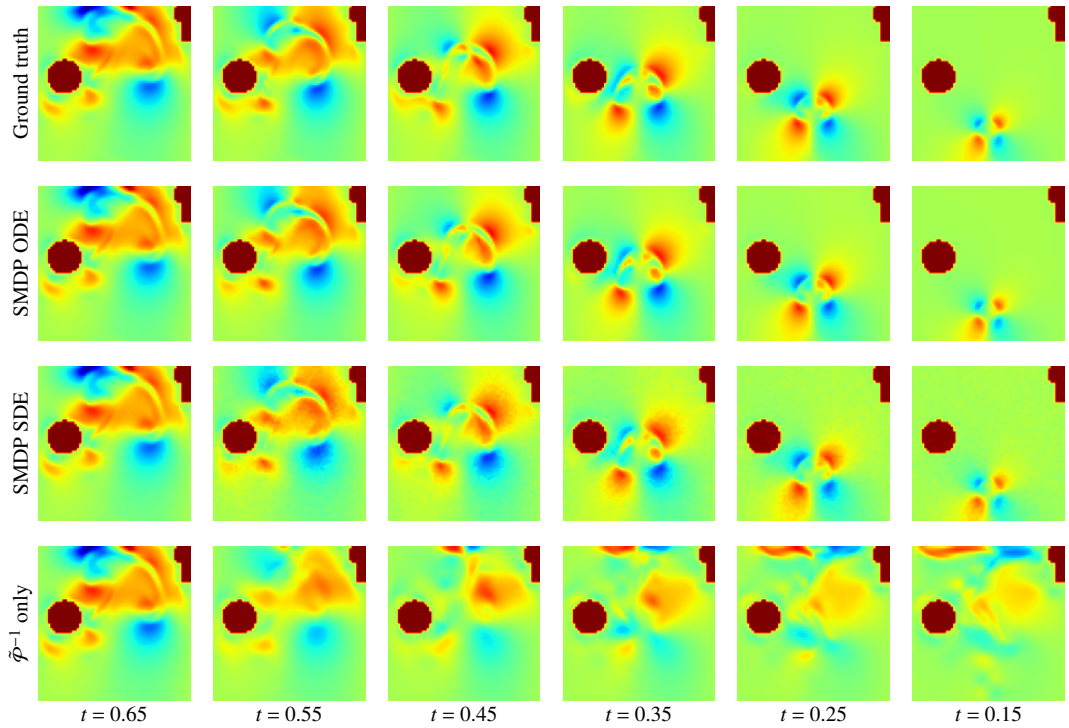


(b) Velocity field (x-direction)

Figure 9: Predictions for buoyancy-driven flow with obstacles (example 1 of 2).



(a) Marker field



(b) Velocity field (x-direction)

Figure 10: Predictions for buoyancy-driven flow with obstacles (example 2 of 2).