Denoisers with Differentiable Constraint Knowledge for Diffusion Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Diffusion models are powerful generative models for complex data distributions, yet they often struggle to generate samples that precisely satisfy constraints inherent in scientific applications. While recent approaches have introduced regularization terms or guidance methods to enforce such constraints, they lead to bias in the generative distribution, compromising the model's ability to faithfully represent the true data distribution. In this extended abstract, we propose a different approach that embeds arbitrary denoiser architectures with differentiable constraints as inductive biases from initialization, maintaining the asymptotic unbiasedness of standard denoising score matching. Through experiments on a representative PDE problem, we show that our method generates constraint-compliant samples without the distributional biases introduced by current methods.

1 Introduction

The challenge of generating realistic data that satisfies specific constraints represents a fundamental requirement across numerous applications in scientific discovery, ranging from finding solutions for ODEs [4] to designing proteins with certain properties [9]. Traditional approaches to constrained generation often struggle to balance the competing objectives of maintaining data fidelity while satisfying these constraints, frequently resulting in artifacts or suboptimal solutions [16]. Recent advances in diffusion models have garnered attention due to their high distributional fidelity to data and sample quality, with methods exploring their use in inverse problems [5, 25, 3] and solving physical ODEs [15, 2].

Previous approaches have explored the use of pre-trained unconditional models, i.e. models that learn the Stein score $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$, to learn a "likelihood-score" adjustment $\nabla_{\boldsymbol{x}_t} \log p(c \mid \boldsymbol{x}_t)$ on a constraint c to guide the diffusion process towards data points that satisfy the constraints [5, 25]. Another branch of research has additionally explored the use of optimization during training the model using a regularizer while training the model [2]. In the former approach, approximations in the inference-time adjustments cause bias in the modeled distribution and the end result may not improve constraint satisfaction significantly over a standard diffusion model. The latter approaches can make optimization more difficult and the methods can suffer from high distributional bias [1].

In this paper, we present Denoisers with Differentiable Constraint Knowledge (DeDiCo). Inspired by guidance literature [5, 25, 26] and second-order autodifferentiation [11] , we propose a method to embed constraint knowledge into arbitrary denoisers by incorporating a differentiable constraint function $r_c : \mathbb{R}^d \mapsto \mathbb{R}$ into the forward pass of the network. Using this approach, we significantly improve constraint satisfaction compared to vanilla diffusion models and approximate guidance based methods, while keeping a low amount of distribution bias. We present our main ideas and results on a toy example and a representative PDE problem.

Submitted to the AI for Science workshop (NeurIPS 2025). Do not distribute.

2 Background

Diffusion models generate samples from a data distribution $p(\boldsymbol{x}_0)$ by learning how to denoise samples from a forward *noising* process [24, 27, 13, 17], which is generally assumed to be of the form:

$$p(\boldsymbol{x}_t) = \int \mathcal{N}\left(\boldsymbol{x}_t; \boldsymbol{x}_0, \sigma(t)^2\mathbf{I}\right) p(\boldsymbol{x}_0)\, \mathrm{d}\boldsymbol{x}_0. \tag{1}$$

The corresponding reverse *denoising* process can be formulated as a probability flow ODE [17]:

$$\mathrm{d}\boldsymbol{x}_t = -\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_t)\, \mathrm{d}t. \tag{2}$$

Starting with a sample from an isotropic Gaussian $\mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}_0, \sigma_{\text{max}}^2\mathbf{I})$ and integrating the ODE backwards in time, it is possible to recover a sample from the original data distribution $\boldsymbol{x} \sim p(\boldsymbol{x}_0)$, as long as the score is learned accurately and σ_{max} is large enough [27]. To get $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$, we first learn a denoiser conditional on the noise level t:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim p(t), \boldsymbol{x}_0 \sim p_{\text{data}}, \boldsymbol{x}_t \sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[w(t) \| D_\theta(\boldsymbol{x}_t, t) - \boldsymbol{x}_0 \|^2 \right], \tag{3}$$

where $w(t)$ and $p(t)$ define the weighting and sampling frequency of noise levels during training. At convergence, $D_\theta(\boldsymbol{x}_t, t) \approx \mathbb{E}[\boldsymbol{x}_0 \,|\, \boldsymbol{x}_t]$. Combined with Tweedie's formula $\mathbb{E}[\boldsymbol{x}_0 \,|\, \boldsymbol{x}_t] = \boldsymbol{x}_t + \sigma(t)^2\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$, this ensures that we can recover an approximation of the score $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$ with $s_\theta(\boldsymbol{x}_t) = \frac{D_\theta(\boldsymbol{x}_t) - \boldsymbol{x}_t}{\sigma(t)^2}$. Accordingly, the loss in Eq. (3) is also called the *denoising score matching* loss in the diffusion literature [28].

Guided Generation. Assume we have a *constraint function* $r_c(\boldsymbol{x})$ where the constraint is satisfied when $r_c(\boldsymbol{x}) = 1$ and not satisfied when $r_c(\boldsymbol{x}) = 0$. It could be a hard constraint such that $r_c(\boldsymbol{x}) \in \{0, 1\}$, or a relaxed continuous constraint $r_c(\boldsymbol{x}) \in [0, 1]$. Given a diffusion model with the output distribution $p(\boldsymbol{x}_0)$, we can turn it into a constrained model with distribution $p(\boldsymbol{x}_0)r_c(\boldsymbol{x}_0)$ by adjusting the score as follows [25, 5]:

$$s_{\text{adjusted}}(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log \int r_c(\boldsymbol{x}_0) p(\boldsymbol{x}_0 \,|\, \boldsymbol{x}_t)\, \mathrm{d}\boldsymbol{x}_0. \tag{4}$$

Many methods have been proposed in the diffusion guidance literature for approximating the second term on the right hand side [25, 5, 14, 26, 23], which generally can be summarized in choosing an approximation for the distribution $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ and an approximation scheme for the integral, e.g. Monte Carlo integration. The idea is to do these approximations at inference time without any changes to the weights of the trained model. While inference-time adjustments are convenient, any error in the approximation results in bias in the output distribution of the diffusion model and constraint satisfaction is only approximate.

Regularization. Another approach to constrain the generative space of a model is to use *regularizers* on the training objective. Popularized by *Physics Informed Neural Networks* [21], the general idea is to have an optimization target that is expanded with a differentiable constraint, typically a residual $\mathcal{R}$ related to a differential equation for a physical system, as follows:

$$\mathcal{L}_{\text{target}}(\theta, \hat{y}) = \mathcal{L}(\theta, \hat{y}) + \lambda \|\mathcal{R}(\hat{y})\|, \tag{5}$$

where $\lambda \geq 0$ is a hyperparameter that defines how much weight to give to the constraint compliance and $\hat{y}$ is the output of the trained model. Although intuitively the target distribution of the learning task should naturally learn to minimize this residual, studies show that these PDE-based regularizers can make the loss landscape hard to optimize [20, 22]. Further, the optimum of Eq. (5) forfeits the property that $D_\theta(\boldsymbol{x}_t, t) \approx \mathbb{E}[\boldsymbol{x}_0 \,|\, \boldsymbol{x}_t]$ at convergence and the connection between the denoiser and $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$ is lost. Thus, the addition of these targets in the loss function can cause bias in the modeled distribution as well.

3 Methods

Our goal is to propose a method that 1) does not cause uncontrolled bias 2) still improves the constraint compliance over a standard diffusion model. The problems with guidance and regularization based methods motivates us to propose the following: Instead of using Eq. (4) to define a guidance function at inference time, we use it to define a constrained denoiser through Tweedie's formula and

backpropagate through the entire expression during training using second-order autodifferentiation. Intuitively, we learn the guidance term to match with the DSM loss in Eq. (3), ensuring no bias.

Crucially, a benefit remains over a standard unconstrained denoiser: In the limit of a perfect approximation to the formula, the diffusion model is forced to generate samples that comply with the constraint throughout training, including initialization. We hypothesize that doing this with even an approximation of Eq. (4) will have two effects: 1) The model will converge in less steps and generate samples that match more accurately with the constraint than an unconstrained model; and 2) by constraining the space of distributions that the model can represent, the method may be more data efficient than an unconstrained model.

To implement the idea in practice, we need to choose an approximation to $p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)$ and a numerical integration scheme. As mentioned in section 2, the guidance literature has developed numerous such approaches: we follow [26], with the choice $p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t) = \mathcal{N}\left(D_\theta(\boldsymbol{x}_t, t), \sigma_{0|t}^2 \mathbf{I}\right)$, where $\sigma_{0|t}^2$ is a hyperparameter typically chosen to be $\frac{\sigma(t)^2}{1+\sigma(t)^2}$, and using Monte Carlo integration for the expectation:

$$\nabla_{\boldsymbol{x}_t} \log p\left(\boldsymbol{x}_t\right) \approx \frac{D_\theta(\boldsymbol{x}_t, t) - \boldsymbol{x}_t}{\sigma_t^2} + \nabla_{\boldsymbol{x}_t} \log \sum\nolimits_{\boldsymbol{x}_0 \sim \mathcal{N}\left(D_\theta(\boldsymbol{x}_t, t), \sigma_{0|t}^2 \mathbf{I}\right)} r_c(\boldsymbol{x}_0). \tag{6}$$

We then use the reparameterization $\boldsymbol{x}_0' = \boldsymbol{x}_0 - D_\theta(\boldsymbol{x}_t, t)$ to allow for backpropagation through the network. Accordingly, we define our constrained denoiser as:

$$D_\theta^c(\boldsymbol{x}_t, t) = D_\theta(\boldsymbol{x}_t, t) + \sigma_t^2 \nabla_{\boldsymbol{x}_t} \log \sum\nolimits_{\boldsymbol{x}_0' \sim \mathcal{N}\left(0, \sigma_{0|t}^2 \mathbf{I}\right)} r_c(\boldsymbol{x}_0' + D_\theta(\boldsymbol{x}_t, t)). \tag{7}$$

Now, we can train the model with the standard denoising score matching loss in Eq. (3) by replacing $D_\theta(\boldsymbol{x}_t)$ with $D_\theta^c(\boldsymbol{x}_t)$. Notice that this requires second-order auto-differentiation: The full forward pass contains the regular forward pass through the original denoiser $D_\theta(\boldsymbol{x}_t, t)$ summed together with a backward pass with respect to inputs $\boldsymbol{x}_t$. During optimization, we then backpropagate through this expanded computational graph.

As mentioned in Section 2, for a wide array of problems we have access to a residual function $\mathcal{R} : \mathbb{R}^d \to \mathbb{R}$ that quantifies how well a data point complies with a constraint, e.g. the solution to an ODE. When we have access to such a function, it is easy to define the constraint in our framework as $r_c(\cdot) = \exp(-\|\mathcal{R}(\cdot)\|)$ for some norm $\|\cdot\|$.

4 Related Work

Conditional Generation and Guidance There exists an ample body of work in guidance for diffusion models: Dhariwal & Nichol [8] introduce *classifier guidance*, where a pre-trained unconditional score is adjusted conditionally through a smaller classifier network. Ho & Salimans [12] propose *classifier-free guidance*, where a diffusion model is adjusted conditionally through a ratio of PDFs, balancing a weight between the unconditional score and the conditional adjustment term. In a recent development, Karras et al. [19] propose contrasting a smaller model with a larger model in a manner similar to classifier-free guidance.

There is a growing body of work focused on using diffusion models to solve inverse problems. Chung et al. [5] and Song et al. [25] propose using pre-trained diffusion models adapted with Eq. (4) for particular inverse problems. Song et al. [26] generalized this notion to any differentiable loss function. Our work differentiates itself by integrating the constraint information into the denoiser itself, embedding an inductive bias in the network at both training and inference.

Recent theory around guidance includes Guo et al. [10] who study the use of gradient information to guide pre-trained diffusion models and give a thorough theoretical analysis of convergence rates for Gaussian linear models. Denker et al. [7] use Doob's h-transform to give insights into conditional generation, with a deep control theoretical analysis. Concurrent work by Dasgupta et al. [6] explores the adaptation of PDEs for posterior score adjustment on diffusion models with particular interest in inverse problem theory. To the best of our knowledge, there is no current work that explores training a constraint-adjusted score from scratch.

Physical Inverse Problems Jacobsen et al. [15] propose CoCoGen, a classifier guidance-based diffusion model, where a pre-trained ControlNet is frozen (i.e., its weights are not updated) and used

as an unconditional score approximation to train a separate network as the posterior adjustment term. They also use additional optimization steps on the diffusion model outputs to refine the residual values, which can cause significant overhead during sampling. Bastek, Sun and Kochmann [2] present Physics-Informed Diffusion Models (PIDM), a framework to train DDPM-based diffusion models with a PDE residual as a regularizer term to minimize along the loss function. Such methods can cause a bias in the generative process to try to comply with the constraint, as the authors themselves recognize in a toy example. Empirically, we have observed no such bias with our method compared to a vanilla diffusion model.

5 Experiments

Toy example: Circles To showcase the unbiasedness of our method, we use a simple toy example introduced by Bastek et al. [2], where the objective is to train a diffusion model to fit a circle of radius 1. Given a sample $(x, y) \in \mathbb{R}^2$, we define a residual function for this setting with the following equation:

$$\mathcal{R}_{\text{AE}}(x, y) = c\left|(x^2 + y^2) - 1\right|, \tag{8}$$

where $c \in \mathbb{R}^+$ is a constant that scales the function. Samples of 500 points trained with vanilla diffusion, PIDM and our method are shown in Fig. 1. The architecture and training details are shown in Appendix B.1. As highlighted by Bastek et al. [2], the performance of PIDM is highly sensitive to the value of c, making it prone to bias with relatively small changes. In this case, PIDM was trained using $c = 1$. On the other hand, DeDiCo shows resilience to hyperparameter changes, since even using a value of $c = 10$ in this experiment shows no particular bias in the learned distribution.

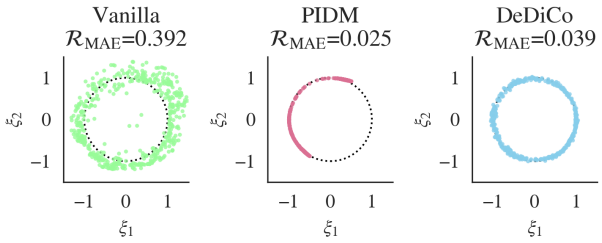

Figure 1: Point distribution learned by different diffusion model approaches with a given training budget. With a higher residual scale, PIDM learns a highly biased distribution, while DeDiCo remains unbiased. Mean residual values are given with $c = 1$.

Darcy Flow The task is to learn how to produce samples of a permeability field $K(x, y)$ and a pressure field $p(x, y)$ in 2 dimensions $(x, y) \in \mathbb{R}^2$ that satisfy the following PDE:

$$\mathcal{R}(K, p) = \nabla \cdot (K \nabla p) + f = 0, \tag{9}$$

where f is the measurement of the flow of some fluid through a porous medium. This PDE doubles as a residual function we can use to verify our generated solutions. For the experiments, the differential terms are estimated through finite differences using the same stencils used by Bastek et al. [2], meaning K and p are sampled as matrices in $\mathbb{R}^{n \times n}$. To tackle this problem, we use a diffusion model with a UNet backbone developed by Karras et al. [18], and use a discretisation of K and p in $\mathbb{R}^{64 \times 64}$. The architecture and training details are available in Appendix B.2.

In Fig. 2, we can see the learned distribution of values of K and p by vanilla diffusion, PIDM, DeDiCo and the vanilla diffusion with inference-time guidance using Eq. (6). We can see that, as highlighted independently by Baldan et al. [1], although PIDM has the lowest residual values in this task, it also shows significant bias in the learned values for the pressure field. On the other hand, DeDiCo shows a clear reduction in residuals compared to vanilla diffusion and diffusion with guidance, with minimal added bias compared to vanilla. Qualitative samples on solutions generated with each method are shown in Appendix A.

We emphasize that there is a baseline level of bias observed in the distribution learned by the vanilla diffusion model. Baldan et al. [1] hypothesized that a contributing factor could be the backbone of the diffusion model used, but in our experiments swapping the UNet for a Diffusion Transformer showed no particular improvement.

6 Limitations and Future Work

One of the limitations is the need of an additional backpropagation through the constraint function and through the network in the forward pass, as well as higher-order auto-differentiation in the backward

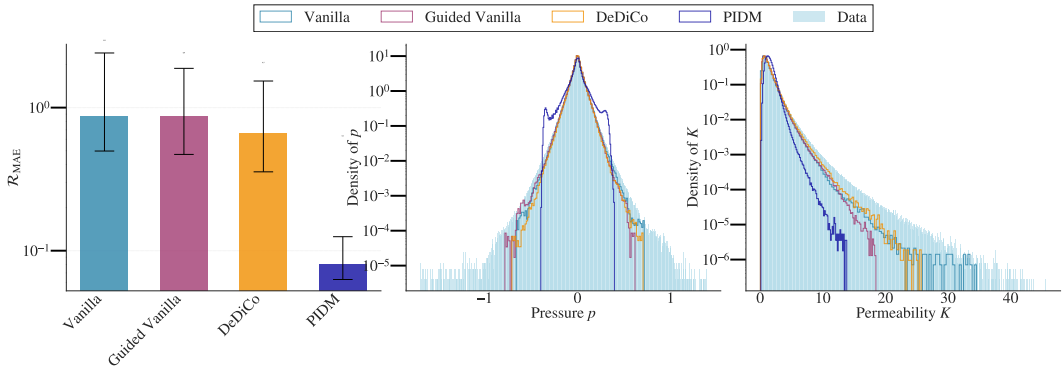

Figure 2: Darcy flow validation. 1000 samples were generated with each approach, using 100 denoising steps for each. Error bars indicate the minimum and maximum from the samples. While PIDM achieves the lowest residual values, this is likely because the distribution it learns is highly biased; all other methods have higher distributional fidelity.

pass. This results in at least twice the time in each network evaluation. Future work could focus on using cheaper approximations for the constraint term of the denoiser, at the cost of lower constraint compliance. Another limitation is that the framework we presented is that only *sets an inductive bias* towards enforcing the constraint, instead of exactly enforcing it. Furthermore, while asymptotically the method should converge to an optimal denoiser, a clear misspecification of the constraint could make it difficult for the denoiser to fit accurately with the data in practice. Future work could focus on quantifying these with error bounds to give better practical guarantees. The current study also does not look into the potential data efficiency aspects of the method, or a comparison to other methods in the presence of slightly incorrectly specified constraints, which are a likely issue in many applications.

References

[1] Giacomo Baldan, Qiang Liu, Alberto Guardone, and Nils Thuerey. Flow Matching Meets PDEs: A Unified Framework for Physics-Constrained Generation, 2025. preprint: arXiv:2506.08604.

[2] Jan-Hendrik Bastek, WaiChing Sun, and Dennis Kochmann. Physics-Informed Diffusion Models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[3] Benjamin Boys, Mark Girolami, Jakiw Pidstrigach, Sebastian Reich, Alan Mosca, and Omer Deniz Akyildiz. Tweedie Moment Projected Diffusions for Inverse Problems. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

[4] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[5] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion Posterior Sampling for General Noisy Inverse Problems. In *The Eleventh International Conference on Learning Representations*, 2023.

[6] Agnimitra Dasgupta, Alexsander Marciano da Cunha, Ali Fardisi, Mehrnegar Aminy, Brianna Binder, Bryan Shaddy, and Assad A. Oberai. Unifying and extending Diffusion Models through PDEs for solving Inverse Problems, 2025. preprint: arXiv:2504.07437.

[7] Alexander Denker, Francisco Vargas, Shreyas Padhy, Kieran Didi, Simon V. Mathis, Riccardo Barbano, Vincent Dutordoir, Emile Mathieu, Urszula Julia Komorowska, and Pietro Lio. DEFT: Efficient Fine-tuning of Diffusion Models by Learning the Generalised h-transform. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024.

[8] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

[9] Nate Gruver, Samuel Don Stanton, Nathan C. Frey, Tim G. J. Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein Design with Guided Discrete Diffusion. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

[10] Yingqing Guo, Hui Yuan, Yukang Yang, Minshuo Chen, and Mengdi Wang. Gradient Guidance for Diffusion Models: An Optimization Perspective. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024.

[11] Adrian Hill, Guillaume Dalle, and Alexis Montoison. An illustrated guide to automatic sparse differentiation. In *ICLR Blogposts 2025*, 2025.

[12] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NeurIPS '20, pages 6840–6851, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-71382-954-6.

[14] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video Diffusion Models. *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 8633–8646, 2022.

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

[16] Petr Karnakov, Sergey Litvinov, and Petros Koumoutsakos. Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks. *PNAS Nexus*, 3(1):pgae005, January 2024. ISSN 2752-6542. doi: 10.1093/pnasnexus/pgae005.

[17] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 26565–26577. Curran Associates, Inc., 2022.

[18] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and Improving the Training Dynamics of Diffusion Models. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24174–24184, 2023.

[19] Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a Diffusion Model with a Bad Version of Itself. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024.

[20] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 26548–26560. Curran Associates, Inc., 2021.

[21] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045.

[22] Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training PINNs: A loss landscape perspective. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[23] Severi Rissanen, Markus Heinonen, and Arno Solin. Free Hunch: Denoiser Covariance Estimation for Diffusion Models Without Extra Costs. In *The Thirteenth International Conference on Learning Representations*, 2025.

[24] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 2015. PMLR.

[25] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-Guided Diffusion Models for Inverse Problems. In *The Eleventh International Conference on Learning Representations*, 2023.

[26] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-Guided Diffusion Models for Plug-and-Play Controllable Generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32483–32498. PMLR, 2023.

[27] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *The Ninth International Conference on Learning Representations*, 2021.

[28] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. doi: 10.1162/NECO_a_00142.

Appendices

A Qualitative samples of Darcy flow using different methods

In Fig. 3 we present some samples generated with each of the main methods presented in this work.

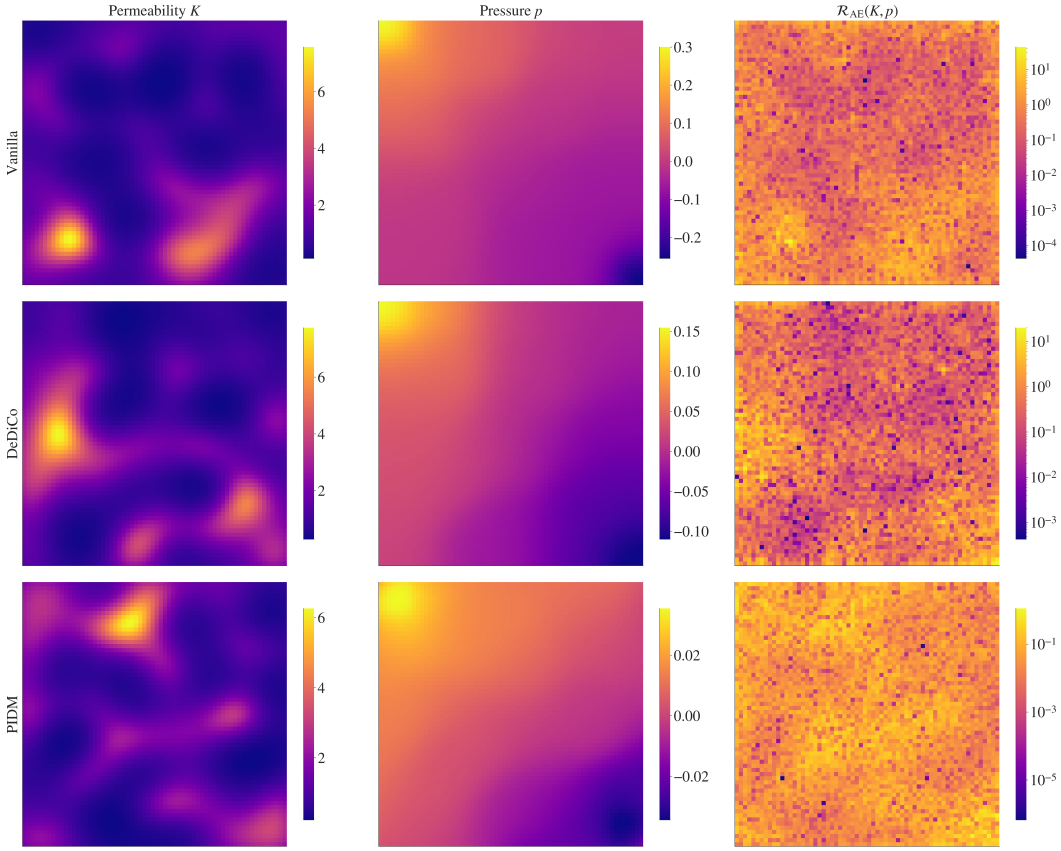

Figure 3: Residuals and samples produced by vanilla diffusion, DeDiCo and PIDM.

B Architectures and training details

The architectures and experiments on this work were implemented using PyTorch 2.4.1. All experiments were ran on single GPUs, from either NVIDIA A100 or V100 GPUs.

B.1 Circles

For the toy example we use a 3 layer MLP with an embedding dimension of 128. To train the networks we use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a fixed learning rate of $\alpha = 1 \cdot 10^{-4}$.

The dataset consisted of 10000 points sampled from the unit circle, and the models were trained over 1000 epochs with a batch size of 128.

B.2 Darcy Flow

For Darcy Flow we used Karras et al.'s UNet [18]. The used hyperparameters on the network are summarized on Table 1.

Table 1: Architecture hyperparameters for the Darcy Flow experiments

Hyperparameter	Value
Model channels	24
Number of residual blocks	8
Per-resolution multipliers	[1, 2, 3, 4]
Attention resolutions	[16, 8]

The PIDM results were reproduced with a pre-trained model provided by Bastek et al. [2]. The vanilla score-matching and DeDiCo networks were trained using Adam with an initial learning rate of $1 \cdot 10^{-2}$ and weight decay as detailed by Karras et al. [18]. The vanilla model was trained on 300k iterations and DeDiCo was trained on 200k iterations, both with a batch size of 64. For DeDiCo, we use 16 Monte Carlo samples for the integral approximation. For all experiments we use Exponentially Moving Averages for sample generation with a decay rate of 0.99.