Denoising Diffusion Restoration Tackles Forward and Inverse Challenges in the Laplace Operator

author names withheld

Editor: Under Review for COLT 2024

Abstract

Diffusion models have emerged as a promising class of generative models that map noisy inputs to realistic images. More recently, they have been employed to generate solutions to partial differential equations (PDEs). However, they still struggle to solve inverse problems in PDEs such as the Poisson equation because the eigenvalues that are large in magnitude amplify the measurement noise. This paper presents a novel approach for the inverse solution of PDEs through the use of denoising diffusion restoration models (DDRM). DDRMs were used in linear inverse problems to restore original clean signals by exploiting the singular value decomposition (SVD) of the linear operator. Equivalently, we present an approach to restore the original parameters in PDEs by exploiting the eigenvalues and the eigenfunctions of the Laplacian operator. Our results show that using denoising diffusion restoration significantly improves the estimation of the parameters.

Keywords: List of keywords

1. Introduction

Denoising diffusion models are the current leading method for generative modeling Ho et al. (2020); Song et al. (2021). These models consist of a two-step process: a forward step where noise is added to iteratively destruct data, and a reverse step, where a deep neural network learns to denoise and recover the original data by reversing the noising process. After training, these models can generate new samples starting from various distributions. Some of the benefits of these models are that they are stable to train and are relatively easy to scale.

Denoising diffusion models have shown great success in applications such as the generation of images, speech, and video, as well as image super-resolution Song et al. (2021); Yang et al. (2023). Other applications include physics-guided human motion (PhysDiff) Yuan et al. (2023), customized ODE solvers that are more efficient than Runge-Kutta methods Lu et al. (2022), molecule generation Hoogeboom et al. (2022), and more Yang et al. (2023). For a more comprehensive review of diffusion models and their applications, we refer readers to the recent review written by Yang et al. 2023.

The Laplace operator is a differential operator of second order, $\Delta = \nabla \cdot \nabla$ Gilbarg and Trudinger (1983). It appears in many partial differential equations (PDEs) such as the Poisson equation, heat equation, and wave equation. It is a compact self-adjoint operator and thus, has an orthonormal set of eigenfunctions and real eigenvalues Chavel (1984).

Since many PDEs do not have an analytical solution, numerical methods are necessary for obtaining solutions to these systems Thomas (2013); Quarteroni et al. (2006). However, numerical methods lead to known numerical errors, especially for complex PDEs. Additionally, while there are many PDE solvers, they are often restricted to the specific type of PDE they are designed for. When working to understand the physics of a system, these numerical errors add noise which makes

the physics difficult to solve. In recent years, deep learning techniques have been introduced to solve PDEs. Such examples include physics-informed neural networks (PINNs) Raissi et al. (2019), deep operator networks (DeepONets) Lu et al. (2019), and Fourier neural operators (FNOs) Li et al. (2020). These techniques have been used to improve computational efficiency, reduce numerical errors, conduct reduced-order modeling, and develop generalized PDE solvers. Although these techniques are effective in solving PDEs, their precision and generalizability, like many machine and deep learning methods, are limited by the scarcity of high-quality training data.

In more recent works, Apte et al. 2023 seek to address the problem of data scarcity for machine learning methods of PDE modeling by developing a method of data generation using a diffusion model. By training on data from the steady 2D Poisson equation on a fixed square domain, they made diffusion models generate paired data samples that adhered to physics laws, despite not including physics into the model directly. They trained a model to generate pairs of u and f satisfying $\Delta u = f$, thus addressing the challenge of capturing the joint distribution of u and f. We intend to build up on this work by adding conditions on u or f drawn from a test set (and hence unseen by the diffusion model during training).

We first start by replicating diffusion model-based data generation for the 2D Poisson equation with homogeneous Dirichlet boundary conditions as in Apte et al. 2023. We trained a denoising diffusion implicit model (DDIM) Song et al. (2021) on a sample of 38,250 data points. After training the diffusion model, we generate numerical solutions $u(x, y)$ conditioned on the parameter $f(x, y)$, which we will refer to as the forward process. Our numerical results are posted in appendix C, and show that the DDIM model is a great, albeit noisy numerical solver to the Poisson equation. We attempt to generate approximations to the parameter $f(x, y)$ conditioned on $u(x, y)$, which we will refer to as the inverse process. Our numerical results are posted in appendix D and show that the DDIM model is a poor numerical solver to the inverse problem. This shows that we need a better method to solve the inverse problem.

To solve this problem, we employ denoising diffusion restoration models (DDRMs) Kawar et al. (2022); Chung et al. (2023); Murata et al. (2023). They are used to restore clean data in linear inverse problems using a pre-trained diffusion model without requiring any fine-tuning. The authors achieve this by using the singular value decomposition (SVD) of the linear operator to transform the original signal and observed signal to a shared spectral space. DDRMs showed state-of-the-art performance in restoring realistic images in super-resolution and deblurring tasks by assuming that the original clean image is returned by a generative model.

We use DDRMs to solve the Poisson equation for $f(x, y)$ conditioned on $u(x, y)$ and to solve $u(x, y)$ conditioned on $f(x, y)$ based on Kawar et al. (2022). Similar to how DDRMs solve linear inverse problems by exploiting the singular value decomposition of the linear operator, we solve inverse problems in the Poisson equation by exploiting the eigenspace of the Laplace operator constrained to homogeneous Dirichlet boundary conditions. Our method shows a significant improvement in the restoration of the parameters, achieving a mean absolute error (MAE) of 0.03215. Furthermore, we achieve an average MAE of $9.675e-07$ in our improved forward process, which is just slightly greater than the MAE of $6.672e-07$ upon using the finite difference method, thus showing the significant improvement due to using DDRM.

2. Background

Consider the following inverse problem defined on a domain $\Omega = [0, 1]^2$

$$\begin{aligned} f_0 &= \Delta u_0, \\ u &= u_0 + z, \end{aligned} \tag{1}$$

where $z \sim N(0, \sigma_f^2)$ is a measurement noise with known covariance σ_f^2. Δu_0 is the Laplacian of a function $u_0 \in C^2(\Omega) \cup C^1(\partial\Omega)$, and $f_0 \in C(\Omega)$ is a forcing function. And we have the boundary conditions $u_0 = 0$ on $\partial\Omega$. As shown in appendix D, a pre-trained diffusion model fails to retrieve the parameter $f_0(x, y)$ conditioned on $u(x, y)$.

In this section, we will introduce some prior work that assists us in deriving an effective method to derive $f(x, y)$ accurately.

2.1. Denoising Diffusion Restoration Models

DDRM is a method that uses a pre-trained diffusion model p_θ as a prior for data Kawar et al. (2022). It is used to restore clean images in non-blind linear inverse problems of the form $y = Hx_0 + z$, where x_0 is the original image, z is measurement noise with known covariance, and H is a linear operator. It is defined as a Markov chain $x_T \to x_{T-1} \to \cdots \to x_1 \to x_0$ conditioned on y:

$$p(x_{0:T}|y) = p_\theta^{(T)}(x_T|y) \prod_{t=0}^{T-1} p_\theta^{(t)}(x_t|x_{t+1}, y). \tag{2}$$

DDRM uses the singular value decomposition of H to project x_T and y into a shared spectral space. It has shown improved performance in restoring clean images in multiple tasks such as image deblurring, inpainting removal, image coloration, and super-resolution Kawar et al. (2022). This work has been followed by the works of Chung et al. (2023); Murata et al. (2023) that extend DDRMs to blind inverse problems where the operator H is unknown.

2.2. Eigenvalues and Eigenfunctions of the Laplacian operator

Although problems like in Eq. 1 do not have a notion of singular value decomposition, its eigenvalue decomposition has been explored in past works in the PDE literature and offers us a method to project u and f_0 into a shared spectral space. That is explained by the following proposition.

Proposition 1 *The eigenfunction and eigenvalue pairs of the Laplacian operator Δ in a domain $\Omega = [0, 1]^2$ subject to the boundary conditions $u = 0$ on $\partial\Omega$ are of the form*

$$u_{n,m}(x, y) = \sin(n\pi x) \sin(m\pi y) \tag{3}$$

$$\lambda_{n,m} = -(n\pi)^2 - (m\pi)^2 \tag{4}$$

For completeness, a proof of this known proposition is provided in appendix A. The proposition also explains the numerical results shown in appendix C and D. Due to the large magnitude of the eigenvalues, the Laplacian operator amplifies the noise measurement noise z, thus making f_0 harder to compute.

3. DDRM for solving PDEs

In this section, we will explain the use of DDRM in sampling f while conditioned on u. We will denote the projection of $f(x, y)$ and $u(x, y)$ into a 64×64 grid as $\mathbf{f}$ and $\mathbf{u}$ respectively. We define the DDRM in this example to be a Markov chain $\mathbf{f}_T \to \mathbf{f}_{T-1} \to \cdots \to \mathbf{f}_1 \to \mathbf{f}_0$ conditioned on $\mathbf{u}$:

$$p(\mathbf{f}_{0:T}|y) = p_\theta^{(T)}(\mathbf{f}_T|\mathbf{u}) \prod_{t=0}^{T-1} p_\theta^{(t)}(\mathbf{f}_t|\mathbf{f}_{t+1}, \mathbf{u}) \tag{5}$$

In the sampling of $\mathbf{f}_T$ and $\mathbf{f}_t$ for $t = 0, ..., T-1$, our approach is a modified version of Kawar et al. (2022), where we consider the projection of u and f in the eigenfunctions of the Laplacian operator instead of singular vectors of a linear operator.

Sampling of $\mathbf{f}_T$.

The sampling of $\mathbf{f}_T$ is performed by sampling from the distribution $p(\mathbf{f}_T|\mathbf{u})$. Sampling from this conditional distribution is intractable, so we use our modified DDRM to approximate the distribution.

We consider the discrete sine transform (DST) of $\mathbf{u}$ and $\mathbf{f}_t$ and perform the diffusion in its spectral space. Define $\overline{\mathbf{u}}^{(n,m)}$ and $\overline{\mathbf{f}}_t^{(n,m)}$ as follows:

$$\overline{\mathbf{u}}^{(n,m)} = \lambda_{n,m} \langle \mathbf{u}, \sin(n\pi x) \sin(m\pi y) \rangle, \tag{6}$$

$$\overline{\mathbf{f}}^{(n,m)} = \langle \mathbf{f}_t, \sin(n\pi x) \sin(m\pi y) \rangle, \tag{7}$$

where $\lambda_{n,m}$ are the eigenvalues from proposition 1. Since none of the eigenvalues $\lambda_{m,n}$ are zero, our DDRM method for sampling $\mathbf{f}_T$ is as follows:

$$p_\theta^{(T)}(\overline{\mathbf{f}}_t^{(n,m)} \mid \mathbf{u}) = \mathcal{N}(\overline{\mathbf{u}}^{(n,m)}, \sigma_T^2 - \sigma_y^2 \lambda_{n,m}^2) \tag{8}$$

In this process, we assume that σ_T is sufficiently large to satisfy $\sigma_T > \sigma_y(2 \times 64^2 \times \pi^2)$, so that the variance term is non-negative.

Sampling of $\mathbf{f}_t$.

The sampling of $\mathbf{f}_t$ is performed by sampling from the distribution $p(\mathbf{f}_t|\mathbf{f}_{t+1}, ..., \mathbf{f}_T, \mathbf{u})$. Sampling from this conditional distribution is intractable, so we use our modified DDRM to approximate the distribution. We denote the prediction of $\mathbf{f}_0$ at time step t as $\mathbf{f}_{\theta,t}$. Our DDRM method of sampling $\mathbf{f}_t$ is as follows:

$$p_\theta^{(t)}\left(\overline{\mathbf{f}}_t^{(n,m)} \mid \mathbf{f}_{t+1}, \mathbf{u}\right) = \begin{cases} \mathcal{N}\left(\overline{\mathbf{f}}_{\theta,t}^{(n,m)} + \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{u}}^{(n,m)} - \overline{\mathbf{f}}_{\theta,t}^{(n,m)}}{\sigma_{\mathbf{f}}/\lambda_{n,m}}, \eta^2\sigma_t^2\right), \\ \qquad\qquad \text{if } \sigma_t < \sigma_{\mathbf{f}}\lambda_{n,m} \\ \mathcal{N}\left((1-\eta_b)\,\overline{\mathbf{f}}_{\theta,t}^{(n,m)} + \eta_b \overline{\mathbf{u}}^{(n,m)}, \sigma_t^2 - \frac{\sigma_{\mathbf{f}}^2}{\lambda_{m,n}^2}\eta_b^2\right), \\ \qquad\qquad \text{if } \sigma_t \geq \sigma_{\mathbf{f}}\lambda_{n,m} \end{cases} \tag{9}$$

where $0 \leq \eta \leq 1$ and $0 \leq \eta_b \leq 1$ are hyperparameters, and $0 = \sigma_0 < \sigma_1 < \sigma_2 < \cdots < \sigma_T$ are noise levels that is the same as that defined with the pre-trained diffusion model.

4. Methods

4.1. Diffusion model for PDE data generation

We replicated the method of PDE solution generation of the 2D Poisson equation with Dirichlet boundary conditions as done in Apte et al. (2023), by training a diffusion model by modifying the GitHub repository Sehwag 2022 that is based on DDIM. This model involves a forward (or "diffusion process") that is a Markov chain, which gradually adds Gaussian noise to the data given by a cosine scheduler, which is the variance scheduler Sehwag (2022).

This model has previously been used for image generation on nine commonly used datasets including MNIST, CIFAR-10, Oxford flowers, traffic signs, and others Sehwag (2022). For almost all the datasets the image resolution was 64×64, which is the same grid size that we use for the PDE data (see Section 4.2). Notably, the model performs well on this variety of image types and therefore is a good candidate for the PDE solutions generations as was shown by Apte et al. 2023.

We note that this is a large diffusion model so it is very computationally expensive to train. We trained our diffusion model on Compute Canada using 4 V100 GPUs for our dataset of 38,250 PDE solution samples (see Section 4.2) and the training time was approximately 4 days.

Some of the unconditionally generated data are posted in appendix G. We compared generated pairs of u and f (labeled "u" and "f" in figure 12) with the finite difference solution (labeled "numerical solution" in figure 12) for comparison. Our MAE between the generated $u(x, y)$ and the finite difference solution is 0.0004373, thus showing that the generated solutions are a good approximation to the true solution.

4.2. Dataset Generation

In our study we investigate PDE data generation and solution restoration of the 2D Poisson equation

$$\Delta u = f \tag{10}$$

on the domain $\Omega = [0, 1]^2$ with with homogeneous Dirichlet boundary conditions, $u = 0$ on $\partial\Omega$ as in Apte et al. 2023.

To train the diffusion mode for PDE data generation, Apte et al. 2023 reported that they generated 10,000 pairs of $[f, u]$ that satisfy Eq. 10 on a 64×64 grid on the domain Ω using a multigrid solver. However, no details were given about the types of functions that were used other than a few figures in the paper so we were unable to replicate or obtain their exact training dataset. This limitation led us to generate our own training dataset in the following way.

For our training dataset, we generated 38,250 samples for the diffusion model based on analytical solutions of Eq. 10. This choice was made to avoid numerical errors in the training data set so that the diffusion model could learn the physics of the system. To do this we used (i) functions based on a neural network (appendix B.1) and (ii) analytical solutions by choosing differentiable u that satisfies the boundary condition and solving for f directly (appendix B.2). This data was used to train the diffusion model for the generation of PDE data.

5. Numerical Results

We conducted our experiments on the dataset described in section 4.2 using a trained DDIM model. Our parameters are $\eta = 0.8$ and $\eta_b = 0.9$. We used $\sigma_y = 1e{-}5$ to ensure that the sampling of $\mathbf{f}_T$ can

be done for any $\sigma_T > 1$. Our test set is 64 samples of neural network pairs described in appendix B.1 with seeds separate from the training set. To ensure the reproducibility of our results, we posted our code in the following GitHub repository: https://github.com/amartyamukherjee/minimal-diffusion

Upon using our modified DDRM to sample $f(x, y)$ conditioned on $u(x, y)$, we got an MAE of 0.03215, which is a significant improvement compared to using the DDIM model without restoration. For qualitative results, we posted three samples in appendix E. This, as a result, demonstrates the practicality of DDRM in solving inverse problems in PDEs.

This method, however, does not outperform finite difference approximation that had an MAE of 0.01663 on the test set in approximating $f(x, y)$. This is because finite difference methods do not assume any notion of periodicity in the dataset, unlike the DST method, thus approximating the Laplacian in the boundary more reliably.

Dry Inverse Process	DDRM	Finite Difference
5.515e-01	3.215e-02	1.163e-02

Table 1: MAE in predicting f conditioned on u, averaged along 64 samples

5.1. Sampling u conditioned on f

PINNs	Dry Forward Process	DeepONet	DDRM	Finite Difference
2.156e-03	1.123e-03		9.675e-07	6.672e-07

Table 2: MAE in predicting u conditioned on f, averaged along 64 samples

6. Discussion

In this study, we replicated the methods for PDE data generation in Apte et al. 2023 with our own generated training data set based on analytical solutions of the 2D Poisson equation (Eq.10). Indeed, the dataset used for training is different from the dataset used in Apte et al. 2023 due to the lack of information about the PDEs used in their study other than the size of the grid and that they used multigrid methods to generate pairs of $[f, u]$. Our approach differs in that we chose to use data that includes differentiable u that satisfies the boundary conditions and f computed directly. Clearly, using a different training dataset will results in a different trained diffusion model. However, our results did show that the diffusion model was able to generate paired PDE solutions that adhere to physics laws for certain function types, as shown in appendix G.

The diffusion model used is popular within this field and has shown success for image generation of various types of datasets from MNIST to celebrity faces to melanoma images Sehwag (2022). However, in our PDE data generation example we seek to generate not just 1 "image" or function solution, but pairs of functions $[f, u]$ that adhere to the physics of the PDE, which in this case is the Poisson equation (Eq.10). This is what makes this approach novel and more difficult than previous applications.

Apte et al. 2023 found that the diffusion model was able to effectively generate pairs of PDE solutions for the 2D Poisson equation based on both visual and statistical analysis. They reported that pairs of functions adhered to the underlying physics despite not including the physics into the loss function like would be done in physics-informed neural networks Blechschmidt and Ernst

(2021); Apte et al. (2023). We found that for PDE solutions without a lot of oscillations, this was true for our model training, but the model had difficulty with functions with a lot of oscillations, as shown in appendix H. It seems that the diffusion model is unable to learn the corresponding function u and just predicts random noise (figure 13). This may be a limitation of this type of PDE data generation. While this should be investigated further, we propose it may be a limitation of the grid size of 64×64. A finer grid may lead to a smaller change between each point on the mesh when there is a high frequency as in the types shown in appendix H.

6.1. Future directions

Indeed, the Poisson equation is solvable analytically for many functions u that are differentiable. Additionally, finite differences are a good numerical solver for the 2D Poisson equation. However, we chose to start with the Poisson equation to be able to determine a good measure for the performance of the diffusion model and restorations by directly comparing our results with finite differences. Indeed, more complex PDEs, such as the Navier-Stokes equation and Darcy flow could be investigated using similar methods.

Diffusion models are a new and very active area of research. While the diffusion models used in our paper do not directly include the physics, future work may be able to include physics more directly in a similar approach to physics-informed neural networks Raissi et al. (2019); Blechschmidt and Ernst (2021); Zhang (2022). We do note that diffusion models do seem to adhere to physics after training as shown in Apte et al. 2023, but have limitations as we found with certain types of solutions.

References

Rucha Apte, Sheel Nidhan, Rishikesh Ranade, and Jay Pathak. Diffusion model based data generation for partial differential equations. *ICML workshop on Synergy of Scientific and Machine Learning Modeling*, 2023.

Jan Blechschmidt and Oliver G Ernst. Three ways to solve partial differential equations with neural networks—a review. *GAMM-Mitteilungen*, 44(2):e202100006, 2021.

Isaac Chavel. *Eigenvalues in Riemannian geometry*. Academic press, 1984.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023.

David Gilbarg and Neil S. Trudinger. *Elliptic Partial Differential Equations of Second Order, Second Edition*. Springer-Verlag, 1983.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR, 2022.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, 2022.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Naoki Murata, Koichi Saito, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Yuki Mitsufuji, and Stefano Ermon. GibbsDDRM: A partially collapsed gibbs sampler for solving blind inverse problems with denoising diffusion restoration. In *International Conference on Machine Learning*, 2023.

Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical mathematics*, volume 37. Springer Science & Business Media, 2006.

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.

Vikash Sehwag. minimal-diffusion, 2022. URL https://github.com/VSehwag/minimal-diffusion.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *International Conference on Learning Representations*, 2021.

James William Thomas. *Numerical partial differential equations: finite difference methods*, volume 22. Springer Science & Business Media, 2013.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16010–16021, 2023.

Zhao Zhang. A physics-informed deep convolutional neural network for simulating and predicting transient darcy flows in heterogeneous reservoirs without labeled data. *Journal of Petroleum Science and Engineering*, 211:110179, 2022.

Appendix A. Proof of proposition 1

Proof To find the eigenvalues and eigenfunctions of the Laplacian operator, we are essentially solving the following PDE

$$\Delta u = \lambda u$$

We use separation of parts to split u as $u(x, y) = X(x)Y(y)$.

$$\Delta u = X''(x)Y(y) + X(x)Y''(y) = \lambda X(x)Y(y)$$

$$\frac{X''}{X} + \frac{Y''}{Y} = \lambda$$

Here, we solve two eigenvalue problems, $X''/X = \lambda_1$ and $Y''/Y = \lambda_2$, which means $\lambda = \lambda_1 + \lambda_2$. Consider the boundary value problem for X.

$$\frac{X''}{X} = \lambda_1, X(0) = X(1) = 0$$

The eigenvalue and eigenfunction pairs are trivially of the form

$$X_n(x) = \sin(n\pi x), \lambda_{1,n} = -(n\pi)^2$$

By symmetry, we have

$$Y_m(y) = \sin(m\pi y), \lambda_{2,m} = -(m\pi)^2$$

Thus, the eigenvalue and eigenfunction pairs of the Laplacian operator are of the form

$$u_{n,m}(x, y) = \sin(n\pi x)\sin(m\pi y)$$
$$\lambda_{n,m} = -(n\pi)^2 - (m\pi)^2$$

■

Appendix B. Dataset solutions

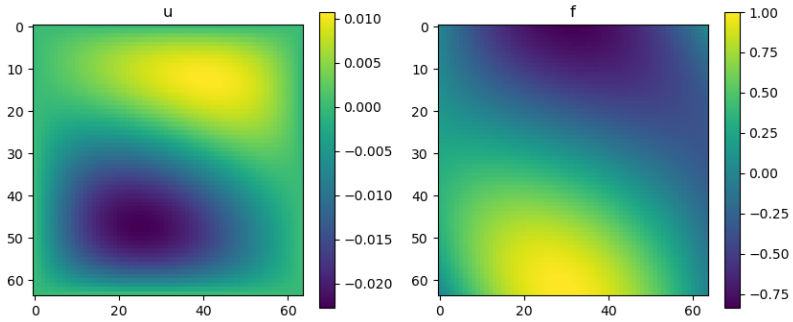
We generated 38,250 samples for the diffusion model for PDE data generation of the following types: neural network pairs and analytical pairs. Here we show some examples of these different types of data.

B.1. Neural network pairs

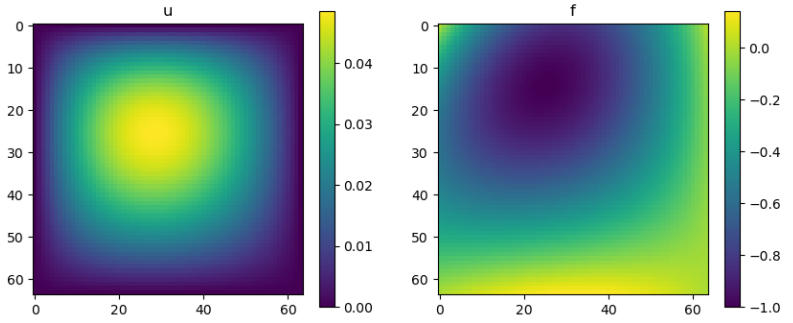
To respect the Dirichlet boundary conditions, we sampled $u(x, y)$ randomly as:

$$u(x, y) = g_{NN}(x, y)x(1 - x)y(1 - y), \tag{11}$$

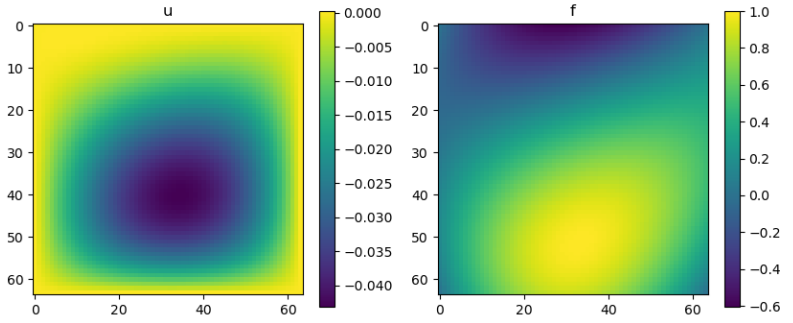
where g_{NN} is a randomly initialized neural network with three hidden layers and *tanh* activation function. Clearly u satisfies the boundary conditions since when x or y is 1 or 0, $u = 0$. Additionally, u is differentiable because of the use of the *tanh* activation function. We then compute $f = \Delta u$ using auto-differentiation by using the `autograd.grad` function from `Pytorch`. Using this, we generated 10,000 samples in our dataset. Example neural network pairs are shown in Figures 1.



(*a*)



(*b*)



(*c*)

Figure 1: Examples of neural network pairs.

B.2. Analytical pairs

To train a diffusion model to generate data for the 2D Poisson equation with homogeneous Diriclet boundary condition (Eq. 10) we generated training data by using smooth functions u that satisfy the boundary conditions and solving for f analytically. We created a 64×64 meshgrid for the domain $\Omega = [0, 1]^2$ and used the analytical solution for u and f. We classified these function pairs $[f, u]$ as different types.

TYPE 1 ANALYTICAL PAIRS

$$u(x, y) = \sin(n\pi x) \sin(k\pi y)$$

where n, k are positive integers. We can solve u analytically to get

$$\nabla^2 u = f = -\pi^2(n^2 + k^2)\sin(n\pi x)\sin(k\pi y).$$

An example solution is shown in Fig. 2.

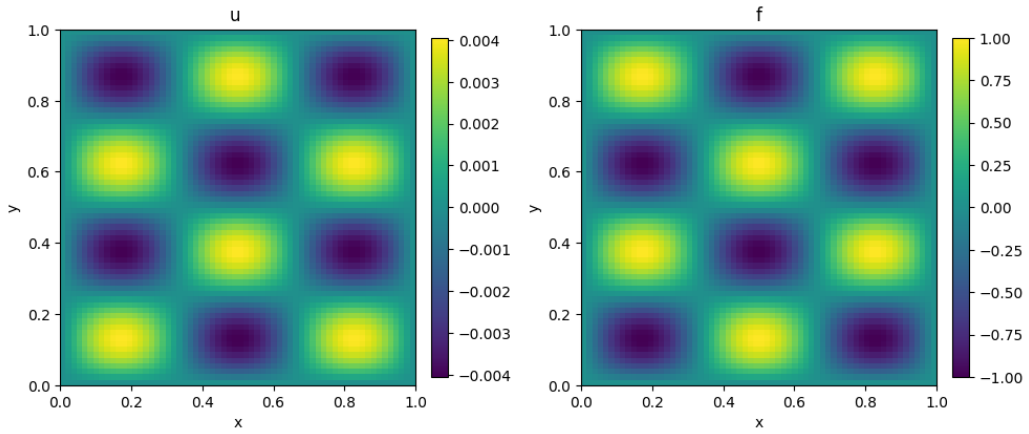


Figure 2: Example type 1 analytical solution with $n = 3$ and $k = 4$.

TYPE 2 ANALYTICAL PAIRS

$$u(x, y) = \sin(n\pi x)\sin(k\pi y)\sin(j\pi x)$$

for n, k, j positive integers. The analytical solution for f is given by

$$\nabla^2 u = f = -\pi^2(-2jn\cos(j\pi x)\cos(n\pi x) + (j^2 + k^2 + n^2)\sin(j\pi x)\sin(n\pi x))\sin(k\pi y).$$

An example solution is shown in Fig. 3.

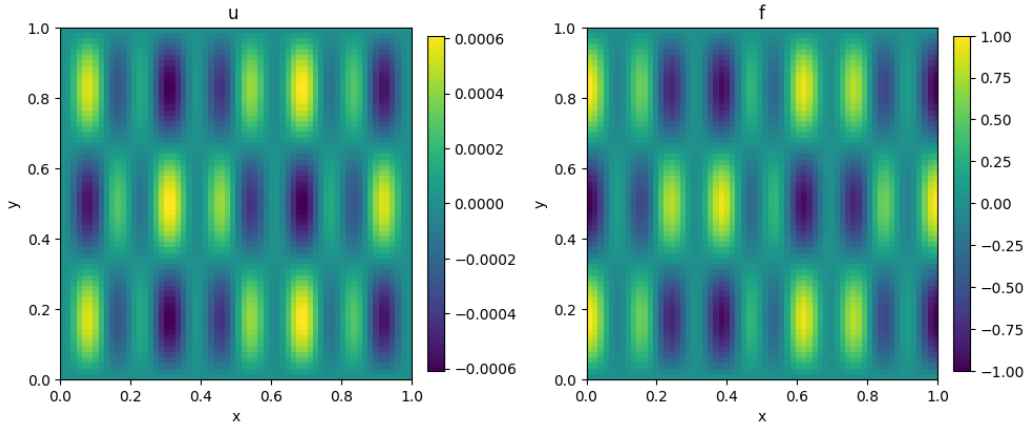


Figure 3: Example type 2 analytical solution with $n = 5$, $k = 3$, and $j = 8$.

TYPE 3 ANALYTICAL PAIRS

$$u(x, y) = \sin(n\pi x)\sin(k\pi y)\cos(n\pi x)$$

for n, k positive integers. The analytical solution f is given by

$$\nabla^2 u = f = -\frac{1}{2}(k^2 + 4n^2)\pi^2 \sin(2n\pi x)\sin(k\pi y).$$

An example solution is shown in Fig. 4.

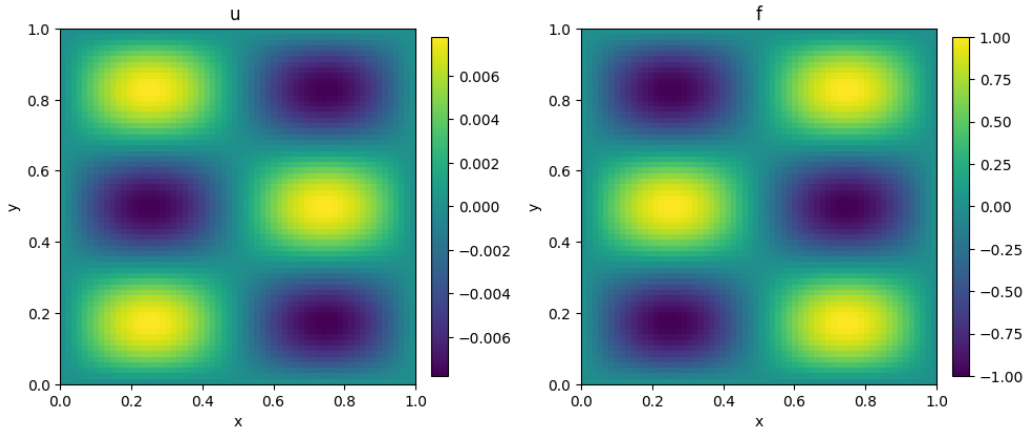


Figure 4: Example type 3 analytical solution with $n = 1$ and $k = 3$.

TYPE 4 ANALYTICAL PAIRS

$$u(x, y) = \sin(n\pi x)\sin(k\pi y)\cos(j\pi x)$$

for n, k, j positive integers. The analytical solution f is given by

$$\nabla^2 u = f = -\pi^2(2jn\cos(n\pi x)\sin(j\pi x) + (j^2 + k^2 + n^2)(\cos(j\pi x)\sin(n\pi x))\sin(k\pi y).$$

An example solution is shown in Fig. 5.

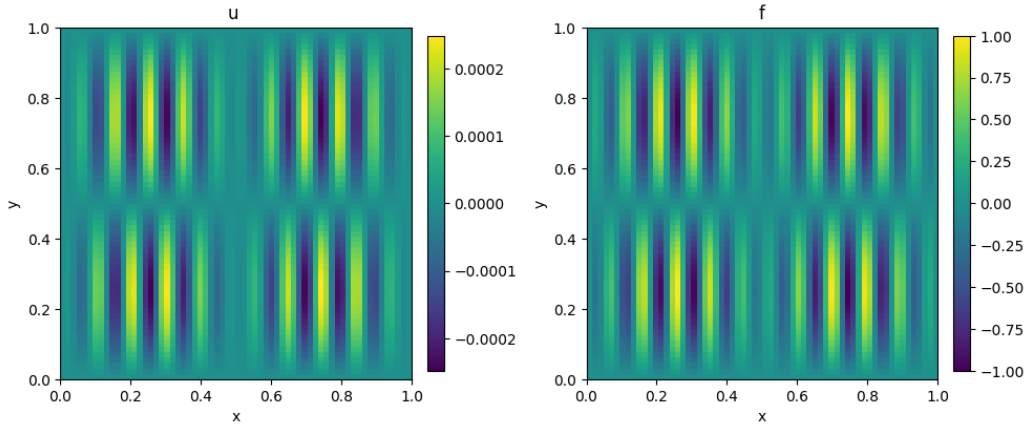


Figure 5: Example type 4 analytical solution with $n = 2$ and $k = 2$.

TYPE 5 ANALYTICAL PAIRS

$$u(x, y) = n(x - 1)x(y - 1)y \exp(x - y)$$

for n positive integers. The analytical solution f is given by

$$\nabla^2 u = f = 2 \exp(x - y)nx(y - 1)(2 + x(y - 2) + y)$$

An example solution is shown in Fig. 6.

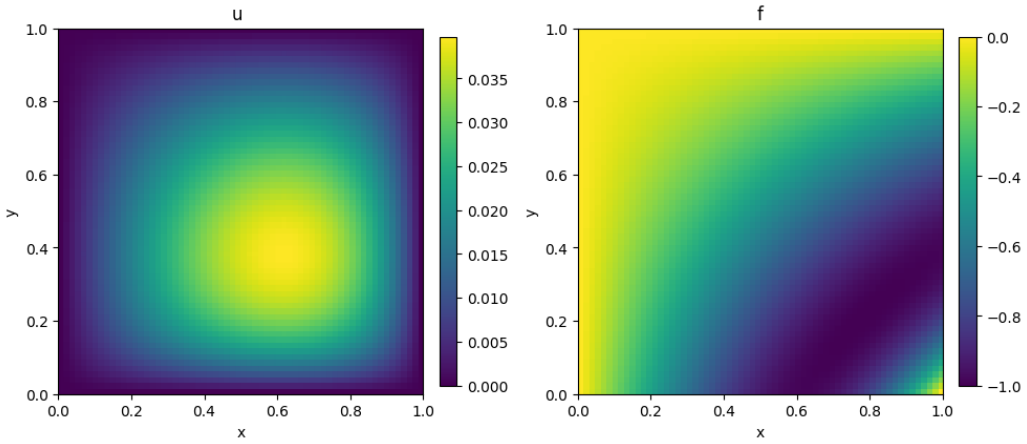


Figure 6: Example type 5 analytical solution with $n = 8$.

TYPE 6 ANALYTICAL PAIRS

$$u(x, y) = n(x - 1)x(y - 1)y \exp(y - x)$$

for n positive integers. The analytical solution f is given by

$$\nabla^2 u = f = 2n \exp(y - x)y(x - 1)(2 + x - 2y + xy).$$

An example solution is shown in Fig. 7.

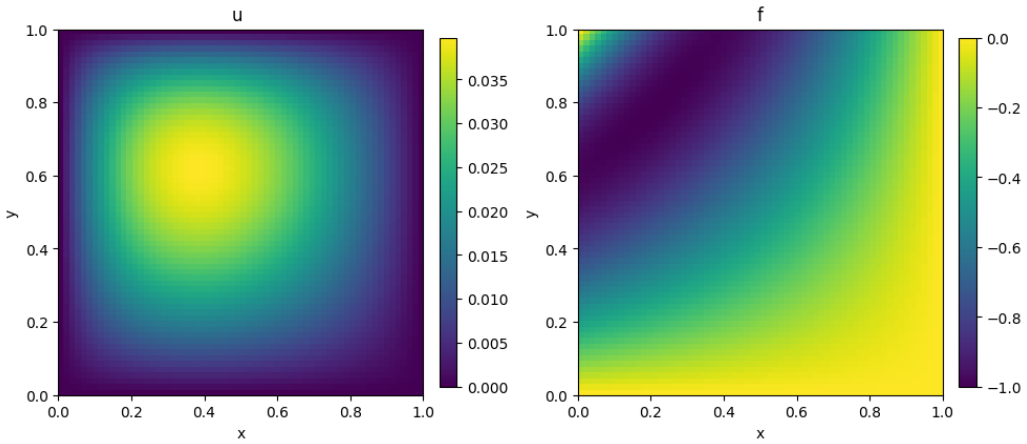
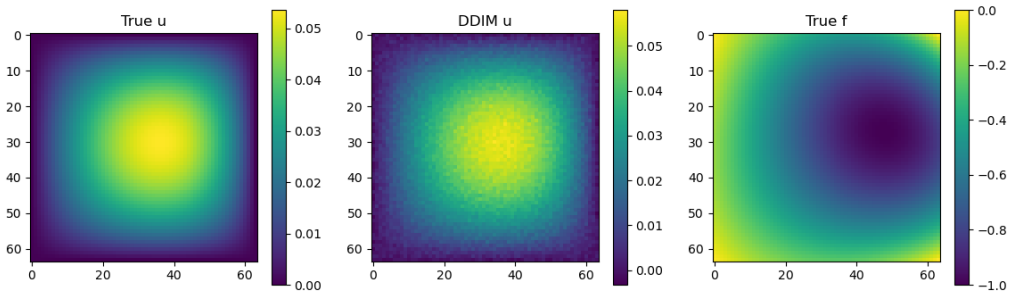


Figure 7: Example type 6 analytical solution with $n = 8$.

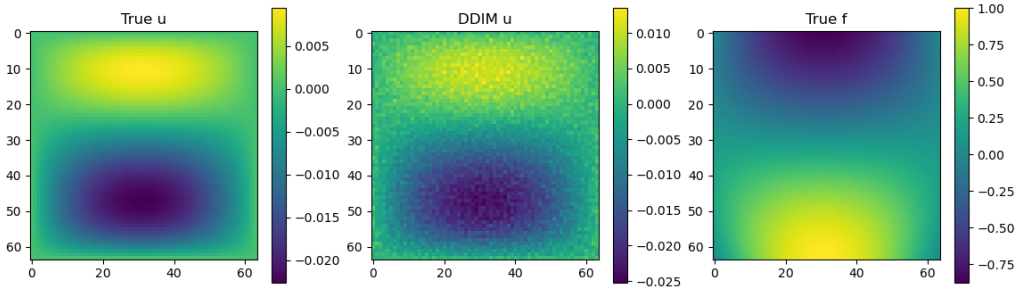
Appendix C. Qualitative results - Dry Forward Process

We trained a DDIM on the dataset explained in section 4.2 and appendix B. In this section, we estimate solutions to the Poisson equation $u(x, y)$ while keeping the $f(x, y)$ channel fixed. We tested this on different randomly generated neural network functions explained in section B.1, three of which are plotted in figure 8.

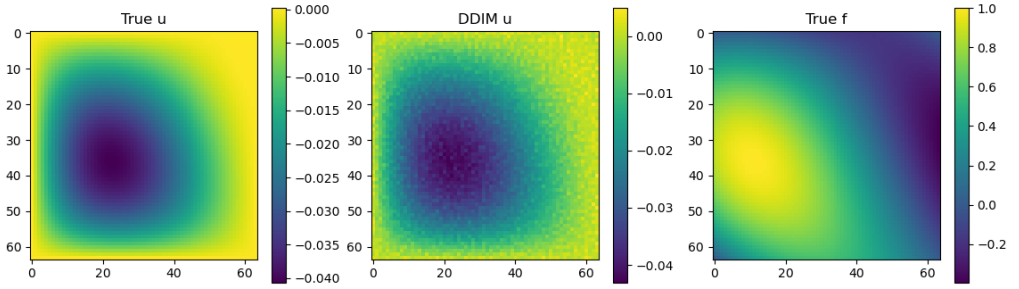
The plots show that the DDIM produces a good approximation of the solution to the Poisson equation, with some additive noise. The average MAE over 64 different test samples is 0.001123.



(*a*)



(*b*)



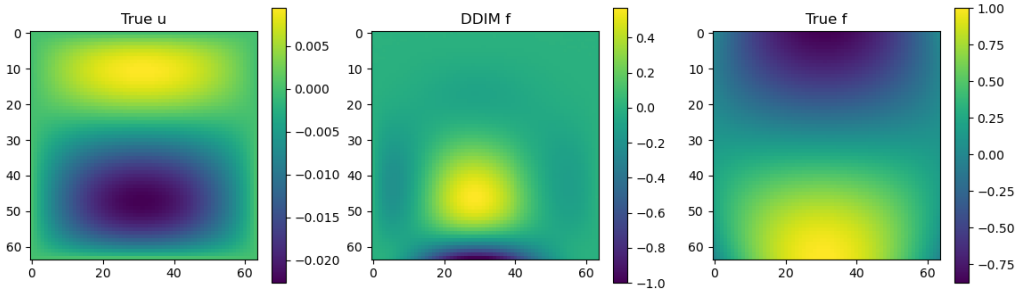
(*c*)

Figure 8: Plots of dry generated solutions of the forward process $u(x, y)$ (middle), true solution $u(x, y)$ (left), and $f(x, y)$ (right).

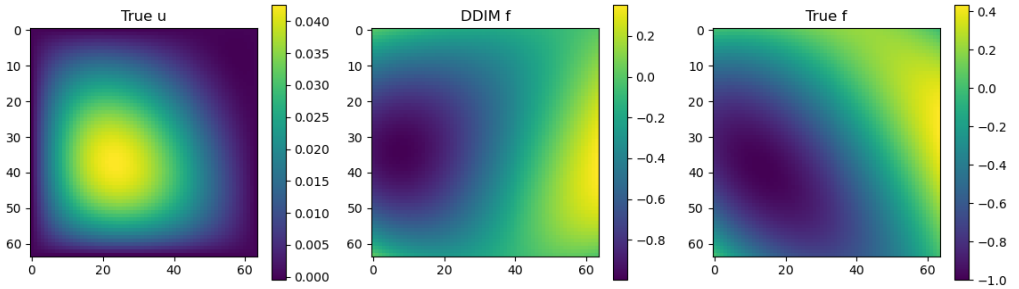
Appendix D. Qualitative Results - Dry Inverse Process

In this section, we estimate the parameter $f(x, y)$ while keeping the $u(x, y)$ channel fixed. We tested this on different randomly generated neural network functions explained in section B.1, three of which are plotted in figure 9.

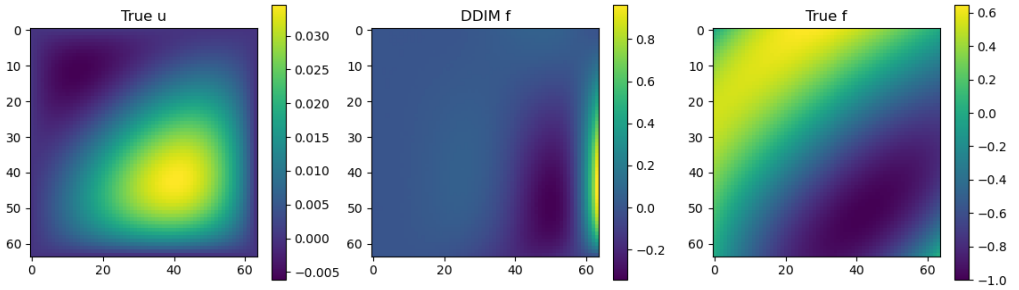
The plots show that the DDIM produces a poor approximation of the solution to the Poisson equation. The average MAE over 64 different test samples is 0.5515.



(*a*)



(*b*)



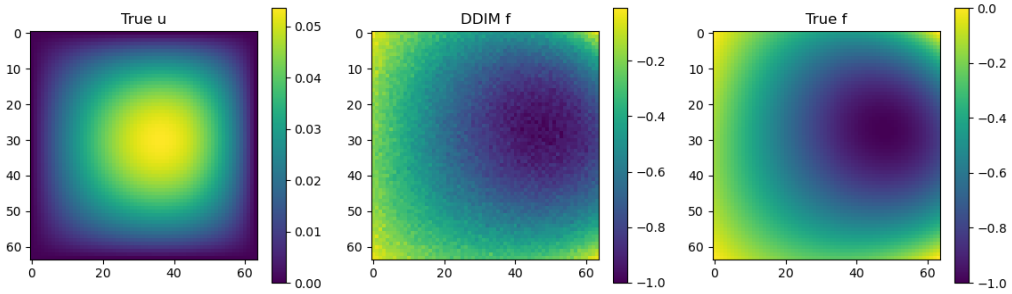
(*c*)

Figure 9: Plots of dry generated solutions of the inverse process $f(x, y)$ (middle), $u(x, y)$ (left), and the true $f(x, y)$ (right).

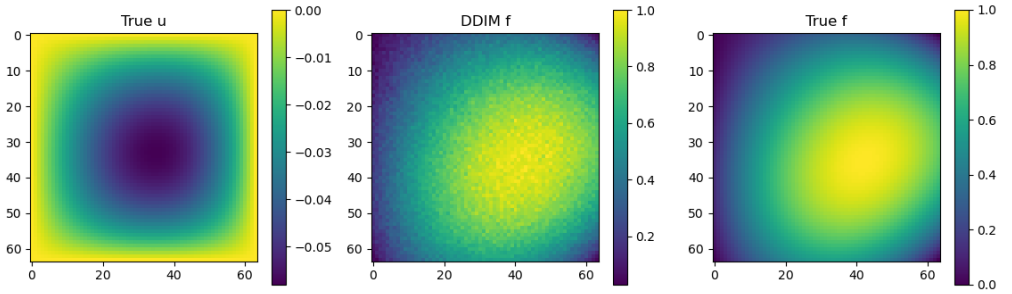
Appendix E. Qualitative Results - DDRM Inverse Process

In this section, we estimate the parameter $f(x, y)$ while keeping the $u(x, y)$ channel fixed using DDRM. We tested this on different randomly generated neural network functions explained in section B.1, three of which are plotted in figure 10.

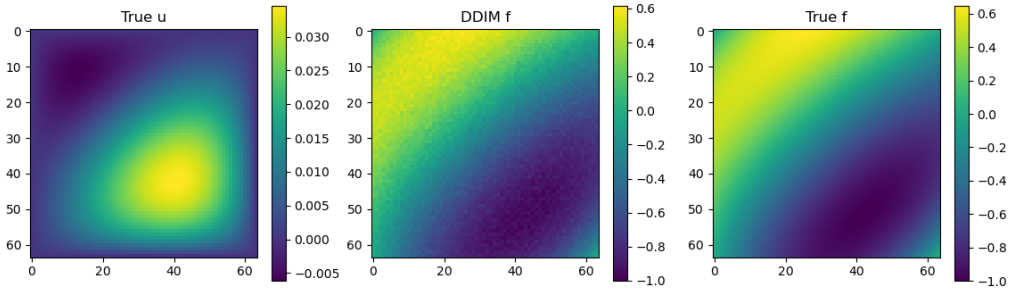
The plots show that the DDRM produces a significantly better approximation of the solution to the Poisson equation. The average MAE over 64 different test samples is 0.03215.



(*a*)



(*b*)



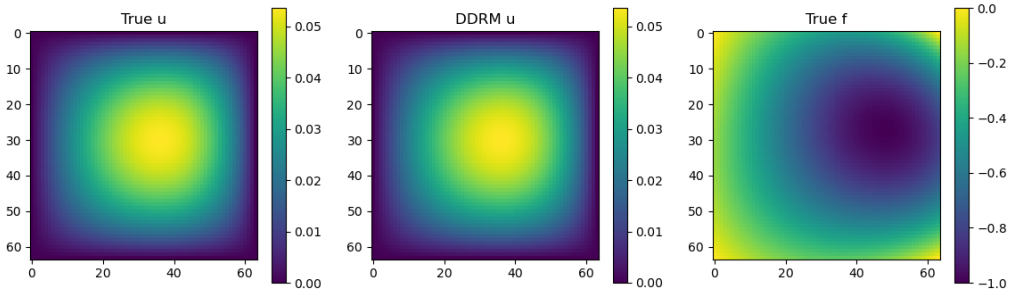
(*c*)

Figure 10: Plots of DDRM generated solutions of the inverse process $f(x, y)$ (middle), $u(x, y)$ (left), and the true $f(x, y)$ (right).

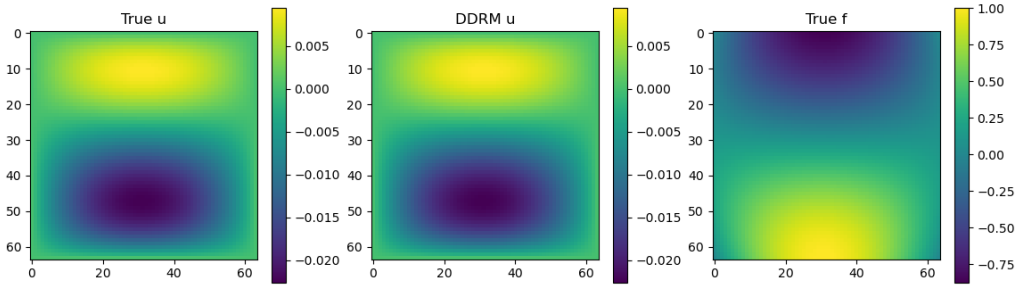
Appendix F. Qualitative Results - DDRM Forward Process

In this section, we estimate the solution $u(x, y)$ while keeping the parameter channel $f(x, y)$ fixed using DDRM. We tested this on different randomly generated neural network functions explained in section B.1, three of which are plotted in figure 11.

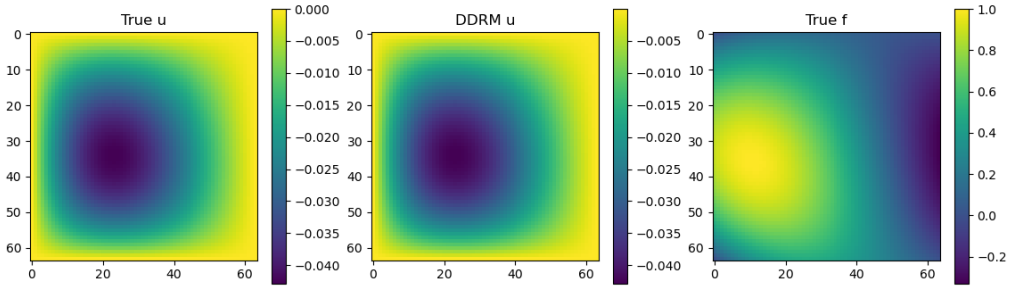
The plots show that the DDRM produces a significantly better approximation of the solution to the Poisson equation. The average MAE over 64 different test samples is $9.675e-07$, which is just slightly greater than the MAE of $6.672e-07$ upon using the finite difference method.



(*a*)



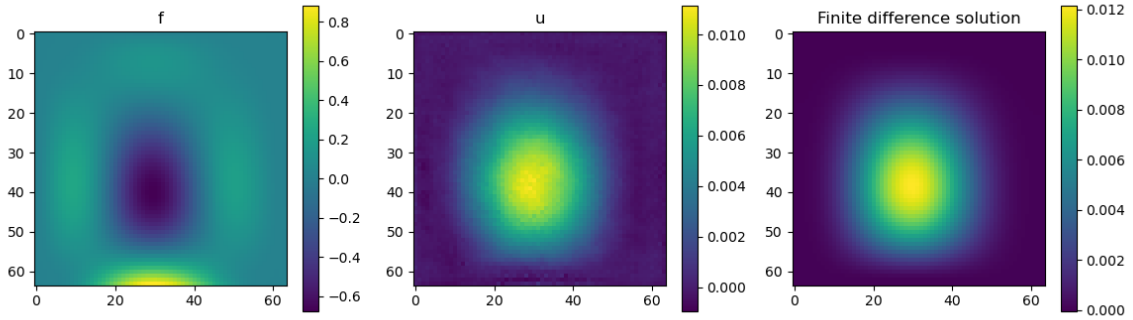
(*b*)



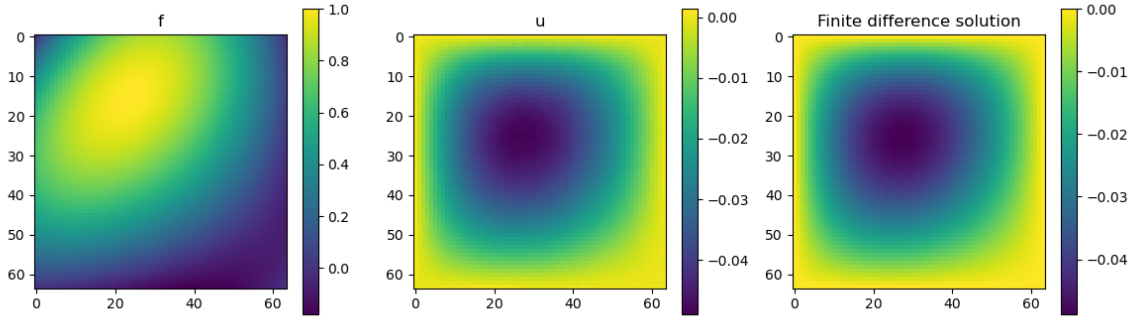
(*c*)

Figure 11: Plots of DDRM generated solutions of the forward process $u(x, y)$ (middle), true $u(x, y)$ (left), and $f(x, y)$ (right).

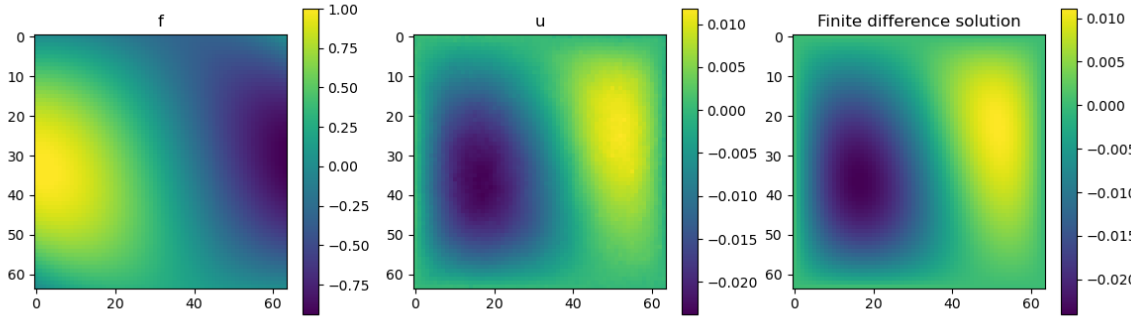
Appendix G. Unconditionally Generated Data



(*a*)



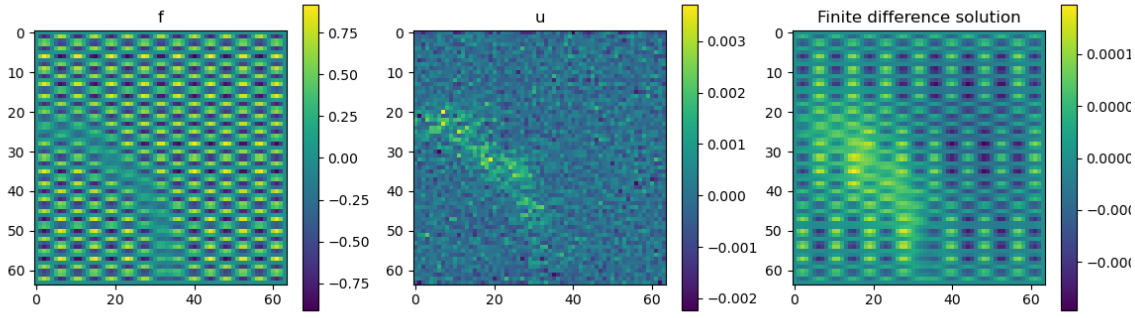
(*b*)



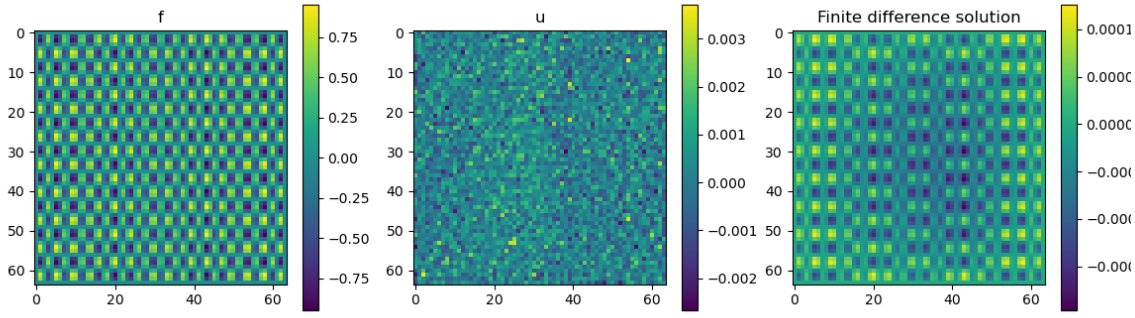
(*c*)

Figure 12: Plots of unconditionally generated pairs of $f(x, y)$ (left), $u(x, y)$ (middle), and the finite difference solution of $u(x, y)$ (right).

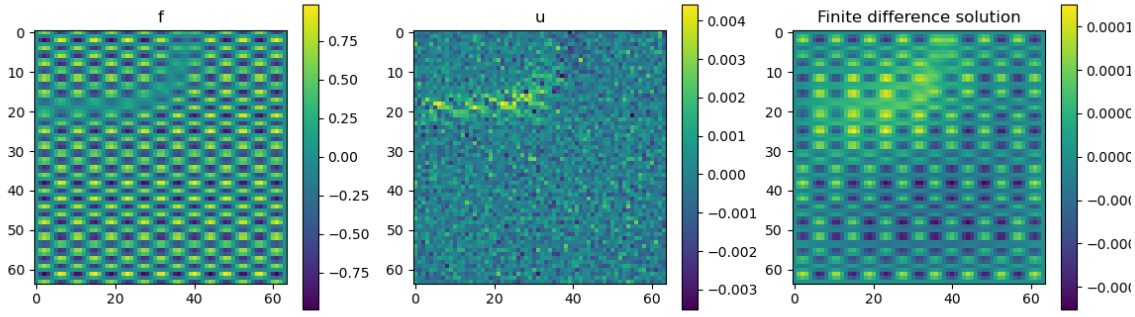
Appendix H. High frequency unconditionally generated data



(*a*)



(*b*)



(*c*)

Figure 13: Plots of unconditionally generated high frequency pairs of $f(x, y)$ (left), $u(x, y)$ (middle), and the finite difference solution of $u(x, y)$ (right).