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 forward problems in the Laplacian operator, for instance, the Poisson equation, because the eigenvalues that are large in magnitude amplify the measurement noise. This paper presents a novel approach for the forward and 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 solution and the parameters in the Poisson equation 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 solution and parameters. Our research, as a result, pioneers the integration of diffusion models with the principles of underlying physics to solve PDEs.

Keywords: Denoising Diffusion Restoration Models, Laplace Operator, Physics-Informed Machine Learning, Inverse Problems

1. Introduction

Denoising diffusion models are among the current leading methods for generative modeling (Ho et al., 2020; Song et al., 2021). They 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 (e.g., 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). Furthermore, they are stable to train and are relatively easy to scale.

The Laplace operator is a differential operator of second order, $\Delta = \nabla \cdot \nabla$ (Gilbarg and Trudinger, 1983), which 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). This paper concerns two problems involving the Laplace operator defined in the domain $\Omega = [0, 1]^2$; (i) the forward problem, where we are given a function $u \in C^2(\Omega)$ with $u = 0$ in the boundary, $\partial\Omega$, and we intend to compute Δu, and (ii) the inverse problem, where we are given a function $f \in C(\Omega)$ and we intend to compute u satisfying $\Delta u = f$ and $u = 0$ in $\partial\Omega$.

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 and are often computationally expensive, 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.

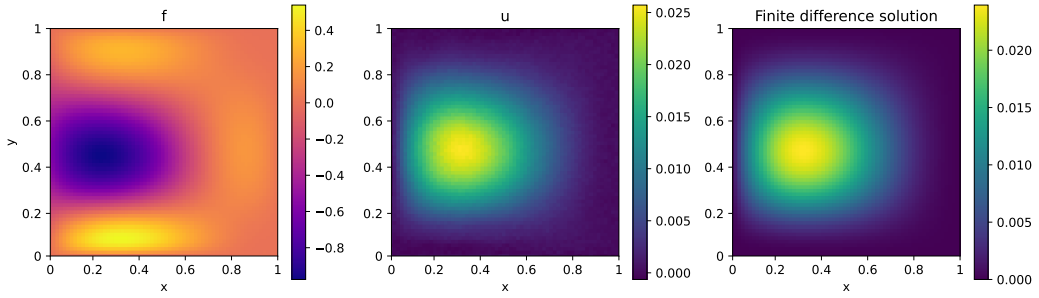
In more recent works, Apte et al. 2023 seek to address the problem of data scarcity for machine learning methods of PDE modeling by developing a method of data generation using a diffusion model. By training on data from the steady 2D Poisson equation on a fixed square domain, they made diffusion models generate paired data samples that adhered to physics laws, despite not including physics in 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. The work of Ovadia et al. (2023) also uses diffusion models in conjunction with vision transformers to model time-dependent PDEs. We intend to build upon these works by adding conditions on u or f drawn from a test set (and hence unseen by the diffusion model during training). Furthermore, contrary to these past works, we exploit the physics of the Laplace operator to derive eigenvalue and eigenfunction pairs that we project u and f onto.

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 inverse process. Our numerical results are posted in appendix H, 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 forward process. Our numerical results are posted in appendix G and demonstrate that the DDIM model is a poor numerical solver for the forward problem. Therefore, we require a better method to solve the forward problem.

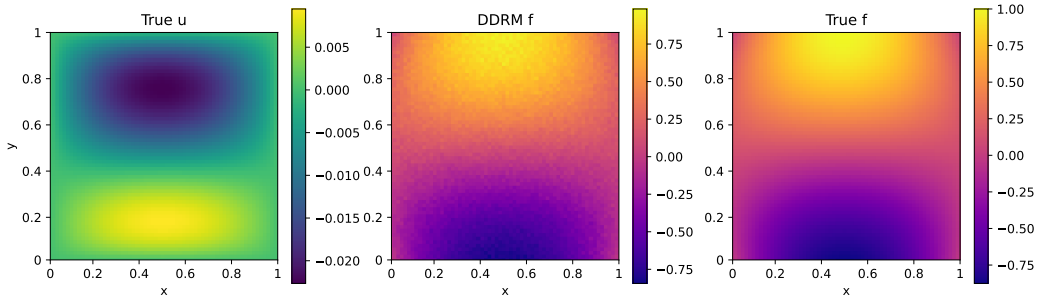
To solve this problem, we employ denoising diffusion restoration models (DDRMs) (Kawar et al., 2022; Chung et al., 2023; Murata et al., 2023). DDRMs are used to restore clean data in linear inverse problems using a pre-trained diffusion model without requiring any fine-tuning. The authors achieve this 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 forward and 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 1.175×10^{-6} in our improved forward process, which is just slightly greater than the MAE of 6.672×10^{-7} upon using the finite difference method, thus showing a significant improvement by using DDRM. Our results are briefly shown in

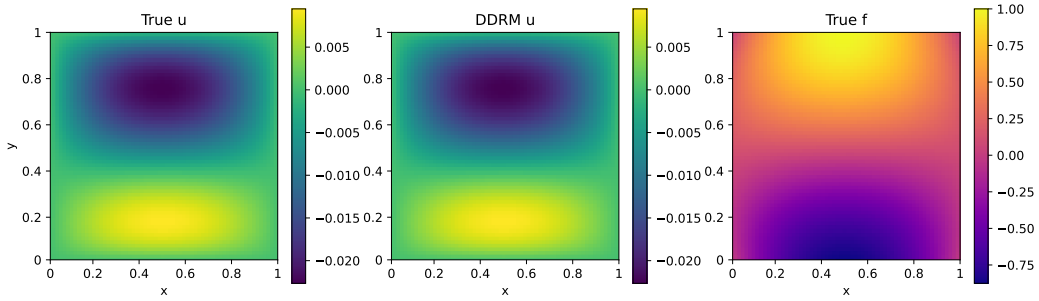
Figure 1. Our work outperforms other data-driven benchmarks as well, and is the first to do so by including the physics in diffusion models.



(a) Unconditionally generated pairs



(b) Forward Problem - Restoring f conditioned on u with DDRM



(c) Inverse Problem - Restoring u conditioned on f with DDRM

Figure 1: Plots of DDRM generated solutions of the inverse process $f(x, y)$ (middle), $u(x, y)$ (left), and the true $f(x, y)$ (right).

2. Problem statement and conditional distributions

We will start by describing the forward and inverse problems associated with the Laplacian operator by considering the Poisson equation defined on two spatial dimensions

$$\Delta u = f \tag{1}$$

where the domain is $\Omega = [0,1]^2$ with homogenous Dirichlet boundary conditions $u = 0$ on $\partial\Omega$. The forward problem concerns computing f conditioned on u, and the inverse problem concerns computing u conditioned on f – in other words, solving the PDE.

Forward problem

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

$$\begin{aligned} f_0 &= \Delta u_0, \\ u &= u_0 + z_u, \end{aligned} \tag{2}$$

where 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 $\partial\Omega$. $z_u(x,y)$ is a Brownian bridge satisfying the same boundary conditions as u_0. u is the observed signal with measurement noise from which we need to estimate the parameter f_0.

As shown in appendix G, a pre-trained diffusion model fails to retrieve the parameter $f_0(x,y)$ conditioned on $u(x,y)$ if we use a dry forward process.

Inverse problem

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

$$f = \Delta u_0 + z_f \tag{3}$$

where $z_f \sim \mathcal{N}(0, \sigma_u^2)$ is measurement noise with known covariance σ_u^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 H, a pre-trained diffusion model is effective in retrieving the solution $u_0(x,y)$ conditioned on the parameter $f(x,y)$ through a dry inverse process. However, there is a significant amount of noise in these estimated solutions, thus increasing the MAE.

In this section, we will introduce some prior work that assists us in deriving an improved method to restore $f(x,y)$ and $u(x,y)$.

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. DDRM is defined as a Markov chain $x_T \to x_{T-1} \to \cdots \to x_1 \to x_0$ conditioned on y:

$$p(x_{0:T} \mid y) = p_\theta^{(T)}(x_T \mid y) \prod_{t=0}^{T-1} p_\theta^{(t)}(x_t \mid x_{t+1}, y). \tag{4}$$

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. 2 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 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*

$$u_{n,m}(x, y) = \sin(n\pi x) \sin(m\pi y) \tag{5}$$

$$\lambda_{n,m} = -(n\pi)^2 - (m\pi)^2 \tag{6}$$

For completeness, a proof of this known proposition is provided in appendix A. This proposition will be useful in our derivation of the modified DDRM algorithm. We are projecting the functions u and f into the eigenfunctions of the Laplacian operator, thus allowing us to solve them on a shared spectral space. The proposition also explains the numerical results shown in appendix G and H. Due to the large magnitude of the eigenvalues, the Laplacian operator minimizes the measurement noise in the inverse process, but it amplifies the noise measurement noise in the forward process, thus making f_0 harder to compute.

2.3. Introducing conditional distributions

In the forward and inverse problem with the Laplacian operator, it is important to note that assuming that the measurement noise is i.i.d along the whole grid may not be realistic. Recognizing that the distribution of the noise varies along x and y is important to sample solutions that are consistent with our knowledge of the problem, such as the PDE and the boundary conditions. Naturally, it makes sense that our solution has the highest uncertainty along the center of the domain (i.e. points near $(0.5, 0.5)$).

2.3.1. FORWARD PROBLEM

We define the noise z_u as a Brownian bridge satisfying the homogenous boundary conditions $z_u = 0$ on $\partial\Omega$. A common approach is to express z_u as a double sum of sinusoidal functions (satisfying the boundary conditions)

$$z_u = \sum_{n=0}^{N} \sum_{m=0}^{N} w_{n,m} \sin(n\pi x) \sin(m\pi y), \tag{7}$$

where $w_{n,m} \sim \mathcal{N}(0, \sigma_{n,m}^2)$ are random coefficients with known variance $\sigma_{n,m}^2$. Putting this together, z_u has the following distribution

$$z_u(x, y) \sim N\left(0, \sum_{n=0}^{N} \sum_{m=0}^{N} \sigma_{n,m}^2 \sin(n\pi x)^2 \sin(m\pi y)^2\right) \tag{8}$$

Three numerical simulations of this Brownian bridge have been plotted in figure 2 with $\sigma_{n,m} = 1e-5$ for all n, m.

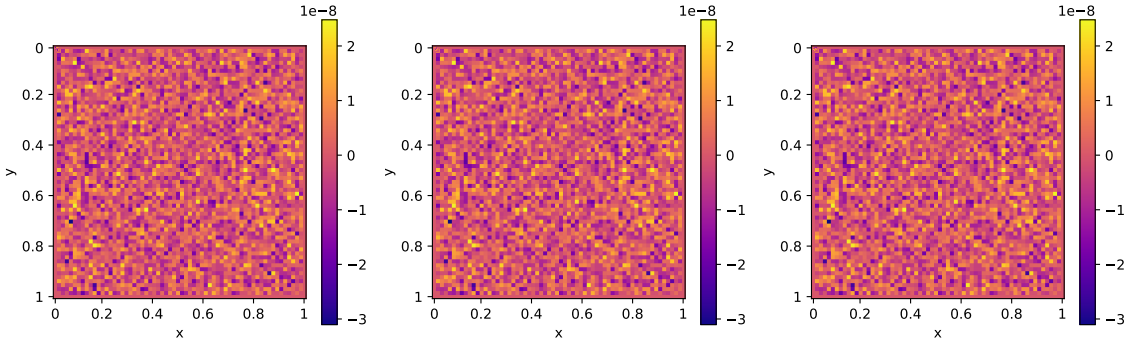


Figure 2: Numerical simulations of Brownian bridge with $\sigma_{n,m} = 1e-6$ for all n, m

2.3.2. INVERSE PROBLEM

We define the noise z_f as i.i.d Gaussian since we do not impose any assumptions on $f(x, y)$ other than continuity. This, however, does not apply to $u(x, y)$. We introduce theorem 2 that models its distribution along (x, y).

Theorem 2 *If $f = \Delta u + z_f$ where $z_f \sim \mathcal{N}(0, \sigma_f^2)$, then the marginal distribution of u conditioned on f is*

$$u(x, y) \mid f_0 \sim \mathcal{N}(u_0, \sigma_f^2 K(x, y)), \tag{9}$$

where

$$K(x, y) = \iint_{\Omega} (\psi((x', y') - (x, y)))^2 dx' dy', \tag{10}$$

where $\psi(\cdot)$ is the Green's function in two dimensions

$$\psi(x, y) = \frac{\ln(\|(x, y)\|)}{2\pi} \tag{11}$$

A proof of the theorem is provided in appendix B. This theorem will be very important in our DDRM algorithm for sampling $u(x, y)$ conditioned on $f(x, y)$.

In the development of our algorithm, we are interested in the distribution of the discrete sine transform of $u(x, y)$. Computing this distribution is expensive, so we introduce a theorem that places an upper bound on the variance of the discrete sine transform of $u(x, y)$.

Theorem 3 *Let $f = \Delta u + z_f$ where $z_f \sim \mathcal{N}(0, \sigma_f^2)$. Consider the discrete sine transform (DST) of $u(x, y)$*

$$\overline{\mathbf{u}}^{(n,m)} = \langle \mathbf{u}, \sin(n\pi x) \sin(m\pi y) \rangle. \tag{12}$$

We can place an upper bound on the variance of $\overline{\mathbf{u}}^{(n,m)}$.

$$Var[\overline{\mathbf{u}}^{(n,m)} | f_0] \leq \left(\frac{1}{\pi^2 (n^2 + m^2)} + \ln 2 \max(n, m) \right)^2 \sigma_f^2 \tag{13}$$

A proof of the theorem is provided in appendix C. This theorem follows from theorem 2 and helps us define distributions to sample $u(x, y)$ conditioned on $f(x, y)$. Notice that the coefficient of the variance term can be computed analytically, thus significantly reducing the computation cost of our algorithm compared to computing $K(x, y)$ using numerical integration.

3. DDRM for solving PDEs

3.1. Forward process: Sampling f conditioned on u

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 $\mathbf{u}$:

$$p_\theta(\mathbf{f}_{0:T} \mid \mathbf{u}) = p_\theta^{(T)}(\mathbf{f}_T \mid \mathbf{u}) \prod_{t=0}^{T-1} p_\theta^{(t)}(\mathbf{f}_t \mid \mathbf{f}_{t+1}, \mathbf{u}) \tag{14}$$

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. Similarly to their work, we will consider the variational distribution conditioned on $\mathbf{u}$.

$$q(\mathbf{f}_{1:T} \mid \mathbf{f}_0, \mathbf{u}) = p_\theta^{(T)}(\mathbf{f}_T \mid \mathbf{f}_0, \mathbf{u}) \prod_{t=0}^{T-1} p_\theta^{(t)}(\mathbf{f}_t \mid \mathbf{f}_{t+1}, \mathbf{f}_0, \mathbf{u}) \tag{15}$$

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:

$$\overline{\mathbf{u}}^{(n,m)} = \lambda_{n,m} \langle \mathbf{u}, \sin(n\pi x) \sin(m\pi y) \rangle, \tag{16}$$

$$\overline{\mathbf{f}}_t^{(n,m)} = \langle \mathbf{f}_t, \sin(n\pi x) \sin(m\pi y) \rangle, \tag{17}$$

where $\lambda_{n,m}$ are the eigenvalues from proposition 1. Since none of the eigenvalues $\lambda_{m,n}$ are zero, we can defined the variational distribution for $\mathbf{f}_T$ for each index n, m in $\overline{\mathbf{f}}_T^{(n,m)}$ as:

$$q^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \mathbf{f}_0, \mathbf{u}) = \mathcal{N}(\overline{\mathbf{u}}^{(n,m)}, \sigma_T^2 - \sigma_{n,m}^2 \lambda_{n,m}^2), \tag{18}$$

where $\sigma_{n,m}$ are defined as the standard deviation of $w_{n,m}$ in equation 7. We assume that $\sigma_T > \sigma_{m,n}\lambda_{n,m}$ for all n, m. We can also define the variational distribution for $\mathbf{f}_t$ for each index n, m in $\overline{\mathbf{f}}_t^{(n,m)}$ as:

$$q^{(t)}\left(\overline{\mathbf{f}}_t^{(n,m)} \mid \mathbf{f}_0, \mathbf{f}_{t+1}, \mathbf{u}\right) = \begin{cases} \mathcal{N}\left(\overline{\mathbf{f}}_0^{(n,m)} + \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{u}}^{(n,m)} - \overline{\mathbf{f}}_0^{(n,m)}}{\sigma_{n,m}\lambda_{n,m}}, \eta^2\sigma_t^2\right), & \text{if } \sigma_t < \sigma_{n,m}\lambda_{n,m} \\ \mathcal{N}\left((1-\eta_b)\,\overline{\mathbf{f}}_0^{(n,m)} + \eta_b \overline{\mathbf{u}}^{(n,m)}, \sigma_t^2 - \sigma_{n,m}^2\lambda_{m,n}^2\eta_b^2\right), & \text{if } \sigma_t \geq \sigma_{n,m}\lambda_{n,m} \end{cases}, \tag{19}$$

where $\eta, \eta_b \in [0, 1]$ are hyperparameters controlling the variance of the distributions. We introduce a proposition that verifies the convergence of this variational distribution.

Proposition 4 *(Modified version of Proposition 3.1 from Kawar et al. (2022)) The conditional distributions defined in equations 18 and 19 satisfy the following Gaussian marginal property:*

$$q^{(t)}\left(\overline{\mathbf{f}}_t^{(n,m)} \mid \mathbf{f}_0\right) = \mathcal{N}(\overline{\mathbf{f}}_0^{(n,m)}, \sigma_{n,m}^2) \tag{20}$$

The proof of this proposition is in Appendix D. Based on the formulation of the conditional distribution in section 2.3.1, this proposition shows that the transitions $\mathbf{f}_t$ converge in distribution to the distribution of $\mathbf{f}_0$.

Sampling of $\mathbf{f}_T$.

The sampling of $\mathbf{f}_T$ is performed by sampling from the distribution $p(\mathbf{f}_T \mid \mathbf{u})$. Sampling from this conditional distribution is intractable, so we use our modified DDRM to approximate the distribution. Since none of the eigenvalues $\lambda_{m,n}$ are zero, our DDRM method for sampling $\mathbf{f}_T$ is as follows:

$$p_\theta^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \mathbf{u}) = \mathcal{N}(\overline{\mathbf{u}}^{(n,m)}, \sigma_T^2 - \sigma_{n,m}^2 \lambda_{n,m}^2) \tag{21}$$

In this process, we assume that σ_T is sufficiently large to satisfy $\sigma_T > \sigma_{n,m}((n\pi)^2 + (m\pi)^2)$ for all $n, m \in \{1, ..., 64\}$, so that the variance term is non-negative. This distribution is identical to the variational distribution from 18.

Sampling of $\mathbf{f}_t$.

The sampling of $\mathbf{f}_t$ is performed by sampling from the distribution $p(\mathbf{f}_t \mid \mathbf{f}_{t+1}, ..., \mathbf{f}_T, \mathbf{u})$. 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:

$$p_\theta^{(t)}\left(\overline{\mathbf{f}}_t^{(n,m)} \mid \mathbf{f}_{t+1}, \mathbf{u}\right) =$$
$$\begin{cases} \mathcal{N}\left(\overline{\mathbf{f}}_{\theta,t}^{(n,m)} + \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{u}}^{(n,m)} - \overline{\mathbf{f}}_{\theta,t}^{(n,m)}}{\sigma_{n,m}\lambda_{n,m}}, \eta^2\sigma_t^2\right), & \text{if } \sigma_t < \sigma_{n,m}\lambda_{n,m} \\ \mathcal{N}\left((1-\eta_b)\,\overline{\mathbf{f}}_{\theta,t}^{(n,m)} + \eta_b \overline{\mathbf{u}}^{(n,m)}, \sigma_t^2 - \sigma_{n,m}^2\lambda_{m,n}^2\eta_b^2\right), & \text{if } \sigma_t \geq \sigma_{n,m}\lambda_{n,m} \end{cases} \tag{22}$$

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. This distribution is obtained by modifying the variational distribution from 19. Since $\mathbf{f}_0$ is unknown at time step t, we need to use our pre-trained diffusion model to approximate it, which we denote as $\mathbf{f}_{\theta,t}$.

3.2. Inverse process: Sampling u conditioned on f

In this process, we consider the DST of $\mathbf{u}_t$ and $\mathbf{f}$ and perform the diffusion in its spectral space. Define $\overline{\mathbf{u}}_t^{(n,m)}$, $\overline{\mathbf{f}}^{(n,m)}$, and $\overline{\mathbf{K}}^{(n,m)}$ as follows:

$$\overline{\mathbf{u}}_t^{(n,m)} = \langle \mathbf{u}_t, \sin(n\pi x)\sin(m\pi y)\rangle, \tag{23}$$

$$\overline{\mathbf{f}}^{(n,m)} = \langle \mathbf{f}, \sin(n\pi x)\sin(m\pi y)\rangle / \lambda_{n,m}, \tag{24}$$

where $\lambda_{n,m}$ are the eigenvalues from proposition 1. To clarify the distinction from the definitions in 16,17, note which variable has a subscript of t. We also define $\overline{\mathbf{K}}^{(n,m)}$ as the variance term of $\overline{\mathbf{u}}_t^{(n,m)}$ introduced in theorem 3:

$$\overline{\mathbf{K}}^{(n,m)} = \left(\frac{1}{|\lambda_{n,m}|} + \ln 2 \max(n,m)\right)^2. \tag{25}$$

$\overline{\mathbf{K}}^{(n,m)}$ has a maximum of 1967.938 corresponding to $n = 1, m = 64$. Since none of the eigenvalues $\lambda_{m,n}$ are zero, we can defined the variational distribution for $\mathbf{u}_T$ for each index n, m in $\overline{\mathbf{u}}_T^{(n,m)}$ as:

$$q^{(T)}(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_0, \mathbf{f}) = \mathcal{N}(\overline{\mathbf{f}}^{(n,m)}, \sigma_T^2 - \sigma_f^2 \overline{\mathbf{K}}^{(n,m)}/\lambda_{n,m}^2), \tag{26}$$

where σ_f are defined as the standard deviation of $w_{n,m}$ in equation 7. We assume that $\sigma_T > \sigma_f\sqrt{\overline{\mathbf{K}}^{(n,m)}}/\lambda_{n,m}$ for all n, m. We can also define the variational distribution for $\mathbf{u}_t$ for each index n, m in $\overline{\mathbf{u}}_t^{(n,m)}$ as:

$$q^{(t)}\left(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_0, \mathbf{u}_{t+1}, \mathbf{f}\right) = \begin{cases} \mathcal{N}\left(\overline{\mathbf{u}}_0^{(n,m)} + \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{f}}^{(n,m)} - \overline{\mathbf{u}}_0^{(n,m)}}{\sigma_f\sqrt{\overline{\mathbf{K}}^{(n,m)}}/\lambda_{n,m}}, \eta^2\sigma_t^2\right), & \sigma_t < \frac{\sigma_f\sqrt{\overline{\mathbf{K}}^{(n,m)}}}{\lambda_{n,m}} \\ \mathcal{N}\left(\left(1-\eta_b\right)\overline{\mathbf{u}}_0^{(n,m)} + \eta_b\overline{\mathbf{f}}^{(n,m)}, \sigma_t^2 - \frac{\sigma_f^2\overline{\mathbf{K}}^{(n,m)}}{\lambda_{n,m}^2}\eta_b^2\right), & \sigma_t \geq \frac{\sigma_f\sqrt{\overline{\mathbf{K}}^{(n,m)}}}{\lambda_{n,m}} \end{cases} \tag{27}$$

where $\eta, \eta_b \in [0, 1]$ are hyperparameters controlling the variance of the distributions. We introduce a proposition that verifies the convergence of this variational distribution.

Proposition 5 *(Modified version of Proposition 3.1 from Kawar et al. (2022)) The conditional distributions defined in equations 26 and 27 satisfy the following Gaussian marginal property:*

$$q^{(t)}\left(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_0\right) = \mathcal{N}(\overline{\mathbf{u}}_0^{(n,m)}, \sigma_f^2\overline{\mathbf{K}}^{(n,m)}) \tag{28}$$

The proof of this proposition is in Appendix E. Based on the formulation of the conditional distribution in section 2.3.2, this proposition shows that the transitions $\mathbf{u}_t$ converge in distribution to the distribution of $\mathbf{u}_0$.

Sampling of $\mathbf{u}_T$.

The sampling of $\mathbf{u}_T$ is performed by sampling from the distribution $p(\mathbf{u}_T \mid \mathbf{f})$. Sampling from this conditional distribution is intractable, so we use our modified DDRM to approximate the distribution. Since none of the eigenvalues $\lambda_{m,n}$ are zero, our DDRM method for sampling x_T is as follows:

$$p_\theta^{(T)}(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{f}) = \mathcal{N}(\overline{\mathbf{f}}^{(n,m)}, \sigma_T^2 - \sigma_f^2\overline{\mathbf{K}}^{(n,m)}/\lambda_{n,m}^2) \tag{29}$$

For this sampling to be tractable, we choose an appropriate value for σ_T such that $\sigma_T > \sigma_f/(\pi\sqrt{2})$. This distribution is identical to the variational distribution from equation 26.

Sampling of $\mathbf{u}_t$.

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

$$p_\theta^{(t)}\left(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_{t+1}, \mathbf{f}\right) = \begin{cases} \mathcal{N}\left(\overline{\mathbf{u}}_{\theta,t}^{(n,m)} + \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{f}}^{(n,m)} - \overline{\mathbf{u}}_{\theta,t}^{(n,m)}}{\sigma_f\sqrt{\overline{\mathbf{K}}^{(n,m)}}/\lambda_{n,m}}, \eta^2\sigma_t^2\right), & \sigma_t < \frac{\sigma_f\sqrt{\overline{\mathbf{K}}^{(n,m)}}}{\lambda_{n,m}} \\ \mathcal{N}\left(\left(1-\eta_b\right)\overline{\mathbf{u}}_{\theta,t}^{(n,m)} + \eta_b\overline{\mathbf{f}}^{(n,m)}, \sigma_t^2 - \frac{\sigma_f^2\overline{\mathbf{K}}^{(n,m)}}{\lambda_{n,m}^2}\eta_b^2\right), & \sigma_t \geq \frac{\sigma_f\sqrt{\overline{\mathbf{K}}^{(n,m)}}}{\lambda_{n,m}} \end{cases} \tag{30}$$

This distribution is obtained by modifying the variational distribution from equation 27. Since $\mathbf{u}_0$ is unknown at time step t, we need to use our pre-trained diffusion model to approximate it, which we denote as $\mathbf{u}_{\theta,t}$.

4. Implementation

Before using DDRM to solve the forward and inverse problems, we had to train a diffusion model on a dataset of solutions of the Poisson equation. We replicated the method of PDE solution generation of the 2D Poisson equation with Dirichlet boundary conditions as done in Apte et al. (2023). In this section, we explain the dataset generation and the training of this model.

4.1. Dataset generation

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. 1 on a 64×64 grid on the domain Ω using a multigrid solver. We decided to instead generate a dataset of analytical solutions so that our diffusion model would not learn the numerical error associated with numerical solutions.

For our training dataset, we generated 38,250 samples for the diffusion model based on analytical solutions of Eq. 1. To do this we used (i) functions based on a neural network (appendix F.1) and (ii) analytical solutions by choosing differentiable u that satisfies the boundary condition and solving for f directly (appendix F.2). This data was used to train the diffusion model for the generation of PDE data.

4.2. Diffusion model

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 by 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.

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 4.1) and the training time was approximately 4 days.

Some of the unconditionally generated data are posted in appendix K. We compared generated pairs of u and f (labeled "u" and "f" in figure 14) with the finite difference solution (labeled "Finite difference solution" in figure 14) for comparison. Our MAE between the generated $u(x, y)$ and the finite difference solution is 4.373e-04, thus showing that the generated solutions are a good approximation to the true solution.

5. Numerical results

We conducted our experiments on the dataset described in section 4.1 using a trained DDIM model. Our test set is 1024 samples of neural network pairs described in appendix F.1 with seeds separate from the training set. To ensure the reproducibility of our results, we posted our code in the supplementary materials. Restoring a single batch (of 64 samples) took approximately 1 minute in a V100 GPU.

We have included results from other data-driven methods for solving PDEs - PINNs Raissi et al. (2019) and physics-informed DeepONet Lu et al. (2019). The results from PINNs were each trained directly on the test set. They were trained in the forward problem by learning to interpolate the grid points in $u(x, y)$, and we compute $f(x, y)$ by computing its Laplacian on the grid points using auto-differentiation. They were trained in the inverse problem by setting $u = 0$ as the boundary condition and adding a physics loss along the grid points of $f(x, y)$. DeepONets were tested identically but through the train set instead. They take either $u(x, y)$ or $f(x, y)$ as inputs to the branch net depending on the problem, and coordinate (x, y) as the input to the trunk net.

5.1. Sampling f conditioned on u

Our parameters are $\eta = 8e{-}04$ and $\eta_b = 9e{-}04$ to ensure slow and stable incremental convergence. We used $\sigma_{n,m} = 1e - 6$ for all n, m to ensure that the sampling of $\mathbf{f}_T$ can be done for any $\sigma_T > 1$. Our results are posted in table 1.

The DeepONet had an average MAE of 4.224e-01, which outperforms the dry forward process conducted with our DDIM model. PINNs achieve a better MAE of 3.704e-01, even though this is because it directly interacts with the test set.

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 I. 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.

PINNs	PI-DeepONet	Dry Forward Process	DDRM	Finite Difference
3.704e-01	4.224e-01	5.515e-01	3.215e-02	1.163e-02

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

5.2. Sampling u conditioned on f

Our parameters are $\eta = 8e-09$ and $\eta_b = 9e-09$. We used $\sigma_f = 1e-6$ to ensure that the sampling of $\mathbf{u}_T$ can be done for any $\sigma_T > 1$. Our results are posted in table 2.

Unfortunately, despite being trained on the test set itself, PINNs had an average MAE of 0.002156, which is worse than the dry inverse process conducted with our DDIM model. The DeepONet had an average MAE of 3.183e-04, which outperforms the dry inverse process conducted with our DDIM model.

Upon using our modified DDRM to sample $u(x, y)$ conditioned on $f(x, y)$, we got an MAE of 1.175e-06, which is a significant improvement compared to using the DDIM model without restoration and the DeepONet. For qualitative results, we posted three samples in appendix J. This, as a result, demonstrates the practicality of DDRM in solving problems in PDEs. This method, however, does not outperform finite difference approximation that had an MAE of 6.672e-07 on the test set in approximating $u(x, y)$.

PINNs	Dry Inverse Process	PI-DeepONet	DDRM	Finite Difference
2.156e-03	1.123e-03	3.183e-04	1.175e-06	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.1). 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 K.

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.1). 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 Raissi et al. (2019); Blechschmidt and Ernst (2021).

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 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 physics in their training process, 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.

Christopher Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:531–537, 2006.

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.

Oded Ovadia, Eli Turkel, Adar Kahana, and George Em Karniadakis. Ditto: Diffusion-inspired temporal transformer operator. *arXiv preprint arXiv:2307.09072*, 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

$$\Delta u = \lambda u \tag{31}$$

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

$$\Delta u = X''(x)Y(y) + X(x)Y''(y) = \lambda X(x)Y(y) \tag{32}$$

$$\frac{X''}{X} + \frac{Y''}{Y} = \lambda \tag{33}$$

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.

$$\frac{X''}{X} = \lambda_1, X(0) = X(1) = 0 \tag{34}$$

The eigenvalue and eigenfunction pairs are trivially of the form

$$X_n(x) = \sin(n\pi x), \lambda_{1,n} = -(n\pi)^2 \tag{35}$$

By symmetry, we have

$$Y_m(y) = \sin(m\pi y), \lambda_{2,m} = -(m\pi)^2 \tag{36}$$

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

$$u_{n,m}(x, y) = \sin(n\pi x)\sin(m\pi y) \tag{37}$$

$$\lambda_{n,m} = -(n\pi)^2 - (m\pi)^2 \tag{38}$$

■

Appendix B. Proof of theorem 2

Proof In a bounded domain $\Omega \subset \mathbb{R}^2$ with smooth boundary, any function $u \in C^2(\overline{\Omega})$ satisfies

$$\begin{aligned} u(x, y) = &\iint_\Omega \psi((x', y') - (x, y))\Delta u(x', y')dx'dy' \\ &+ \int_{\partial\Omega} u(x', y')\frac{\partial\psi}{\partial n}((x', y') - (x, y))dS_{x',y'} \\ &- \int_{\partial\Omega} \psi((x', y') - (x, y))\frac{\partial u}{\partial n}(x', y')dS_{x',y'}, \end{aligned} \tag{39}$$

where $\psi(x, y) = \frac{\ln(||(x,y)||}{2\pi}$ is the Green's function in two dimensions. We commonly refer to the first integral as the Newtonian potential, the second integral as the double-layer potential, and the third integral as the single-layer potential (we will use these terms in the proof of theorem 3

in appendix C). This known result was retrieved from Gilbarg and Trudinger (1983). Given that $\Delta u = f + z$ with $z \sim \mathcal{N}(0, \sigma_f^2)$, the first integrand can be written as

$$\begin{aligned} &\iint_\Omega \psi((x', y') - (x, y))\Delta u(x', y')dx'dy' \\ &= \iint_\Omega \psi((x', y') - (x, y))f dx'dy' \\ &\quad + \iint_\Omega \psi((x', y') - (x, y))z dx'dy'. \end{aligned} \tag{40}$$

Consequently, it follows that: $E[u(x, y)] = u_0$ where u_0 is the deterministic solution to $\Delta u_0 = f$. Furthermore, we can compute $Var[u(x, y)] = \sigma_f^2 K(x, y)$ where

$$K(x, y) = \iint_\Omega (\psi((x', y') - (x, y)))^2 dx'dy'. \tag{41}$$

■

Appendix C. Proof of theorem 3

Proof This proof follows from the proof of theorem 2 provided in appendix B. Consider the stochastic component of equation 40.

$$u_z(x, y) = \iint_\Omega \psi((x', y') - (x, y))z dx'dy', \tag{42}$$

where $z \sim N(0, \sigma_f^2)$. Let us compute the discrete sine transformation (DST) of $u_z(x, y)$.

$$\overline{\mathbf{u}_z}^{(n,m)} = \iint_\Omega \iint_\Omega \psi((x', y') - (x, y))z dx'dy' \sin(n\pi x) \sin(m\pi y)dxdy \tag{43}$$

$$= \iint_\Omega \left[\iint_\Omega \psi((x', y') - (x, y)) \sin(n\pi x) \sin(m\pi y)dxdy \right] z dx'dy' \tag{44}$$

Notice that the term in the square brackets is the DST of $\psi((x', y') - (x, y))$. It is also identical to the Newtonian potential from equation 39. This is identical to computing an analytical solution for the following PDE:

$$\Delta u = \sin(nx) \sin(mx), \; u = 0 \text{ in } \partial\Omega, \tag{45}$$

where the known solution is

$$u = -\frac{1}{\pi^2(n^2 + m^2)} \sin(nx) \sin(mx). \tag{46}$$

Plugging this into equation 39 gives:

$$\begin{aligned} -\frac{1}{\pi^2(n^2 + m^2)} \sin(nx) \sin(mx) = &\iint_\Omega \psi((x', y') - (x, y)) \sin(n\pi x') \sin(m\pi y')dx'dy' \\ &+ \int_{\partial\Omega} u(x', y') \frac{\partial \psi}{\partial n}((x', y') - (x, y))dS_{x',y'} \\ &- \int_{\partial\Omega} \psi((x', y') - (x, y)) \frac{\partial u}{\partial n}(x', y')dS_{x',y'}. \end{aligned} \tag{47}$$

Due to the boundary conditions, the double-layer potential term is 0. We can compute the gradient vector of u:

$$\nabla u(x,y) = \begin{pmatrix} n\pi\cos(n\pi x)\sin(m\pi y) \\ m\pi\sin(n\pi x)\cos(m\pi y) \end{pmatrix}. \tag{48}$$

The single-layer potential is expensive to compute, but we can bound it.

$$|\int_{\partial\Omega} \psi((x',y')-(x,y))\frac{\partial u}{\partial n}(x',y')dS_{x',y'}| \tag{49}$$

$$\leq \sup_{(x',y')} |\psi((x',y')-(x,y))| \|\nabla u(x,y)\| \int_{\partial\Omega} dS_{x',y'} \tag{50}$$

Since $\Omega = [0,1]^2$, $\int_{\partial\Omega} dS_{x',y'}$ is the sum of its edges, which is 4. The maximum of ∇u along the boundary is $\pi\max(n,m)$.

$$\sup_{(x',y')} |\psi((x',y')-(x,y))| = |\psi((1,1)-(0,0))| = \frac{\ln\sqrt{2}}{2\pi} = \frac{\ln 2}{4\pi} \tag{51}$$

Putting it all together, we get the following upper-bound

$$|\int_{\partial\Omega} \psi((x',y')-(x,y))\frac{\partial u}{\partial n}(x',y')dS_{x',y'}| \tag{52}$$

$$\leq \frac{\ln 2}{4\pi}\cdot\pi\max(n,m)\cdot 4 \tag{53}$$

$$= \ln 2\max(n,m), \tag{54}$$

which gives us the following upper bound for the variance

$$Var[\overline{\mathbf{u}_z}^{(n,m)}|f_0] = Var\left[\iint_\Omega \left[\iint_\Omega \psi((x',y')-(x,y))\sin(n\pi x)\sin(m\pi y)dxdy\right] zdx'dy'\right] \tag{55}$$

$$\leq Var\left[\iint_\Omega \left[\frac{1}{\pi^2(n^2+m^2)}|\sin(nx')\sin(mx')| + \ln 2\max(n,m)\right] zdx'dy'\right] \tag{56}$$

$$\leq Var\left[\iint_\Omega \left[\frac{1}{\pi^2(n^2+m^2)} + \ln 2\max(n,m)\right] zdx'dy'\right] \tag{57}$$

$$= (\frac{1}{\pi^2(n^2+m^2)} + \ln 2\max(n,m))^2\sigma_f^2 \tag{58}$$

■

Appendix D. Proof of proposition 4

Proof This proof uses properties of Gaussian marginals. We refer the reader to Bishop (2006).

Distribution of $\overline{\mathbf{f}}_T^{(n,m)}$

We have equation 18.

$$q^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \mathbf{f}_0, \mathbf{u}) = \mathcal{N}(\overline{\mathbf{u}}^{(n,m)}, \sigma_T^2 - \sigma_{n,m}^2\lambda_{n,m}^2). \tag{59}$$

We can assume that $q^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \mathbf{f}_0, \mathbf{u}) = q^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \overline{\mathbf{f}}_0^{(n,m)}, \overline{\mathbf{u}}^{(n,m)})$, and we know from section 2.3.1 that

$$p(\overline{\mathbf{f}}_0^{(n,m)} \mid \overline{\mathbf{u}}^{(n,m)}) = \mathcal{N}(\overline{\mathbf{f}}_0^{(n,m)}, \sigma_{n,m}^2\lambda_{n,m}^2). \tag{60}$$

Using the property of Gaussian marginals, we can derive the following result

$$q^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \mathbf{f}_0) = \mathcal{N}(\overline{\mathbf{f}}_0^{(n,m)}, \sigma_{n,m}^2\lambda_{n,m}^2 + \sigma_T^2 - \sigma_{n,m}^2\lambda_{n,m}^2) = \mathcal{N}(\overline{\mathbf{f}}_0^{(n,m)}, \sigma_T^2). \tag{61}$$

Distribution of $\overline{\mathbf{f}}_t^{(n,m)}$

We have equation 19. Consider the condition where $\sigma_t < \sigma_{n,m}\lambda_{n,m}$.

$$q^{(t)}\left(\overline{\mathbf{f}}_t^{(n,m)} \mid \mathbf{f}_0, \mathbf{f}_{t+1}, \mathbf{u}\right) = \mathcal{N}\left(\overline{\mathbf{f}}_0^{(n,m)} + \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{u}}^{(n,m)} - \overline{\mathbf{f}}_0^{(n,m)}}{\sigma_{n,m}\lambda_{n,m}}, \eta^2\sigma_t^2\right). \tag{62}$$

We can similarly assume that $q^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \mathbf{f}_0, \mathbf{f}_{t+1}, \mathbf{u}) = q^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \overline{\mathbf{f}}_0^{(n,m)}, \overline{\mathbf{f}}_{t+1}^{(n,m)}, \overline{\mathbf{u}}^{(n,m)})$. And we can also state that $q^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \overline{\mathbf{f}}_0^{(n,m)}, \overline{\mathbf{f}}_{t+1}^{(n,m)}, \overline{\mathbf{u}}^{(n,m)}) = q^{(T)}(\overline{\mathbf{f}}_T^{(n,m)} \mid \overline{\mathbf{f}}_0^{(n,m)}, \overline{\mathbf{u}}^{(n,m)})$ since there is no dependence on $\overline{\mathbf{f}}_{t+1}^{(n,m)}$ in the distribution. We use the property that $\frac{\overline{\mathbf{u}}^{(n,m)} - \overline{\mathbf{f}}_0^{(n,m)}}{\sigma_{n,m}\lambda_{n,m}}$ is a standard Gaussian. Using the property of Gaussian marginals, we can derive the following result

$$\begin{aligned} q^{(t)}\left(\overline{\mathbf{f}}_t^{(n,m)} \mid \mathbf{f}_0\right) &= \mathcal{N}\Big(\overline{\mathbf{f}}_0^{(n,m)} + \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{u}}^{(n,m)} - \overline{\mathbf{f}}_0^{(n,m)}}{\sigma_{n,m}\lambda_{n,m}} - \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{u}}^{(n,m)} - \overline{\mathbf{f}}_0^{(n,m)}}{\sigma_{n,m}\lambda_{n,m}}, \\ &\qquad \eta^2\sigma_t^2 + (1-\eta^2)\sigma_t^2\Big) \qquad (63) \\ &= \mathcal{N}(\overline{\mathbf{f}}_0^{(n,m)}, \sigma_t^2). \qquad (64) \end{aligned}$$

Next, consider the condition where $\sigma_t \geq \sigma_{n,m}\lambda_{n,m}$.

$$q^{(t)}\left(\overline{\mathbf{f}}_t^{(n,m)} \mid \mathbf{f}_0, \mathbf{f}_{t+1}, \mathbf{u}\right) = \mathcal{N}\Big((1-\eta_b)\,\overline{\mathbf{f}}_0^{(n,m)} + \eta_b\overline{\mathbf{u}}^{(n,m)}, \sigma_t^2 - \sigma_{n,m}^2\lambda_{m,n}^2\eta_b^2\Big) \tag{65}$$

Using the property that $\eta_b(\overline{\mathbf{f}}_0^{(n,m)} + \overline{\mathbf{u}}^{(n,m)}) = \mathcal{N}(0, \sigma_{n,m}^2\lambda_{m,n}^2\eta_b^2)$, we can derive using Gaussian marginals

$$\begin{aligned} q^{(t)}\left(\overline{\mathbf{f}}_t^{(n,m)} \mid \mathbf{f}_0\right) &= \mathcal{N}\Big((1-\eta_b)\,\overline{\mathbf{f}}_0^{(n,m)} + \eta_b\overline{\mathbf{u}}^{(n,m)} + \eta_b(\overline{\mathbf{f}}_0^{(n,m)} + \overline{\mathbf{u}}^{(n,m)}), \\ &\qquad \sigma_t^2 - \sigma_{n,m}^2\lambda_{m,n}^2\eta_b^2 + \sigma_{n,m}^2\lambda_{m,n}^2\eta_b^2\Big) \qquad (66) \\ &= \mathcal{N}(\overline{\mathbf{f}}_0^{(n,m)}, \sigma_t^2). \qquad (67) \end{aligned}$$

■

Appendix E. Proof of proposition 5

Proof We notify the reader that the proof of this proposition is nearly a repetition of the proof of proposition 4 presented in Appendix D, with the only difference being the variance terms in the forward and inverse variational distributions.

This proof uses properties of Gaussian marginals. We refer the reader to Bishop (2006).

Distribution of $\overline{\mathbf{u}}_T^{(n,m)}$

We have equation 26.

$$q^{(T)}(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_0, \mathbf{f}) = \mathcal{N}(\overline{\mathbf{f}}^{(n,m)}, \sigma_T^2 - \sigma_f^2 \overline{\mathbf{K}}^{(n,m)}/\lambda_{n,m}^2), \tag{68}$$

We can assume that $q^{(T)}(\overline{\mathbf{u}}_T^{(n,m)} \mid \mathbf{u}_0, \mathbf{f}) = q^{(T)}(\overline{\mathbf{u}}_T^{(n,m)} \mid \overline{\mathbf{u}}_0^{(n,m)}, \overline{\mathbf{f}}^{(n,m)})$, and we know from section 2.3.2 that

$$p(\overline{\mathbf{u}}_0^{(n,m)} \mid \overline{\mathbf{f}}^{(n,m)}) = \mathcal{N}(\overline{\mathbf{u}}_0^{(n,m)}, \sigma_f^2 \overline{\mathbf{K}}^{(n,m)}/\lambda_{n,m}^2). \tag{69}$$

Using the property of Gaussian marginals, we can derive the following result

$$q^{(T)}(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_0) = \mathcal{N}\left(\overline{\mathbf{u}}_0^{(n,m)}, \sigma_T^2 - \sigma_f^2 \overline{\mathbf{K}}^{(n,m)}/\lambda_{n,m}^2 + \sigma_f^2 \overline{\mathbf{K}}^{(n,m)}/\lambda_{n,m}^2\right) = \mathcal{N}\left(\overline{\mathbf{u}}_0^{(n,m)}, \sigma_T^2\right). \tag{70}$$

Distribution of $\overline{\mathbf{u}}_t^{(n,m)}$

We have equation 27. Consider the condition where $\sigma_t < \frac{\sigma_f \sqrt{\overline{\mathbf{K}}^{(n,m)}}}{\lambda_{n,m}}$

$$q^{(t)}\left(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_0, \mathbf{u}_{t+1}, \mathbf{f}\right) = \mathcal{N}\left(\overline{\mathbf{u}}_0^{(n,m)} + \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{f}}^{(n,m)} - \overline{\mathbf{u}}_0^{(n,m)}}{\sigma_f \sqrt{\overline{\mathbf{K}}^{(n,m)}}/\lambda_{n,m}}, \eta^2 \sigma_t^2\right) \tag{71}$$

We can similarly assume that $q^{(T)}(\overline{\mathbf{u}}_T^{(n,m)} \mid \mathbf{u}_0, \mathbf{u}_{t+1}, \mathbf{f}) = q^{(T)}(\overline{\mathbf{u}}_T^{(n,m)} \mid \overline{\mathbf{u}}_0^{(n,m)}, \overline{\mathbf{u}}_{t+1}^{(n,m)}, \overline{\mathbf{f}}^{(n,m)})$. And we can also state that $q^{(T)}(\overline{\mathbf{u}}_T^{(n,m)} \mid \overline{\mathbf{u}}_0^{(n,m)}, \overline{\mathbf{u}}_{t+1}^{(n,m)}, \overline{\mathbf{f}}^{(n,m)}) = q^{(T)}(\overline{\mathbf{u}}_T^{(n,m)} \mid \overline{\mathbf{u}}_0^{(n,m)}, \overline{\mathbf{f}}^{(n,m)})$ since there is no dependence on $\overline{\mathbf{u}}_{t+1}^{(n,m)}$ in the distribution. We use the property that $(\overline{\mathbf{f}}^{(n,m)} - \overline{\mathbf{u}}_0^{(n,m)}) / \left(\sigma_f \sqrt{\overline{\mathbf{K}}^{(n,m)}}/\lambda_{n,m}\right)$ is a standard Gaussian. Using the property of Gaussian marginals, we can derive the following result

$$\begin{aligned} q^{(t)}\left(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_0\right) = \mathcal{N}\Bigg(&\overline{\mathbf{u}}_0^{(n,m)} + \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{f}}^{(n,m)} - \overline{\mathbf{u}}_0^{(n,m)}}{\sigma_f \sqrt{\overline{\mathbf{K}}^{(n,m)}}/\lambda_{n,m}} \\ &- \sqrt{1-\eta^2}\sigma_t \frac{\overline{\mathbf{f}}^{(n,m)} - \overline{\mathbf{u}}_0^{(n,m)}}{\sigma_f \sqrt{\overline{\mathbf{K}}^{(n,m)}}/\lambda_{n,m}}, \\ &\eta^2\sigma_t^2 + (1-\eta^2)\sigma_t^2\Bigg) \end{aligned} \tag{72}$$

$$= \mathcal{N}\left(\overline{\mathbf{u}}_0^{(n,m)}, \sigma_t^2\right) \tag{73}$$

Nwext, consider the condition where $\sigma_t \geq \frac{\sigma_f \sqrt{\overline{\mathbf{K}}^{(n,m)}}}{\lambda_{n,m}}$

$$q^{(t)}\left(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_0, \mathbf{u}_{t+1}, \mathbf{f}\right) = \mathcal{N}\Big((1-\eta_b)\,\overline{\mathbf{u}}_0^{(n,m)} + \eta_b \overline{\mathbf{f}}^{(n,m)}, \sigma_t^2 - \frac{\sigma_f^2 \overline{\mathbf{K}}^{(n,m)}}{\lambda_{n,m}^2}\eta_b^2\Big) \tag{74}$$

Using the property that $\eta_b(\overline{\mathbf{u}}_0^{(n,m)} + \overline{\mathbf{f}}^{(n,m)}) = \mathcal{N}(0, \frac{\sigma_f^2 \overline{\mathbf{K}}^{(n,m)}}{\lambda_{n,m}^2}\eta_b^2)$, we can derive using Gaussian marginals

$$\begin{aligned} q^{(t)}\left(\overline{\mathbf{u}}_t^{(n,m)} \mid \mathbf{u}_0\right) &= \mathcal{N}\Big((1-\eta_b)\,\overline{\mathbf{u}}_0^{(n,m)} + \eta_b \overline{\mathbf{f}}^{(n,m)} + \eta_b(\overline{\mathbf{u}}_0^{(n,m)} + \overline{\mathbf{f}}^{(n,m)}), \\ &\qquad \sigma_t^2 - \frac{\sigma_f^2 \overline{\mathbf{K}}^{(n,m)}}{\lambda_{n,m}^2}\eta_b^2 + \frac{\sigma_f^2 \overline{\mathbf{K}}^{(n,m)}}{\lambda_{n,m}^2}\eta_b^2\Big) \qquad (75) \\ &= \mathcal{N}\Big(\overline{\mathbf{u}}_0^{(n,m)}, \sigma_t^2\Big) \qquad (76) \end{aligned}$$

■

Appendix F. Dataset solutions

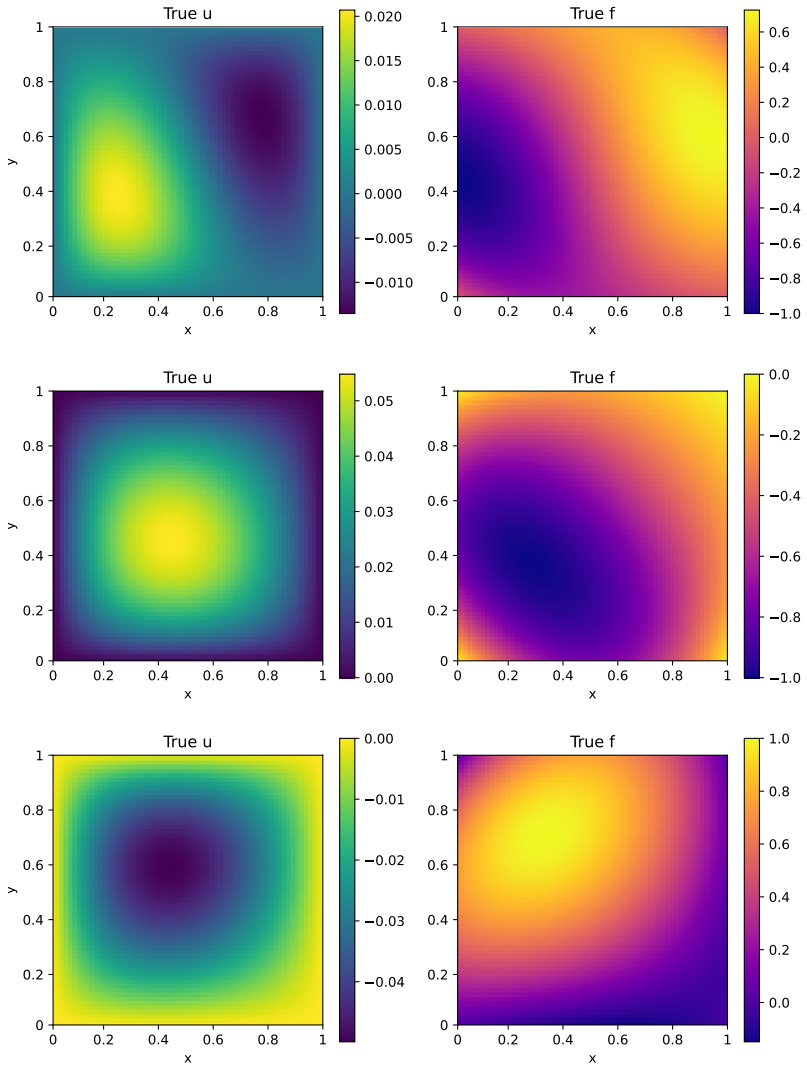
We generated 38,250 samples for the diffusion model for PDE data generation of the following types: neural network pairs and analytical pairs. Here we show some examples of these different types of data.

F.1. Neural network pairs

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

$$u(x, y) = g_{NN}(x, y)x(1-x)y(1-y), \tag{77}$$

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 3.



(*a*)

Figure 3: Examples of neural network pairs.

F.2. Analytical pairs

To train a diffusion model to generate data for the 2D Poisson equation with homogeneous Diriclet boundary condition (Eq. 1) 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. 4.

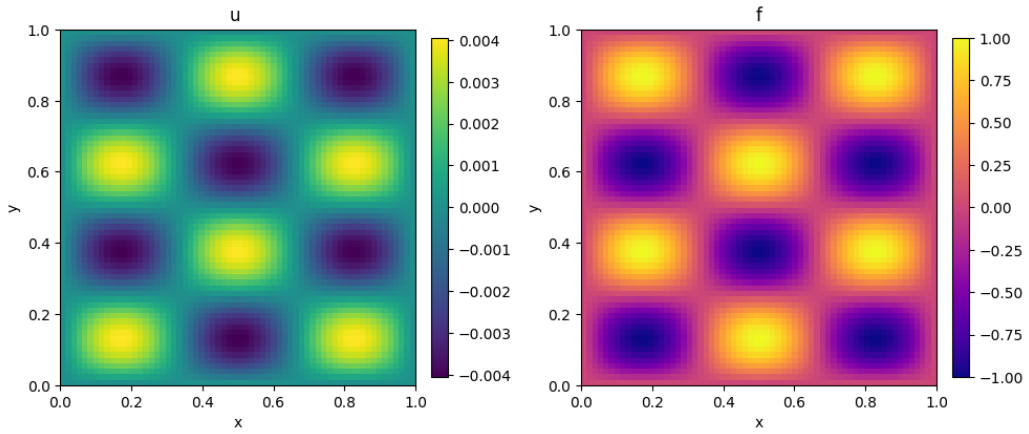


Figure 4: Example type 1 analytical solution with $n = 3$ and $k = 4$.

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. 5.

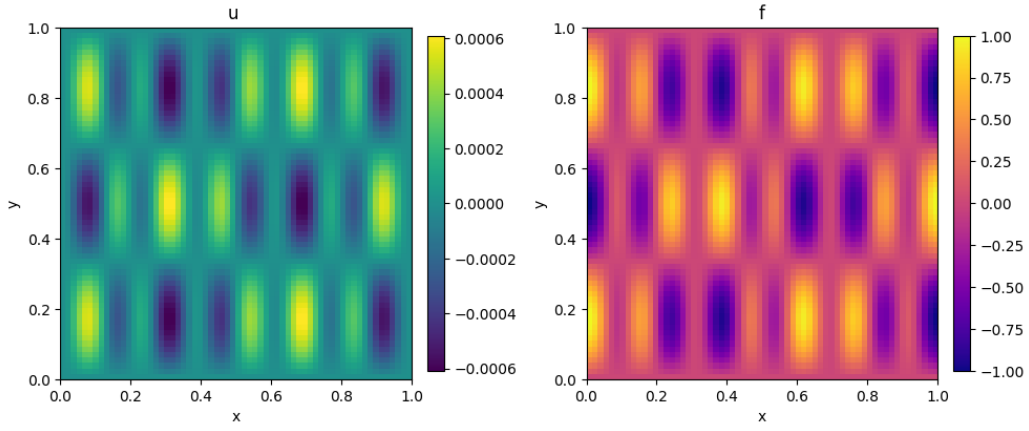


Figure 5: Example type 2 analytical solution with $n = 5$, $k = 3$, and $j = 8$.

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. 6.

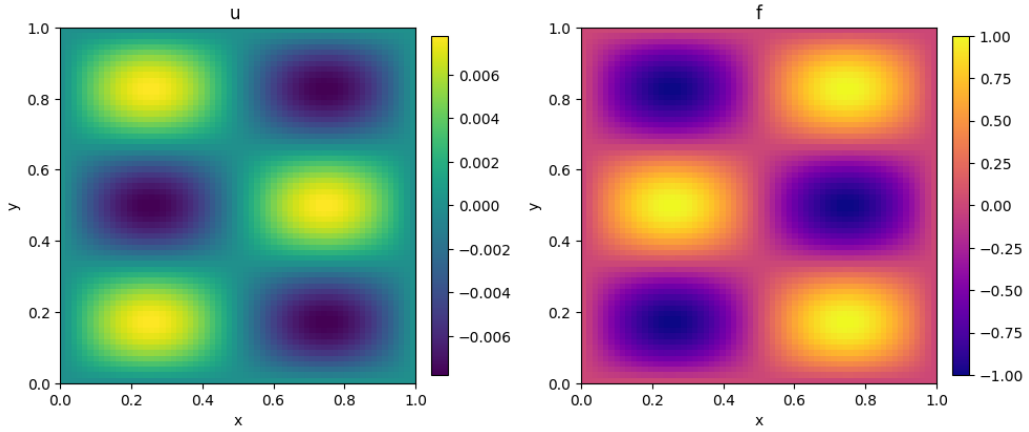


Figure 6: Example type 3 analytical solution with $n = 1$ and $k = 3$.

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. 7.

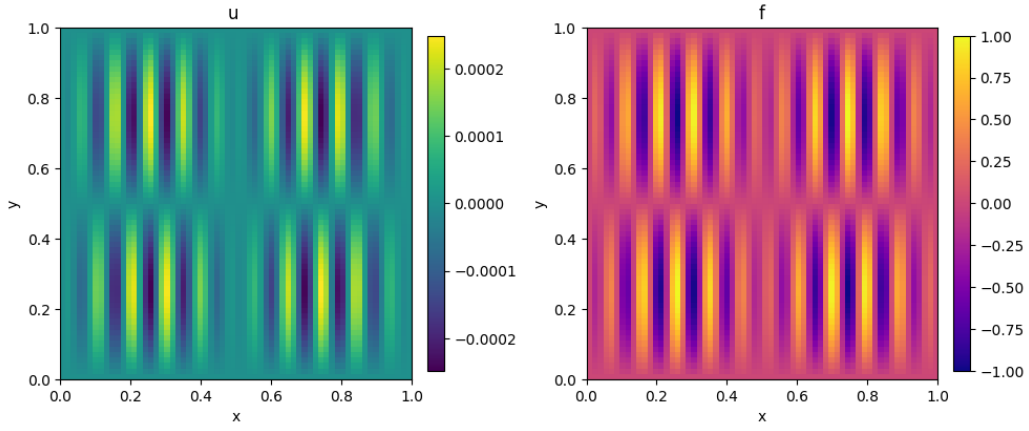


Figure 7: Example type 4 analytical solution with $n = 2$ and $k = 2$.

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. 8.

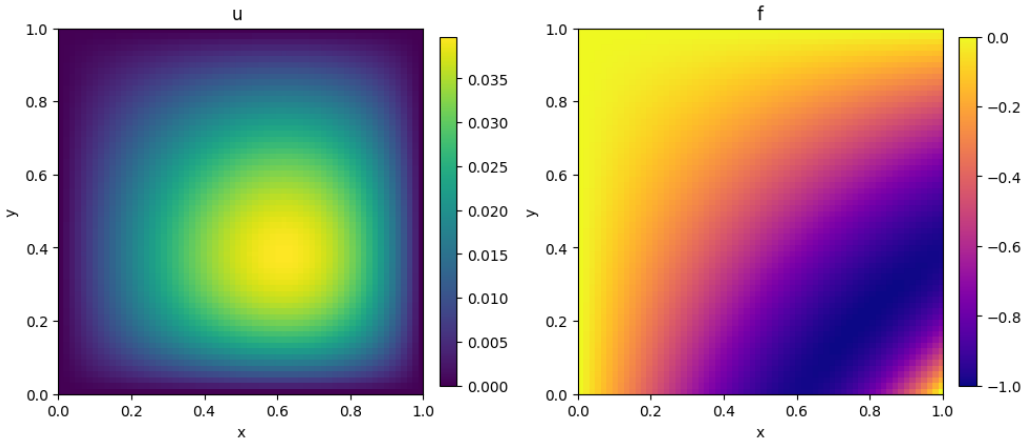


Figure 8: Example type 5 analytical solution with $n = 2$.

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. 9.

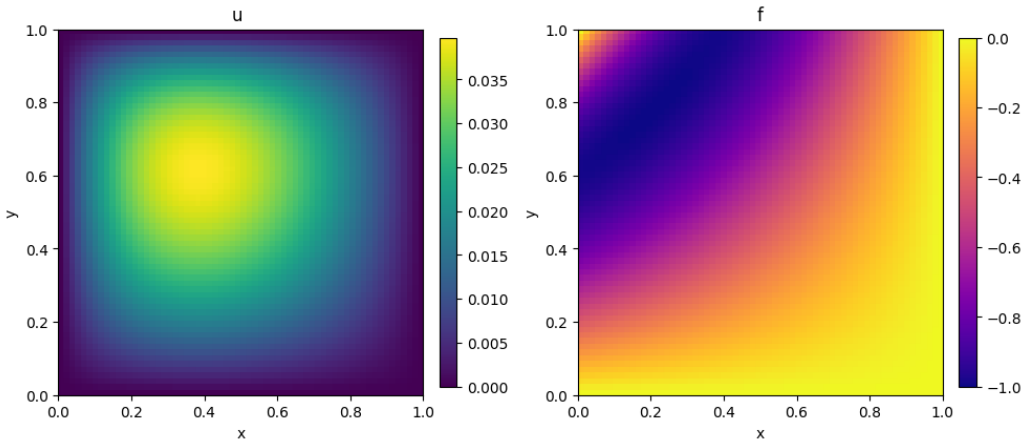


Figure 9: Example type 6 analytical solution with $n = 8$.

Appendix G. Qualitative Results - Dry Forward 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 F.1, three of which are plotted in figure 10.

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.

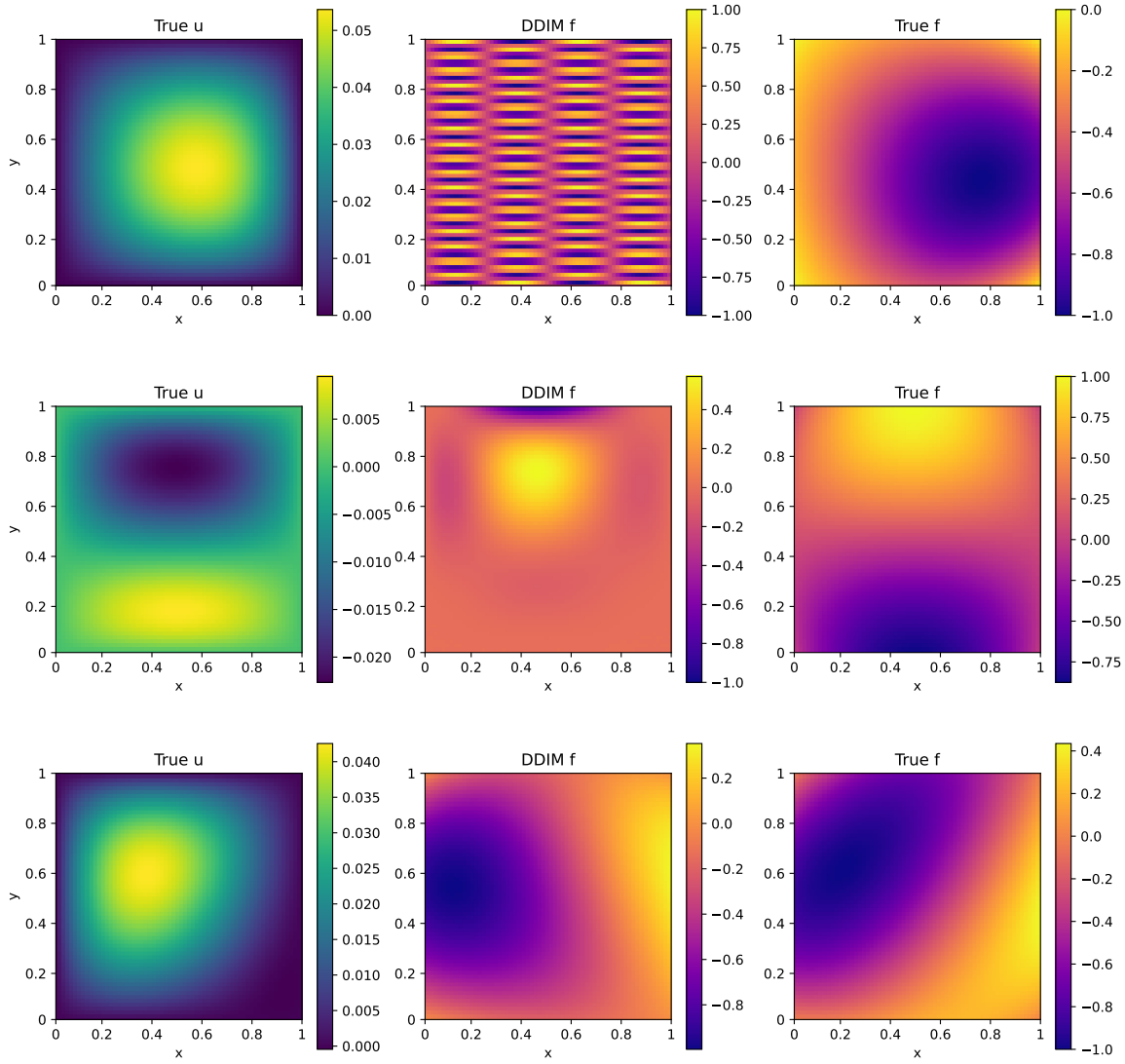


Figure 10: Plots of dry generated solutions of the inverse process $f(x, y)$ (middle), $u(x, y)$ (left), and the true $f(x, y)$ (right).

Appendix H. Qualitative results - Dry Inverse Process

We trained a DDIM on the dataset explained in section 4.1 and appendix F. 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 F.1, three of which are plotted in figure 11.

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.

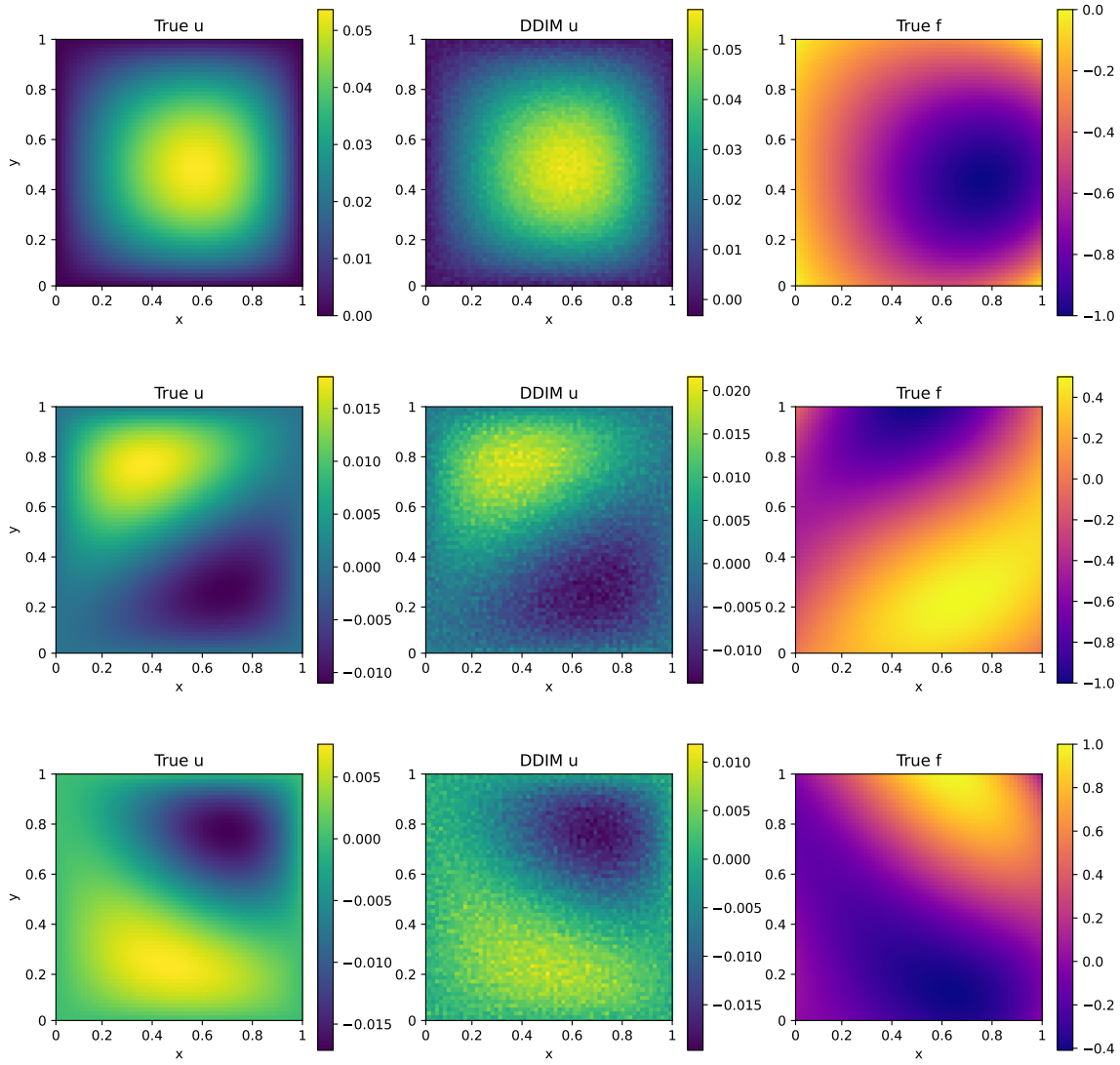


Figure 11: Plots of dry generated solutions of the forward process $u(x, y)$ (middle), true solution $f(x, y)$ (left), and $f(x, y)$ (right).

Appendix I. Qualitative Results - DDRM Forward 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 F.1, three of which are plotted in figure 12.

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.

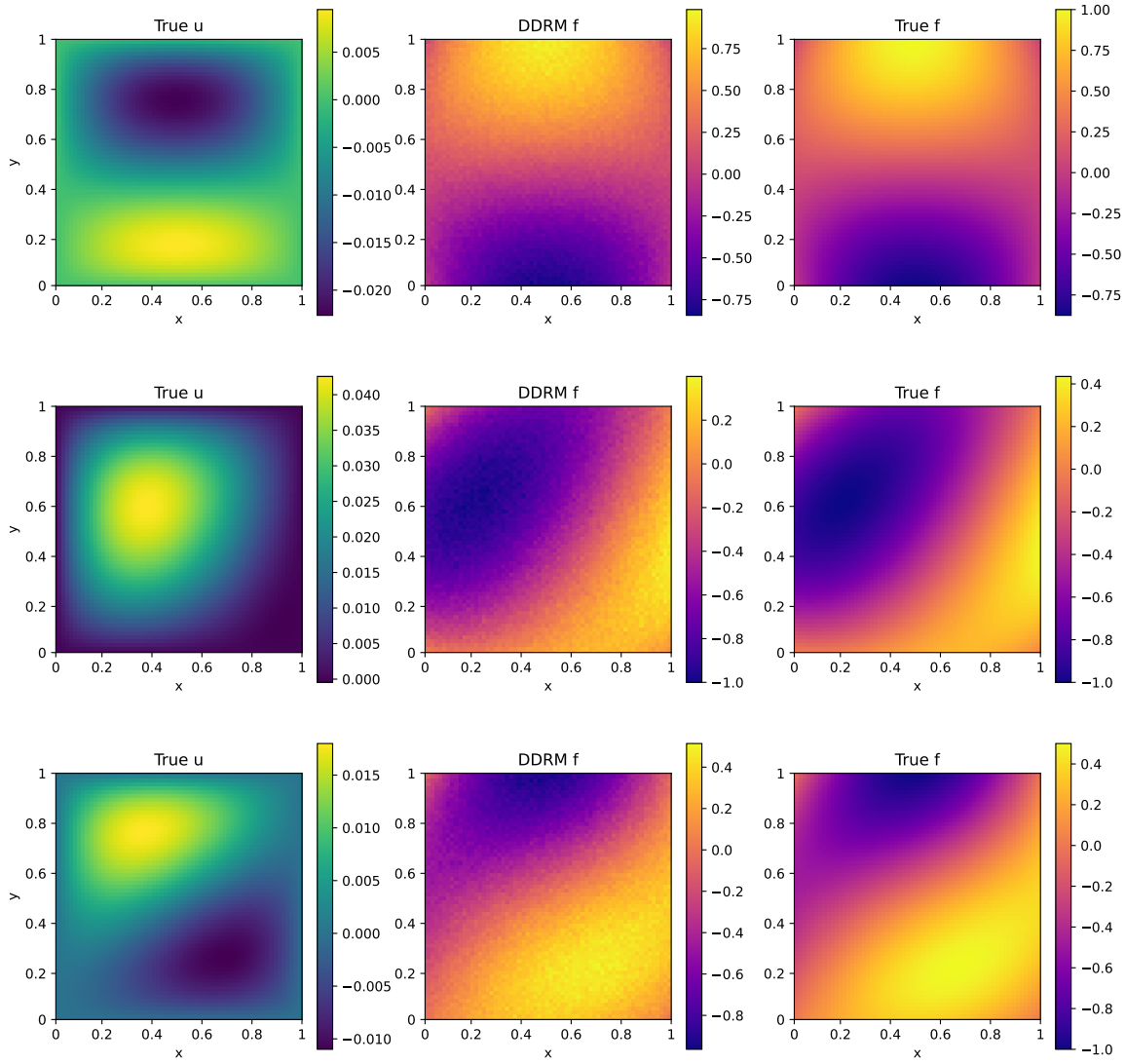


Figure 12: Plots of DDRM generated solutions of the inverse process $f(x, y)$ (middle), $u(x, y)$ (left), and the true $f(x, y)$ (right).

Appendix J. Qualitative Results - DDRM Inverse 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 F.1, three of which are plotted in figure 13.

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 $1.175e-06$, which is just slightly greater than the MAE of $6.672e-07$ upon using the finite difference method.

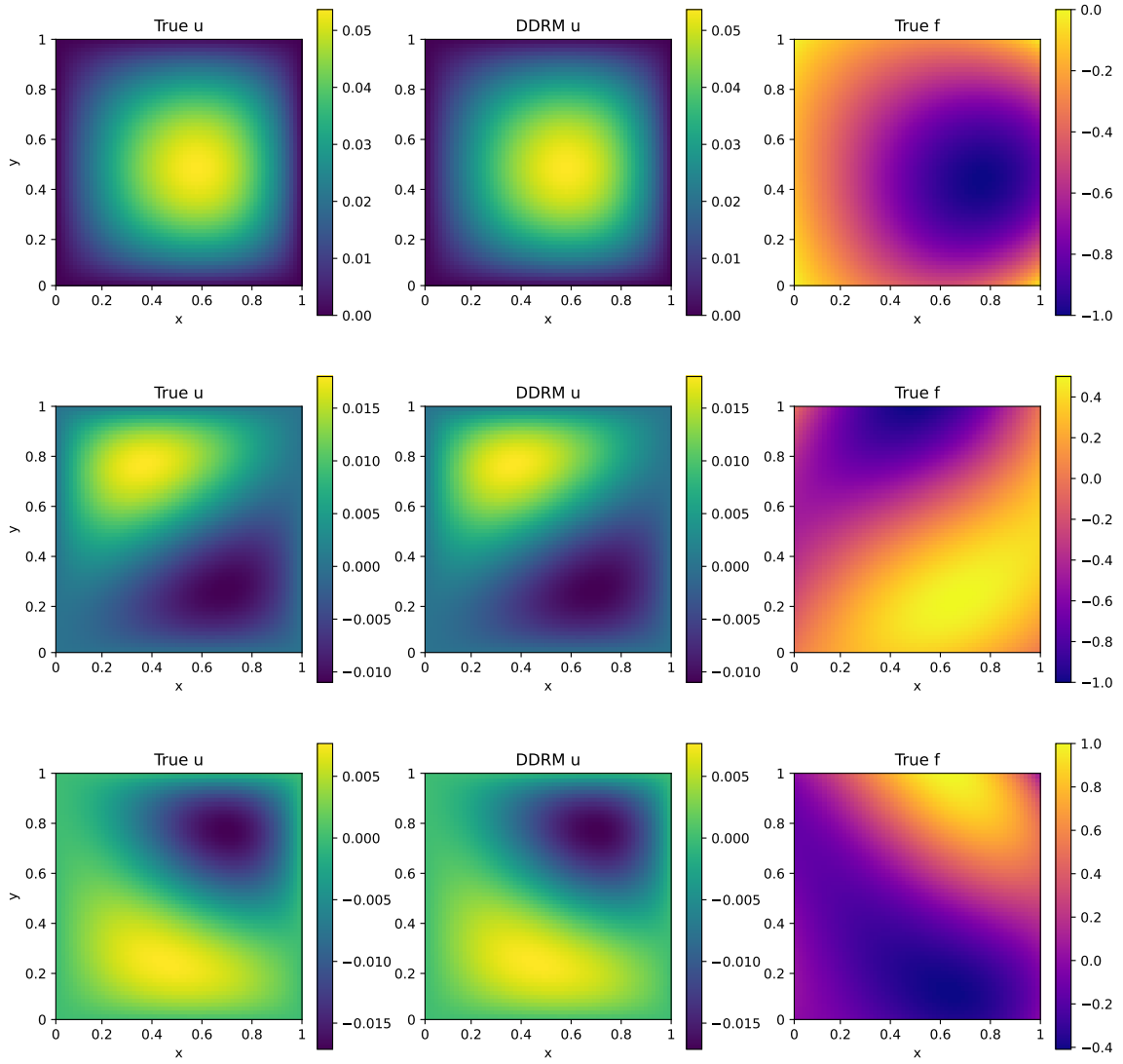


Figure 13: Plots of DDRM generated solutions of the forward process $u(x, y)$ (middle), true $u(x, y)$ (left), and $f(x, y)$ (right).

Appendix K. Unconditionally Generated Data

We plotted three examples of solutions generated by our model unconditionally in figure 14. The MAE between the generated solution $u(x, y)$ and the finite difference solution is 4.373e-04, thus showing that the generated solutions are a good approximation to the true solution.

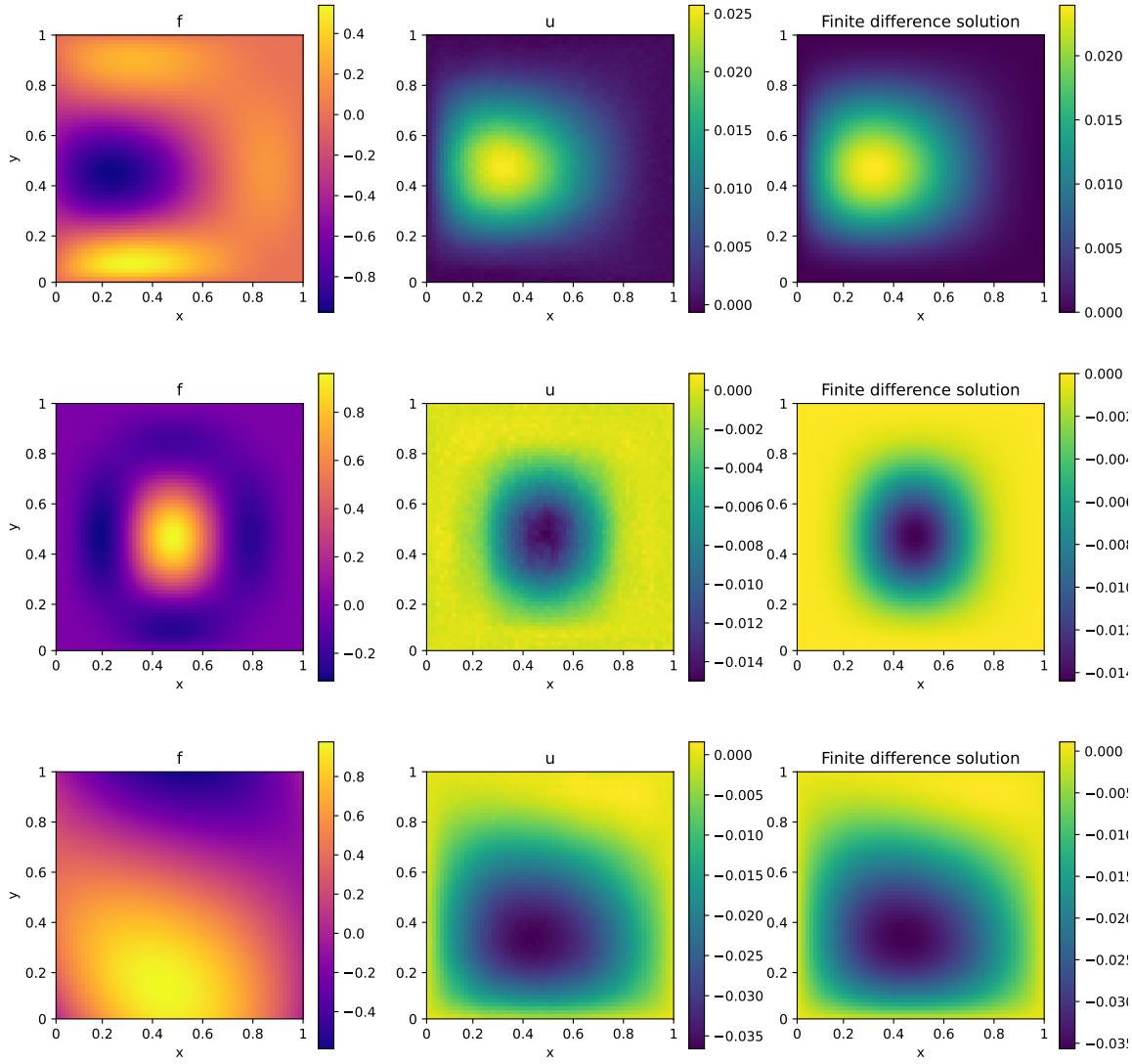


Figure 14: Plots of unconditionally generated pairs of $f(x, y)$ (left), $u(x, y)$ (middle), and the finite difference solution of $u(x, y)$ (right).