Score Matching via Differentiable Physics

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

[1]Technical University of Munich
[2]Max Planck Institute for Astrophysics

Abstract

Diffusion models based on stochastic differential equations (SDEs) gradually perturb a data distribution $p(\mathbf{x})$ over time by adding noise to it. A neural network is trained to approximate the score $\nabla_\mathbf{x} \log p_t(\mathbf{x})$ at time t, which can be used to reverse the corruption process. In this paper, we focus on learning the score field that is associated with the time evolution according to a physics operator in the presence of natural non-deterministic physical processes like diffusion. A decisive difference to previous methods is that the SDE underlying our approach transforms the state of a physical system to another state at a later time. For that purpose, we replace the drift of the underlying SDE formulation with a differentiable simulator or a neural network approximation of the physics. At the core of our method, we optimize the so-called probability flow ODE to fit a training set of simulation trajectories inside an ODE solver and solve the reverse-time SDE for inference to sample plausible trajectories that evolve towards a given end state.

1 Introduction

Many physical systems are time-reversible on a microscopic scale. For example, a continuous material can be represented by a collection of interacting particles [Gurtin, 1982, Blanc et al., 2002] based on which we can predict future states of the material. We can also compute earlier states, meaning we can evolve the simulation backwards in time [Martyna et al., 1996]. When taking a macroscopic perspective, we only know the average quantities within specific regions [Farlow, 1993], which constitutes a loss of information. It is only then that time-reversibility is no longer possible, since many macroscopic and microscopic initial states exist that evolve to yield the same macroscopic state.

In the following, we target inverse problems to reconstruct the distribution of initial macroscopic states for a given end state. This genuinely tough problem has applications in many areas of scientific machine learning [Zhou et al., 1996, Gómez-Bombarelli et al., 2018, Delaquis et al., 2018, Lim and Psaltis, 2022], and existing methods lack tractable approaches to represent and sample the distribution of states. We address this issue by leveraging continuous approaches for diffusion models in the context of physical simulations. In particular, our work builds on the *reverse-diffusion* theorem [Anderson, 1982]. Given the functions $f(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$, called *drift*, and $g(\cdot) : \mathbb{R} \to \mathbb{R}$, called *diffusion*, it can be shown that under mild conditions, for the forward stochastic differential equation (SDE) $d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)dw$ there is a corresponding reverse-time SDE $d\mathbf{x} = [f(\mathbf{x}, t) - g(t)^2 \nabla_\mathbf{x} \log p_t(\mathbf{x})]dt + g(t)d\tilde{w}$. In particular, this means that given a marginal distribution of states $p_0(\mathbf{x})$ at time $t = 0$ and $p_T(\mathbf{x})$ at $t = T$ such that the forward SDE transforms $p_0(\mathbf{x})$ to $p_T(\mathbf{x})$, then the reverse-time SDE runs backward in time and transforms $p_T(\mathbf{x})$ into $p_0(\mathbf{x})$. The term $\nabla_\mathbf{x} \log p_t(\mathbf{x})$ is called the *score*.

*Correspondence to: `benjamin.holzschuh@tum.de`

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

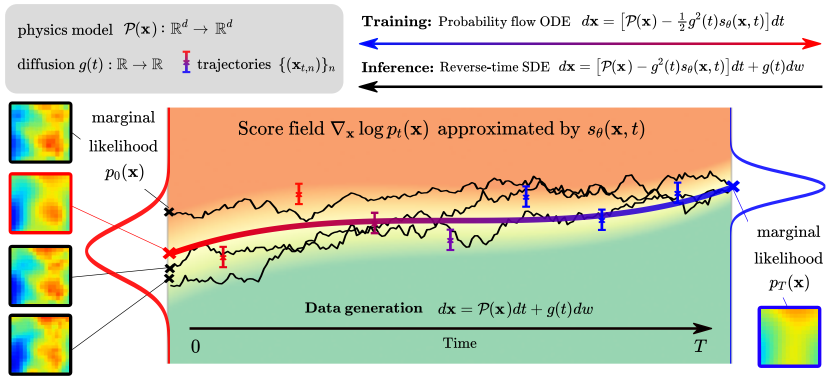

Figure 1: Overview: we employ a physics simulator $\mathcal{P}$ to learn the score field $\nabla_x \log p_t(\mathbf{x})$ with a neural network s_θ in the presence of noise or uncertainties. The trained model allows for sampling the posterior of p_0, i.e. different states that explain an observation p_T, by solving the reverse-time SDE or obtaining a maximum likelihood solution from the probability flow ODE.

This theorem is a central building block for SDE-based diffusion models and denoising score matching [Song et al., 2021b, Jolicoeur-Martineau et al., 2021], which parameterize the drift and diffusion in such a way that the forward SDE corrupts the data and transforms it into random noise. By training a neural network to represent the score, the reverse-time SDE can be deployed as a generative model, which transforms samples from random noise $p_T(\mathbf{x})$ to the data distribution $p_0(\mathbf{x})$.

In this paper, we show that a similar methodology can likewise be employed to model physical processes. We replace the drift $f(\mathbf{x}, t)$ by a physics model $\mathcal{P}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$, which is implemented by a differentiable solver or a neural network that represent the dynamics of a physical system, thus deeply integrating physical knowledge into our method. The end state at $t = T$ on which the forward SDE acts is not fully destroyed by the diffusion $g(t)$, but instead, the noise acts as a perturbation of the system state over time. An overview of our method is shown in figure 1.

2 Method

We choose to model the time evolution of the system by a stochastic differential equation

$$d\mathbf{x} = \mathcal{P}(\mathbf{x})dt + g(t)dw, \tag{1}$$

with diffusion process $g(\cdot) : \mathbb{R} \to \mathbb{R}$ that perturbs the states. We consider a training set of N trajectories $\left\{(\mathbf{x}_{t_i,n})_{i=0}^M\right\}_{n=1}^N$ with time discretization $0 \leq t_0 < ... < t_M \leq T$ sampled from this SDE.

In line with previous work in score-based generative modelling [Song and Ermon, 2019, Song et al., 2021b], we approximate the score $\nabla_\mathbf{x} \log p_t(\mathbf{x})$ of the marginal likelihoods by a neural network $s_\theta(\mathbf{x}, t)$. We optimize $s_\theta(\mathbf{x}, t)$ via maximum likelihood training of the probability flow ODE

$$d\mathbf{x} = \left[\mathcal{P}(\mathbf{x}) - \frac{1}{2}g^2(t)s_\theta(\mathbf{x}, t)\right] dt. \tag{2}$$

Specifically, we maximize a variational lower bound of the likelihood objective by minimizing the L2 distance between the deterministic trajectories of the probability flow ODE and the SDE trajectories $\left\{(\mathbf{x}_{t_i,n})_{i=0}^M\right\}_{n=1}^N$. See Song et al. [2021a], Huang et al. [2021] for details on the equivalency of score matching and maximum likelihood. In contrast to previous work, our method deeply integrates a prior about the physical system in the form of the simulation operator $\mathcal{P}(\mathbf{x})$ into the training process.

Given an end state $\mathbf{x}_T$, we can solve the probability flow backwards in time using the trained score function $s_\theta(\mathbf{x}, t)$ to obtain a trajectory $(\mathbf{x}_{t_i}^{\text{pred}})_{i=1}^M$. However, this will only give a single solution and thus not allow for sampling from the posterior $p(\mathbf{x}\,|\mathbf{x}_T)$. Therefore, we simulate trajectories from the reverse-time SDE

$$d\mathbf{x} = \left[\mathcal{P}(\mathbf{x}) - g^2(t)s_\theta(\mathbf{x}, t)\right] dt + g(t)dw. \tag{3}$$

When $s_\theta(\mathbf{x}, t) \equiv \nabla_\mathbf{x} \log p_t(\mathbf{x})$, then the evolution of marginal probabilities $p_t(\mathbf{x})$ for this SDE is the same as for the probability flow ODE equation 2 [Song et al., 2021b]. Moreover, by the reverse-diffusion theorem [Anderson, 1982], SDE equation 3 is the time-reverse of the physical system SDE from equation 1.

In the following, we refer to the integration of the physics model $\mathcal{P}(\mathbf{x})$ into the score-based modelling approach as *score matching via differentiable physics*, or *SMDP* in short. We denote trajectories from the probability flow ODE by *SMDP-ODE*, and those obtained by simulating the reverse-time SDE by *SMDP-SDE*.

Training and Inference Algorithm 1 gives an overview of SMDP inference for the ODE as well as the SDE variant when using the explicit Euler method as ODE solver. For simplicity, we employ the explicit Euler method for training and backpropagate gradients through multiple solver steps when computing the ODE trajectory in equation (2) to obtain updates for θ. We also refer to this procedure as unrolling the dynamics. Our training setup is similar to Um et al. [2020], which was originally developed for training correction functions in the context of controlling numerical errors for physical simulations. In particular, in our implementation, we consider a sliding window for unrolling the dynamics, which makes our training very flexible, i.e. we can consider single-step updates as well as unrolling the entire simulation.

Algorithm 1 SMDP-ODE, SMDP-SDE

Require: $\mathbf{x}_{t_M}$, $\{t_m\}_{m=0}^M$, $\{g_{t_m}\}_{m=0}^M$

for $m \leftarrow M$ to 1 **do**
  $\mathbf{p} \leftarrow \mathcal{P}(\mathbf{x}_{t_m})$
  $\mathbf{s} \leftarrow -g_{t_m}^2 s_\theta(\mathbf{x}_{t_m}, t_m)/2$
  if SMDP-ODE **then**
    $\mathbf{x}_{t_{m-1}} \leftarrow \mathbf{x}_{t_m} - (t_m - t_{m-1}) \cdot (\mathbf{p} + \mathbf{s})$
  if SMDP-SDE **then**
    $\mathbf{x}_{t_{m-1}} \leftarrow \mathbf{x}_{t_m} - (t_m - t_{m-1}) \cdot (\mathbf{p} + 2\mathbf{s})$
    $z \sim \mathcal{N}(0, I)$
    $\mathbf{x}_{t_{m-1}} \leftarrow \mathbf{x}_{t_{m-1}} + g_{t_m}\sqrt{t_m - t_{m-1}}\, z$
return $\mathbf{x}_{t_0}$

3 Experiments: Heat Equation

We consider the heat equation $\frac{\partial u}{\partial t} = \alpha \Delta u$, which plays a fundamental role in many physical systems. Here, we set the diffusivity constant to $\alpha = 1$ and initial conditions at $t = 0$ are generated from Gaussian random fields with $n = 4$ at resolution 32×32. We simulate the heat diffusion using spectral methods until $t = 0.2$ with a fixed number of simulation steps $M = 32$ using the Euler-Mayurama method with $g \equiv 0.1$.

Our training data set consists of 2.500 initial conditions and trajectories. The test set is comprised of 500 initial conditions and their corresponding end states solved directly without any noise.

Training We consider a small *ResNet*-like architecture based on an encoder and decoder part as representation for the score function $s_\theta(\mathbf{x}, t)$. The physics model $\mathcal{P}$ is implemented via differentiable programming in *JAX* [Schoenholz and Cubuk, 2020].

For better comparison with the baseline methods, these are trained with a Gaussian random noise with $\sigma^2 = 0.1$ added to the inputs. This noise is also added to the initial states during testing of all networks, including SMDP-ODE and SMDP-SDE.

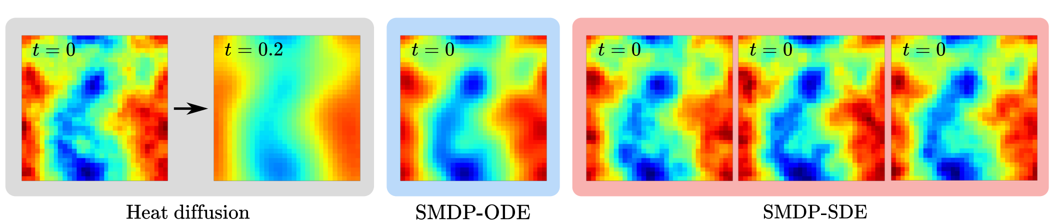

Figure 2: Heat diffusion case. We simulate a Gaussian random field at $t = 0$ forwards in time until $t = 0.2$. Given s_θ we can either solve the probability flow ODE or simulate trajectories of the reverse-time SDE to obtain solutions for the state at $t = 0$.

Method	**MSE** $[10^{-5}]\downarrow$	**Spectral error** $\downarrow$	**Full posterior**
SMDP-ODE	**0.74**	3.62	✗
SMDP-SDE	5.56	**0.56**	✓
ResNet-S	2.17	1.67	✗
ResNet-P	2.30	1.09	✗
BNN-S	3.47×10^2	1.25	✓
BNN-P	3.81×10^2	0.99	✓
FNO-S	2.54×10^4	1.60	✗
FNO-P	2.50×10^4	1.47	✗
HeatGen	1.39	4.45	✗
HeatGen+noise	4.45	3.24	✓

Table 1: Evaluation of reconstruction MSE and spectral error for SMDP and baselines. The column full posterior indicates whether models yield point estimates or allow to sample from the posterior.

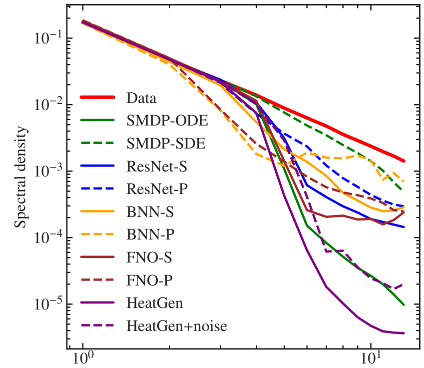

Figure 3: Spectral density on different scales, the red line indicating ground truth. The closer a method is to the ground truth, the better it produces structures of a similar scale.

Baseline methods As baseline methods, we consider the ResNet-like architecture from above, in addition to a *Bayesian neural network* (BNN) based on a U-Net architecture with spatial dropout [Mueller et al., 2022], as well as a *Fourier neural operator* (FNO) network [Li et al., 2020]. For each of these three methods, we consider two variants: the first variant is trained with a *supervised loss*, i.e. the training data consists of pairs $(\mathbf{x}_0, \mathbf{x}_T)$ with initial state $\mathbf{x}_0$ and end state $\mathbf{x}_T$. The supervised loss corresponds to the squared L2 distance between the network prediction $\mathbf{x}_0^{\text{pred}}$ and the ground truth, i.e. $||\mathbf{x}_0^{\text{pred}} - \mathbf{x}_0||_2^2$. For the second variant, the *reconstruction loss*, we rely on the differentiable solver and only make use of the end state $\mathbf{x_T}$ such that the loss becomes $||\mathcal{P}(\mathbf{x}_0^{\text{pred}}; T) - x_T||_2^2$, i.e we simulate the network output forward in time using $\mathcal{P}$ to obtain a state at $t = T$, which we compare to the desired end state x_T. We denote the supervised variant by *S* and the physics-based one by *P*. Additionally, we consider an adopted generative model from Rissanen et al. [2022], denoted by *HeatGen*. We train this network similarly to SMDP-ODE, but without the solver $\mathcal{P}$, such that the network has to learn the score and the physics at the same time.

Reconstruction accuracy vs. fitting the data manifold We give an evaluation of our method and the baselines by considering the *reconstruction MSE* on the test set: how well a predicted initial state $\mathbf{x}_0^{\text{pred}}$ that is simulated forward in time yields states that correspond to the reference end state $\mathbf{x}_T$ in terms of MSE. This metric has the disadvantage that it does not measure how well the prediction matches the training data manifold, i.e. for this case whether the prediction resembles the statistics of the Gaussian random field. For that reason, we additionally compare the power spectral density of the states as the *spectral loss*. The corresponding measurements are given in table 1, which show that our method SMDP-ODE performs best in terms of the reconstruction MSE. However, solutions obtained by SMDP-ODE are very smooth and do not contain the small-scale structures of the references, which is reflected in a high spectral error that is also visually prominent, as shown in figure 3. SMDP-SDE on the other hand performs very well in terms of spectral error and yields visually convincing solutions with only a slight increase in the reconstruction loss. We note that there is a natural tradeoff between both metrics, and SMDP-ODE and SMDP-SDE perform best for both cases respectively while using the same set of weights.

4 Conclusion

We presented first results of SMDP, a derivative of score matching in the context of physical simulations and differentiable physics. We demonstrated the versatility of SMDP with two variants: while the *neural ODE* variant focuses on high MSE accuracies, the *neural SDE* variant allows for sampling the posterior and yields an improved coverage of the target data manifold. Our work presents a first step towards combining physics problems with score matching that has the potential for applications in many areas of numerical simulations.

Broader Impact

Our work targets generating simulations with score-based generative models. While numerical simulations are a broad field that has numerous applications with a positive impact on society, they could potentially also be used with malicious intent, e.g., for manufacturing weapons.

References

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

X. Blanc, C. Le Bris, and P.-L. Lions. From molecular models to continuum mechanics. *Archive for Rational Mechanics and Analysis*, 164(4):341–381, 2002.

S. Delaquis, M. Jewell, I. Ostrovskiy, M. Weber, T. Ziegler, J. Dalmasson, L. Kaufman, T. Richards, J. Albert, G. Anton, et al. Deep neural networks for energy and position reconstruction in exo-200. *Journal of Instrumentation*, 13(08):P08023, 2018.

S. J. Farlow. *Partial differential equations for scientists and engineers*. Courier Corporation, 1993.

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

M. E. Gurtin. *An introduction to continuum mechanics*. Academic press, 1982.

C. Huang, J. H. Lim, and A. C. Courville. A variational perspective on diffusion-based generative models and score matching. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 22863–22876, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/c11abfd29e4d9b4d4b566b01114d8486-Abstract.html`.

A. Jolicoeur-Martineau, K. Li, R. Piché-Taillefer, T. Kachman, and I. Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv:2010.08895*, 2020.

J. Lim and D. Psaltis. Maxwellnet: Physics-driven deep neural network training based on maxwell's equations. *Apl Photonics*, 7(1):011301, 2022.

G. J. Martyna, M. E. Tuckerman, D. J. Tobias, and M. L. Klein. Explicit reversible integrators for extended systems dynamics. *Molecular Physics*, 87(5):1117–1157, 1996.

M. Mueller, R. Greif, F. Jenko, and N. Thuerey. Leveraging stochastic predictions of bayesian neural networks for fluid simulations. *arXiv preprint arXiv:2205.01222*, 2022.

S. Rissanen, M. Heinonen, and A. Solin. Generative modelling with inverse heat dissipation. *arXiv:2206.13397*, 2022.

S. Schoenholz and E. D. Cubuk. Jax md: a framework for differentiable physics. *Advances in Neural Information Processing Systems*, 33:11428–11441, 2020.

Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11895–11907, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html`.

Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum likelihood training of score-based diffusion models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1415–1428, 2021a. URL `https://proceedings.neurips.cc/paper/2021/hash/0a9fdbb17feb6ccb7ec405cfb85222c4-Abstract.html`.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. 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, 2021b. URL `https://openreview.net/forum?id=PxTIG12RRHS`.

K. Um, R. Brand, Y. R. Fei, P. Holl, and N. Thuerey. 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`.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? yes
   (b) Did you describe the limitations of your work? no
   (c) Did you discuss any potential negative societal impacts of your work? yes
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? yes
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? n/a
   (b) Did you include complete proofs of all theoretical results? n/a
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? no
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? no
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? no
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? no
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? n/a
   (b) Did you mention the license of the assets? n/a
   (c) Did you include any new assets either in the supplemental material or as a URL? n/a
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? n/a
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? n/a
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? n/a
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? n/a
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? n/a