DiffusionPDE: Generative PDE-Solving Under Partial Observation

Jiahe Huang [1] **Guandao Yang** [2] **Zichen Wang** [1] **Jeong Joon Park** [1]

Abstract

We introduce a general framework for solving partial differential equations (PDEs) using generative diffusion models. In particular, we focus on the scenarios where we do not have the full knowledge of the scene necessary to apply classical solvers. Most existing forward or inverse PDE approaches perform poorly when the observations on the data or the underlying coefficients are incomplete, which is a common assumption for real-world measurements. In this work, we propose *DiffusionPDE* that can simultaneously fill in the missing information and solve a PDE by modeling the joint distribution of the solution and coefficient spaces. We show that the learned generative priors lead to a versatile framework for accurately solving a wide range of PDEs under partial observation, significantly outperforming the state-of-the-art methods for both forward and inverse directions. See our project page for results: jhhuangchloe.github.io/Diffusion-PDE/.

1. Introduction

Partial differential equations (PDEs) are a cornerstone of modern science, underpinning many contemporary physical theories that explain natural phenomena. The ability to solve PDEs grants us the power to predict future states of a system (forward process) and estimate underlying physical properties from state measurements (inverse process).

To date, numerous methods (Aliabadi, 2020; Ŝolín, 2005) have been proposed to numerically solve PDEs for both the forward and inverse directions. However, the classical methods can be prohibitively slow, prompting the development of data-driven, learning-based solvers that are significantly faster and capable of handling a family of PDEs. These learning-based approaches (Li et al., 2020; 2021; Lu et al., 2021; Raissi et al., 2019) typically learn a *deterministic* mapping between input coefficients and their solutions using deep neural networks.

Despite the progress, existing learning-based approaches, much like classical solvers, rely on complete observations of the coefficients to map solutions. However, complete information on the underlying physical properties or the state of a system is rarely accessible; in reality, most measurements are sparse in space and time. Both classical solvers and the state-of-the-art data-driven models often overlook these scenarios and consequently fail when confronted with partial observations. This limitation confines their use primarily to synthetic simulations, where full scene configurations are available by design, making their application to real-world cases challenging.

We present a comprehensive framework, DiffusionPDE, for solving PDEs in both forward and inverse directions under conditions of highly partial observations—typically just 1˜3% of the total information. This task is particularly challenging due to the numerous possible ways to complete missing data and find subsequent solutions. Our approach uses a generative model to formulate the joint distribution of the coefficient and solution spaces, effectively managing the uncertainty and simultaneously reconstructing both spaces. During inference, we sample random noise and iteratively denoise it following standard diffusion models (Ho et al., 2020). However, we uniquely guide this denoising process with sparse observations and relevant PDE constraints, generating plausible outputs that adhere to the imposed constraints. Notably, DiffusionPDE can handle observations with arbitrary density and patterns with a single pre-trained generative network.

We conduct extensive experiments to show the versatility of DiffusionPDE as a general PDE-solving framework. We evaluate it on a diverse set of static and temporal PDEs, including Darcy Flow, Poisson, Helmholtz, Burger's, and Navier-Stokes equations. DiffusionPDE significantly outperforms existing state-of-the-art learning-based methods for solving PDEs (Li et al., 2020; 2021; Lu et al., 2021; Raissi et al., 2019; Zhao et al., 2022) in both forward and inverse directions with sparse measurements, while achieving comparable results with full observations. Highlighting the effectiveness of our model, DiffusionPDE accurately reconstructs the complete state of Burgers' equation using

[1]University of Michigan [2]Stanford University. Correspondence to: Jeong Joon Park <jjparkcv@umich.edu>.

5th AI for Science Workshop at the 41^{st} *International Conference on Machine Learning (ICML)*, 2024.

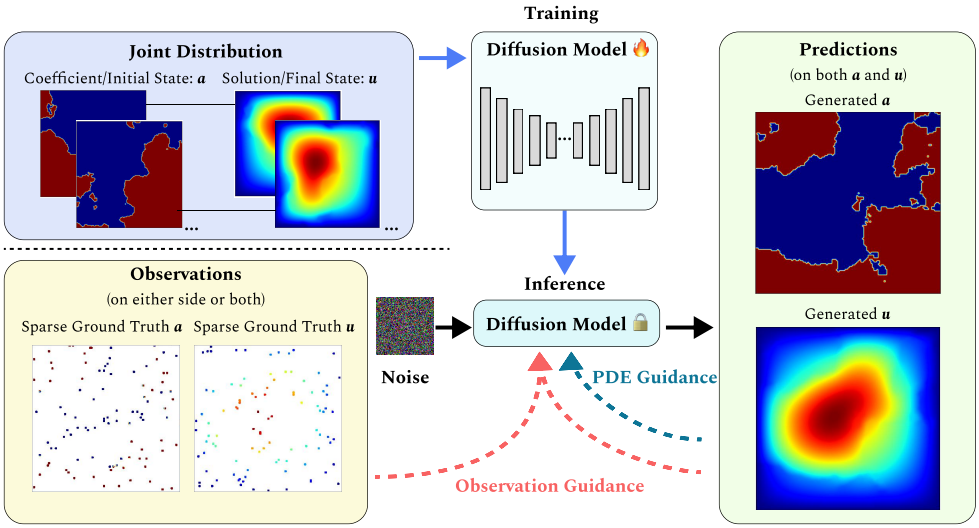


Figure 1. We propose DiffusionPDE, a generative PDE solver under partial observations. Given a family of PDE with coefficient (initial state) a and solution (final state) u, we train the diffusion model on the joint distribution of a and u. During inference, we gradually denoise a Gaussian noise, guided by sparse observation and known PDE function, to recover the full prediction of both a and u that align well with the sparse observations and the given equation.

time-series data from just five sensors (Fig. 4), suggesting the potential of generative models to revolutionize physical modeling in real-world applications.

2. Related Works

Our work builds on the extensive literature of three areas: forward PDE solvers, inverse PDE solvers, and diffusion models. Please see relevant surveys for more information (Evans, 2022; Omori & Kotera, 2007; Po et al., 2023; Strauss, 2007; Yang et al., 2023).

Forward PDE Solvers. PDE solvers take the specification of a physics system and predict its state in unseen space and time by solving an equation involving partial derivatives. Since Most PDEs are very challenging to solve analytically, people resolve to numerical techniques, such as Finite Element Method (Quarteroni & Valli, 2008; Ŝolín, 2005) and Boundary Element Method (Aliabadi, 2020; Idelsohn et al., 2003). While these techniques show strong performance and versatility in some problems, they can be computationally expensive or difficult to set up for complex physics systems. Recently, advancements in deep-learning methods have inspired a new set of PDE solvers. Raissi et al. (Raissi et al., 2017; 2019) introduce Physics-Informed Neural Networks (PINNs), which optimize a neural network using PDE constraints as self-supervised losses to output the PDE solutions. PINNs have been extended to solving specific fluid (Cai et al., 2021a; Mao et al., 2020), Reynolds-averaged Navier–Stokes equations (Eivazi et al., 2022), heat equations (Cai et al., 2021b), and dynamic power systems (Misyris et al., 2020). While PINNs can tackle a wide range of complex PDE problems, they are difficult to scale due to the need for network optimization. An alternative approach, neural operators (Li et al., 2020; Lu et al., 2021), directly learn the mapping from PDE parameters (*e.g.*initial and boundary condition) to the solution function. Once trained, this method avoids expensive network optimization and can instantly output the solution result. This idea has been extended to solve PDE in 3D (Li et al., 2024; Serrano et al., 2024) , multiphase flow (Wen et al., 2022), seismic wave (Lehmann et al., 2023; Li et al., 2023a), 3D turbulence (Li et al., 2022; Peng et al., 2023), and spherical dynamics (Bonev et al., 2023). People have also explored using neural networks as part of the PDE solver, such as compressing the physics state (Chen et al., 2022; Li et al., 2023b; Müller et al., 2021; Nam et al., 2024). These solvers usually assume known PDE parameters, and applying them to solve the inverse problem can be challenging.

PDE inverse problem. The inverse problem refers to finding the coefficients of a PDE that can induce certain observations, mapping from the solution of a PDE solver to its input parameters. People have tried to extend traditional numerical methods to this inverse problem (Cho et al., 2017; Fox & Nicholls, 2001; Harrach, 2021; Kumar & Choi, 2023; van Leeuwen & Herrmann, 2015), but these extensions are non-trivial to implement efficiently. There are similar attempts to inverse deep-learning PDE solvers. For example, one can inverse PINNs by optimizing the network parameters such that their outputs satisfy both the observed data and the governing equations. iFNO (Long & Zhe, 2024) and NIO (Molinaro et al., 2023) tries to extend FNO (Li et al., 2020). Other methods (de Hoop et al., 2022; Pakravan et al., 2021)

directly learn the operator functions for the inverse problem. PINO (Li et al., 2021) further combines neural operators with physics constraints to improve the performance of both forward and inverse problems. These methods assume full observations are available. To address the inverse problem with partial observations, people have tried to leverage generative priors with Graph neural networks (Iakovlev et al., 2020; Zhao et al., 2022). These works have not demonstrated the ability to solve high-resolution PDEs, possibly limited by the power of generative prior. We want to leverage the state-of-the-art generative model, diffusion models, to develop a better inverse PDE solver.

Diffusion models. Diffusion models have shown great promise in learning the prior with higher resolutions by progressively estimating and removing noise. Models like DDIM (Song et al., 2020), DDPM (Ho et al., 2020), and EDM (Karras et al., 2022) offer expressive generative capabilities but face challenges when sampling with specific constraints. Guided diffusion models (Chung et al., 2022a;b; Dhariwal & Nichol, 2021; Xu et al., 2023) enhance generation processes with constraints such as image inpainting, providing more stable and accurate solutions. Prior works on diffusion models for PDEs highlight the potential of diffusion approaches by generating PDE datasets such as 3D turbulence (Jacobsen et al., 2023; Lienen et al., 2023) and Navier-Stokes equations (Yang & Sommer, 2023) with diffusion models. Diffusion models can also be used to model frequency spectrum and denoise the solution space (Lippe et al., 2024), and conditional diffusion models are applied to solve 2D flows with sparse observation (Shu et al., 2023). However, the application of diffusion models to solve inverse problems under partial observation remains underexplored. In this work, we aim to take the initial steps towards addressing this gap.

3. Methods

3.1. Overview

To solve physics-informed forward and inverse problems under uncertainty, we start by pre-training a diffusion generative model on a family of partial differential equations (PDEs). This model is designed to learn the joint distribution of the PDE coefficients (or the initial state) and its corresponding solutions (or the final state). Our approach involves recovering full data in both spaces using sparse observations from either or both sides. We achieve this through the iterative denoising of random Gaussian noise as in regular diffusion models but with additional guidance from the sparse observations and the PDE function enforced during denoising. The schematic description of our approach is shown in Fig. 1.

3.2. Prelimary: Diffusion Models and Guided Diffusion

Diffusion models involve a predefined forward process that gradually adds Gaussian noise to the data and a learned reverse process that denoises the data to reconstruct the original distribution. Specifically, Song et al. (Song et al., 2021) propose a deterministic diffusion model that learns an N-step denoising process that eventually outputs a denoised data $\boldsymbol{x}_N$ and satisfies the following ordinary differential equations (ODE) at each timestep t_i where $i \in \{0, 1, ..., N-1\}$

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}; \sigma(t)\big)\mathrm{d}t. \tag{1}$$

Here $\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}; \sigma(t)\big)$ is the score function (Hyvärinen & Dayan, 2005) that helps to transform samples from a normal distribution $\mathcal{N}(0, \sigma(t_0)^2\mathbf{I})$ to a target probability distribution $p(\boldsymbol{x}; \sigma(t))$. To estimate the score function, Karras et al. (Karras et al., 2022) propose to learn a denoiser function $D(\boldsymbol{x}; \sigma)$ such that

$$\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}; \sigma(t)\big) = \big(D(\boldsymbol{x}; \sigma(t)) - \boldsymbol{x}\big)/\sigma(t)^2 \tag{2}$$

To enable control over the generated data, guided diffusion methods (Dhariwal & Nichol, 2021) add guidance gradients to the score function during the denoising process. Recently, diffusion posterior sampling (DPS) (Chung et al., 2022a) made notable progress in guided diffusion for tackling various inverse problems. DPS uses corrupted measurements $\boldsymbol{y}$ derived from $\boldsymbol{x}$ to guide the diffusion model in outputting the posterior distribution $p(\boldsymbol{x}|\boldsymbol{y})$. A prime application of DPS is the inpainting problem, which involves recovering a complete image from sparsely observed pixels, which suits well with our task. This approach modifies Eq. 1 to

$$\begin{aligned}\mathrm{d}\boldsymbol{x} = &-\dot{\sigma}(t)\sigma(t)\big(\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}; \sigma(t)\big) \\ &+ \nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{y}|\boldsymbol{x}; \sigma(t)\big)\big)\mathrm{d}t.\end{aligned} \tag{3}$$

DPS (Chung et al., 2022a) showed that under Gaussian noise assumption of the sparse measurement operator $\mathcal{M}(\cdot)$, i.e., $\boldsymbol{y}|\boldsymbol{x} \sim \mathcal{N}(\mathcal{M}(\boldsymbol{x}), \delta^2\mathbf{I})$ with some S.D. δ, the log-likelihood function can be approximated with:

$$\begin{aligned}&\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{y}|\boldsymbol{x}_i; \sigma(t_i)\big) \\ &\approx \nabla_{\boldsymbol{x}_i} \log p\big(\boldsymbol{y}|\hat{\boldsymbol{x}}_N^i; \sigma(t_i)\big) \\ &\approx -\frac{1}{\delta^2}\nabla_{\boldsymbol{x}_i}\|\boldsymbol{y} - \mathcal{M}(\hat{\boldsymbol{x}}_N^i(\boldsymbol{x}_i; \sigma(t_i))\|_2^2,\end{aligned} \tag{4}$$

where $\hat{\boldsymbol{x}}_N^i := D(\boldsymbol{x}_i; \sigma(t_i))$ denotes the estimation of the final denoised data at each denoising step i. Applying the Baye's rule, the gradient direction of the guided diffusion is therefore:

$$\nabla_{\boldsymbol{x}_i} \log p(\boldsymbol{x}_i|\boldsymbol{y}) \approx s(\boldsymbol{x}_i) - \zeta\nabla_{\boldsymbol{x}_i}\|\boldsymbol{y} - \mathcal{M}(\hat{\boldsymbol{x}}_N^i)\|_2^2, \tag{5}$$

where $s(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}\big)$ is the original score function, and $\zeta = 1/\delta^2$.

Algorithm 1 Sparse Observation and PDE Guided Diffusion Sampling Algorithm.

input DeterministicSampler $D_\theta(\boldsymbol{x};\sigma)$, $\sigma(t_{i\in\{0,\dots,N\}})$, TotalPointCount m, ObservedPointCount n, Observation $\boldsymbol{y}$, PDEFunction f, Weights ζ_{obs}, ζ_{pde}
sample $\boldsymbol{x}_0 \sim \mathcal{N}\big(\mathbf{0},\ \sigma(t_0)^2\mathbf{I}\big)$ ▷ Initial sampling noise
for $i \in \{0,\dots,N-1\}$ **do**
  $\hat{\boldsymbol{x}}_N^i \leftarrow D_\theta\left(\boldsymbol{x}_i;\sigma(t_i)\right)$ ▷ Estimated denoised data at t_i
  $\boldsymbol{d}_i \leftarrow \left(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_N^i\right)/\sigma(t_i)$ ▷ Evaluated $\mathrm{d}\boldsymbol{x}/\mathrm{d}\sigma(t)$ at t_i
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + (\sigma(t_{i+1}) - \sigma(t_i))\boldsymbol{d}_i$ ▷ Euler step
  if $\sigma(t_{i+1}) \neq 0$ **then**
    $\hat{\boldsymbol{x}}_N^i \leftarrow D_\theta(\boldsymbol{x}_{i+1};\sigma(t_{i+1}))$ ▷ 2^{nd} order correlation
    $\boldsymbol{d}_i' \leftarrow \left(\boldsymbol{x}_{i+1} - \hat{\boldsymbol{x}}_N^i\right)/\sigma(t_{i+1})$
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + (\sigma(t_{i+1}) - \sigma(t_i))\left(\frac{1}{2}\boldsymbol{d}_i + \frac{1}{2}\boldsymbol{d}_i'\right)$
  end if
  $\mathcal{L}_{obs} \leftarrow \frac{1}{n}\|\boldsymbol{y} - \hat{\boldsymbol{x}}_N^i\|_2^2$ ▷ Evaluated observation loss
  $\mathcal{L}_{pde} \leftarrow \frac{1}{m}\|\mathbf{0} - f(\hat{\boldsymbol{x}}_N^i)\|_2^2$ ▷ Evaluated PDE loss
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_{i+1} - \zeta_{obs}\nabla_{\boldsymbol{x}_i}\mathcal{L}_{obs} - \zeta_{pde}\nabla_{\boldsymbol{x}_i}\mathcal{L}_{pde}$
end for$\boldsymbol{x}_N$ ▷ Denoised data

3.3. Solving PDEs with Guided Diffusion

Our work focuses on two classes of PDEs: static PDEs and dynamic time-dependent PDEs. Static systems (e.g., Darcy Flow or Poisson equations) are defined by a time-independent function f:

$$f(\boldsymbol{c};\boldsymbol{a},\mathbf{u}) = 0 \ \text{ in } \ \Omega \subset \mathbb{R}^d, \quad \mathbf{u}(\boldsymbol{c}) = \boldsymbol{g}(\boldsymbol{c}) \ \text{ in } \ \partial\Omega, \quad (6)$$

where Ω is a bounded domain, $\boldsymbol{c} \in \Omega$ is a spatial coordinate, $\boldsymbol{a} \in \mathcal{A}$ is the PDE coefficient field, and $\mathbf{u} \in \mathcal{U}$ is the solution field. $\partial\Omega$ is the boundary of the domain Ω and $\mathbf{u}|_{\partial\Omega} = \boldsymbol{g}$ is the boundary constraint. We aim to recover both $\boldsymbol{a}$ and $\mathbf{u}$ from sparse observations on either $\boldsymbol{a}$ or $\mathbf{u}$ or both.

Similarly, we consider the dynamic systems (e.g., Navier-Stokes):

$$\begin{aligned} f(\boldsymbol{c},\tau;\boldsymbol{a},\mathbf{u}) &= 0, && \text{in } \Omega\times(0,\infty) \\ \mathbf{u}(\boldsymbol{c},\tau) &= \boldsymbol{g}(\boldsymbol{c},\tau), && \text{in } \partial\Omega\times(0,\infty) \\ \mathbf{u}(\boldsymbol{c},\tau) &= \boldsymbol{a}(\boldsymbol{c},\tau), && \text{in } \bar{\Omega}\times\{0\} \end{aligned} \quad (7)$$

where τ is a temporal coordinate, $\boldsymbol{a} = \mathbf{u}_0 \in \mathcal{A}$ is the initial condition, $\mathbf{u}$ is the solution field, and $\mathbf{u}|_{\Omega\times(0,\infty)} = \boldsymbol{g}$ is the boundary constraint. We aim to simultaneously recover both $\boldsymbol{a}$ and the solution $\mathbf{u}_T := \mathbf{u}(\cdot,T)$ at a specific time T from sparse observations on either $\boldsymbol{a}$, $\mathbf{u}_T$, or both.

Finally, we explore the recovery of the states across all timesteps $\mathbf{u}_{0:T}$ in 1D dynamic systems governed by Burger's equation. Our network D_θ models the distribution of all 1D states, including the initial condition $\mathbf{u}_0$ and solutions $\mathbf{u}_{1:T}$ stacked in the temporal dimension, forming a 2D dataset.

Guided Diffusion Algorithm In the data-driven PDE literature, the above tasks can be achieved by learning directional mappings between $\boldsymbol{a}$ and $\mathbf{u}$ (or u_T for dynamic systems). Thus, existing methods typically train separate neural networks for the forward solution operator $\mathcal{F}: \mathcal{A} \to \mathcal{U}$ and the inverse solution operator $\mathcal{I}: \mathcal{U} \to \mathcal{A}$.

Our method unifies the forward and inverse operators with a single network and an algorithm using the guided diffusion framework. DiffusionPDE can handle arbitrary sparsity patterns with one pre-trained diffusion model D_θ that learns the joint distribution of $\mathcal{A}$ and $\mathcal{U}$, concatenated on the channel dimension, denoted $\mathcal{X}$. Thus, our data $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X} := \mathcal{A} \times \mathcal{U}$. We follow the typical diffusion model procedures (Karras et al., 2022) to train our model on a family of PDEs.

Once we train the diffusion model D_θ, we employ our physics-informed DPS (Chung et al., 2022a) formulation during inference to guide the sampling of $\boldsymbol{x} \in \mathcal{X}$ that satisfies the sparse observations and the given PDE, as detailed in Algorithm 1. We follow Eq. 5 to modify the score function using the two guidance terms:

$$\begin{aligned} &\nabla_{\boldsymbol{x}_i}\log p(\boldsymbol{x}_i|\boldsymbol{y}_{obs}, f) \\ &\approx \nabla_{\boldsymbol{x}_i}\log p\big(\boldsymbol{x}_i\big) - \zeta_{obs}\nabla_{\boldsymbol{x}_i}\mathcal{L}_{obs} - \zeta_{pde}\nabla_{\boldsymbol{x}_i}\mathcal{L}_{pde}, \end{aligned} \quad (8)$$

where $\boldsymbol{x}_i$ is the noisy data at denoising step i, $\boldsymbol{y}_{obs}$ are the observed values, and $f(\cdot) = \mathbf{0}$ is the underlying PDE condition. $\mathcal{L}_{obs}$ and $\mathcal{L}_{pde}$ respectively represent the MSE loss of the sparse observations and the PDE equation residuals:

$$\begin{aligned} \mathcal{L}_{obs}(\boldsymbol{x}_i, \boldsymbol{y}_{obs}; D_\theta) &= \frac{1}{n}\|\boldsymbol{y}_{obs} - \hat{\boldsymbol{x}}_N^i\|_2^2 \\ &= \frac{1}{n}\sum_{j=1}^{n}(\boldsymbol{y}_{obs}(\boldsymbol{o}_j) - \hat{\boldsymbol{x}}_N^i(\boldsymbol{o}_j))^2, \\ \mathcal{L}_{pde}(\boldsymbol{x}_i; D_\theta, f) &= \frac{1}{m}\|\mathbf{0} - f(\hat{\boldsymbol{x}}_N^i)\|_2^2 \\ &= \frac{1}{m}\sum_j\sum_k f(\boldsymbol{c}_j, \tau_k; \hat{\mathbf{u}}_j, \hat{\boldsymbol{a}}_j)^2, \end{aligned} \quad (9)$$

where $\hat{\boldsymbol{x}}_N^i = D_\theta(\boldsymbol{x}_i)$ is the clean image estimate at denoising timestep i, which can be split into coefficient $\hat{\mathbf{u}}_i$ and solution $\hat{\boldsymbol{a}}_i$. Here, m is the total number of grid points (i.e., pixels), n is the number of sparse observation points. $\boldsymbol{o}_j$ represents the spatio-temporal coordinate of jth observation. Note that, without loss of generality, $\mathcal{L}_{pde}$ can be accumulated for all applicable PDE function f in the system, and the time component τ_k is ignored for static systems.

4. Experiments

4.1. PDE Problem Settings

We show the usefulness of DiffusionPDE across various PDEs for inverse and forward problems and compare it

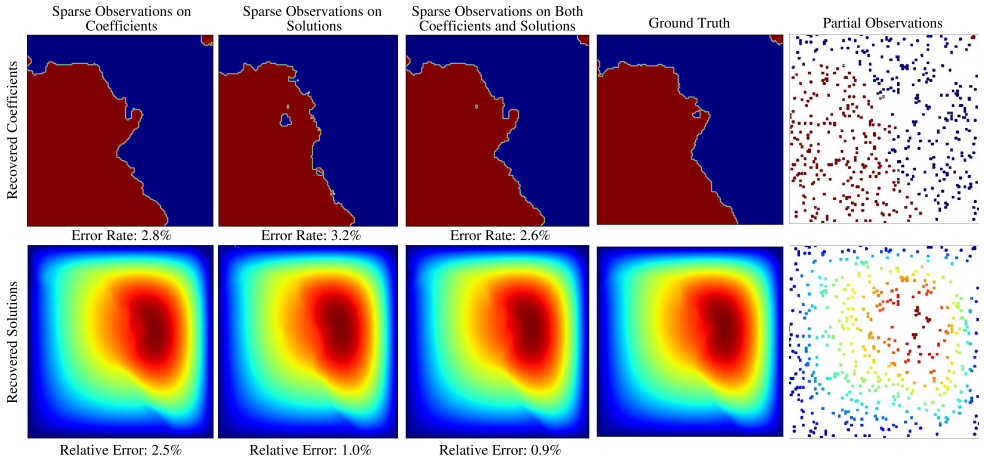


Figure 2. Different from forward and inverse PDE solvers, DiffusionPDE can take sparse observations on either the coefficient $\boldsymbol{a}$ or the solution $\mathbf{u}$ to recover both of them, using one trained network. Here, we show the recovered $\boldsymbol{a}$ and $\mathbf{u}$ of the Darcy's eqaution given sparse observations on $\boldsymbol{a}$, $\mathbf{u}$, or both. Compared with the ground truth, we see that our method successfully recovers the PDE in all cases.

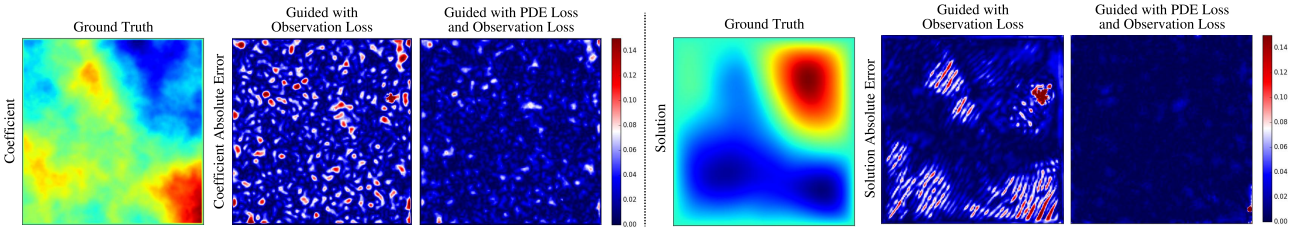


Figure 3. Usefulness of PDE loss. We visualize the absolute errors of the recovered coefficient and solution of the Helmholtz equation with and w/o PDE loss. We compare having only the observation loss with applying the additional PDE loss. The errors drop significantly when using PDE loss.

against recent learning-based techniques. We test on the following families of PDEs.

Darcy Flow. Darcy flow describes the movement of fluid through a porous medium. In our experiment, we consider the static Darcy Flow with a no-slip boundary $\partial\Omega$

$$\begin{aligned} -\nabla \cdot (\boldsymbol{a}(\boldsymbol{c})\nabla \mathbf{u}(\boldsymbol{c})) &= q(\boldsymbol{c}), & \boldsymbol{c} &\in \Omega \\ \mathbf{u}(\boldsymbol{c}) &= 0, & \boldsymbol{c} &\in \partial\Omega \end{aligned} \tag{10}$$

Here the coefficient $\boldsymbol{a}$ has binary values. We set $q(\boldsymbol{c}) = 1$ for constant force. The PDE guidance function is thus $f = \nabla \cdot (\boldsymbol{a}(\boldsymbol{c})\nabla \mathbf{u}(\boldsymbol{c})) + q(\boldsymbol{c})$.

Inhomogeneous Helmholtz Equation. We consider the static inhomogeneous Helmholtz Equation with a no-slip boundary on $\partial\Omega$, which describes wave propagation:

$$\begin{aligned} \nabla^2 \mathbf{u}(\boldsymbol{c}) + k^2 \mathbf{u}(\boldsymbol{c}) &= \boldsymbol{a}(\boldsymbol{c}), & \boldsymbol{c} &\in \Omega \\ \mathbf{u}(\boldsymbol{c}) &= 0, & \boldsymbol{c} &\in \partial\Omega \end{aligned} \tag{11}$$

The coefficient $\boldsymbol{a}$ is a piecewise constant function and k is a constant. Note 11 is the Poisson equation when $k = 0$. Setting $k = 1$ for Helmholtz equations, the PDE guidance function is $f = \nabla^2 \mathbf{u}(\boldsymbol{c}) + k^2 \mathbf{u}(\boldsymbol{c}) - \boldsymbol{a}(\boldsymbol{c})$.

Non-bounded Navier-Stokes Equation. We study the non-bounded incompressive Navier-Stokes equation regarding the vorticity.

$$\begin{aligned} \partial_t w(\boldsymbol{c}, \tau) + v(\boldsymbol{c}, \tau)\nabla w(\boldsymbol{c}, \tau) &= \nu \Delta w(\boldsymbol{c}, \tau) + q(\boldsymbol{c}), \\ \nabla \cdot v(\boldsymbol{c}, \tau) &= 0, \end{aligned} \tag{12}$$

Here $\boldsymbol{c} \in \Omega, \tau \in (0, T]$. $w = \nabla \times v$ is the vorticity, $v(\boldsymbol{c}, \tau)$ is the velocity at $\boldsymbol{c}$ at time τ, and $q(\boldsymbol{c})$ is a force field. We set the viscosity coefficient $\nu = 10^{-3}$ and correspondingly the Reynolds number $Re = \frac{1}{\nu} = 1000$.

DiffusionPDE learns the joint distribution of w_0 and w_T and we take $T = 10$ which simulates 1 second. Since $T \gg 0$, we cannot accurately compute the PDE loss from our model outputs. Therefore, given that $\nabla \cdot w(\boldsymbol{c}, \tau) = \nabla \cdot (\nabla \times v) = 0$, we use simplified $f = \nabla \cdot w(\boldsymbol{c}, \tau)$.

Bounded Navier-Stokes Equation. We study the bounded 2D imcompressive Navier Stokes regarding the velocity v and pressure p.

$$\begin{aligned} \partial_t v(\boldsymbol{c}, \tau) + v(\boldsymbol{c}, \tau)\nabla v(\boldsymbol{c}, \tau) + \frac{\nabla p}{\rho} &= \nu \nabla^2 v(\boldsymbol{c}, \tau), \\ \nabla \cdot v(\boldsymbol{c}, \tau) &= 0, \end{aligned} \tag{13}$$

Table 1. Relative errors of solutions (or final states) and coefficients (or initial states) when solving forward and inverse problems respectively with sparse observations. Error rates are used for the inverse problem of Darcy Flow.

		DiffusionPDE	PINO	DeepONet	PINNs	FNO
Darcy Flow	Forward	**2.5%**	35.2%	38.3%	48.8%	28.2%
	Inverse	**3.2%**	49.2%	41.1%	59.7%	49.3%
Poisson	Forward	**4.5%**	107.1%	155.5%	128.1%	100.9%
	Inverse	**20.0%**	231.9%	105.8%	130.0%	232.7%
Helmholtz	Forward	**8.8%**	106.5%	123.1%	142.3%	98.2%
	Inverse	**22.6%**	216.9%	132.8%	160.0%	218.2%
Non-bounded Navier-Stokes	Forward	**6.9%**	101.4%	103.2%	142.7%	101.4%
	Inverse	**10.4%**	96.0%	97.2%	146.8%	96.0%
Bounded Navier-Stokes	Forward	**3.9%**	81.1%	97.7%	100.1%	82.8%
	Inverse	**2.7%**	69.5%	91.9%	105.5%	69.6%

Here $\boldsymbol{c} \in \Omega, \tau \in (0, T]$. We set the viscosity coefficient $\nu = 0.001$ and the fluid density $\rho = 1.0$. We generate 2D cylinders of random radius at random positions inside the grid. Random turbulence flows in from the top of the grid, with the velocity field satisfying no-slip boundary conditions at the left and right edges, as well as around the cylinder $\partial\Omega_{left,right,cylinder}$. DiffusionPDE learns the joint distribution of v_0 and v_T at $T = 4$, which simulates 0.4 seconds. Therefore, we similarly use $f = \nabla \cdot v(\boldsymbol{c}, \tau)$ as before.

Burgers' Equation. We study the Burgers' equation with periodic boundary conditions on a 1D spatial domain of unit length $\Omega = (0, 1)$. We set the viscosity to $\nu = 0.01$. In our experiment, the initial condition u_0 has a shape of 128×1, and we take 127 more time steps after the initial state to form a 2D $u_{0:T}$ of size 128×128.

$$\begin{aligned} \partial_t u(\boldsymbol{c}, \tau) + \partial_{\boldsymbol{c}}(u^2(\boldsymbol{c}, \tau)/2) &= \nu \partial_{\boldsymbol{cc}} u(\boldsymbol{c}, \tau), \\ u(\boldsymbol{c}, 0) &= u_0(\boldsymbol{c}), \end{aligned} \tag{14}$$

Here $\boldsymbol{c} \in \Omega, \tau \in (0, T]$. We can reliably compute $f = \partial_t u(\boldsymbol{c}, \tau) + \partial_{\boldsymbol{c}}(u^2(\boldsymbol{c}, \tau)/2) - \nu \partial_{\boldsymbol{cc}} u(\boldsymbol{c}, \tau)$ with finite difference since we model densely on the time dimension.

4.2. Dataset Preparation and Training

We first test DiffusionPDE on jointly learning the forward mapping $\mathcal{F} : \mathcal{A} \to \mathcal{U}$ and the inverse mapping $\mathcal{I} : \mathcal{U} \to \mathcal{A}$ given sparse observations. In our experiments, we define our PDE over the unit square $\Omega = (0, 1)^2$, which we represent as a 128×128 grid. We utilize Finite Element Methods (FEM) to generate our training data. Specifically, we run FNO's (Li et al., 2020) released scripts to generate Darcy Flows and the vorticities of the Navier-Stokes equation. Similarly, we generate the dataset of Poisson and Helmholtz using second-order finite difference schemes. To add more complex boundary conditions, we use Difftaichi (Hu et al., 2019) to generate the velocities of the bounded Navier-Stokes equation. We train the joint diffusion model for each PDE on three A40 GPUs for approximately 4 hours, using 50,000 data pairs. For Burgers' equation, we train the diffusion model on a dataset of 50,000 samples produced as outlined in FNO (Li et al., 2020). We randomly select 5 out of 128 spatial points on Ω to simulate sensors that provide measurements across time.

4.3. Baseline Methods

We compare DiffusionPDE with state-of-the-art learning-based methods, including PINO (Li et al., 2021), DeepONet (Lu et al., 2021), PINNs (Raissi et al., 2019), and FNO (Li et al., 2020). However, note that none of these methods show operation on partial observations. These methods can learn mappings between $\boldsymbol{a}$ and $\mathbf{u}$ or $\mathbf{u}_0$ and $\mathbf{u}_{1:T}$ with full observations, allowing them to also solve the mapping between $\mathbf{u}_0$ and $\mathbf{u}_T$. PINNs map input $\boldsymbol{a}$ to output $\mathbf{u}$ by optimizing a combined loss function that incorporates both the solution $\mathbf{u}$ and the PDE residuals. DeepONet employs a branch network to encode input function values sampled at discrete points and a trunk network to handle the coordinates of the evaluated outputs. FNO maps from the parametric space to the solution space using Fourier transforms. PINO enhances FNO by integrating PDE loss during training and refining the model with PDE loss finetuning. We train all four baseline methods on both forward and inverse mappings using full observation of $\boldsymbol{a}$ or $\mathbf{u}$ for both static and dynamic PDEs. We tried training the baseline models on partial observations, but we noticed degenerate training outcomes (see supplementary for details). Overall, they are intended for *full observations* and may not be suitable for sparse measurements.

More closely related to our method, GraphPDE (Zhao et al., 2022) demonstrates the ability to recover the initial state using sparse observations on the final state, a task that other baselines struggle with. Therefore, we compare against GraphPDE for the inverse problem of bounded Navier-Stokes (NS) equation, which is the setup used in their re-

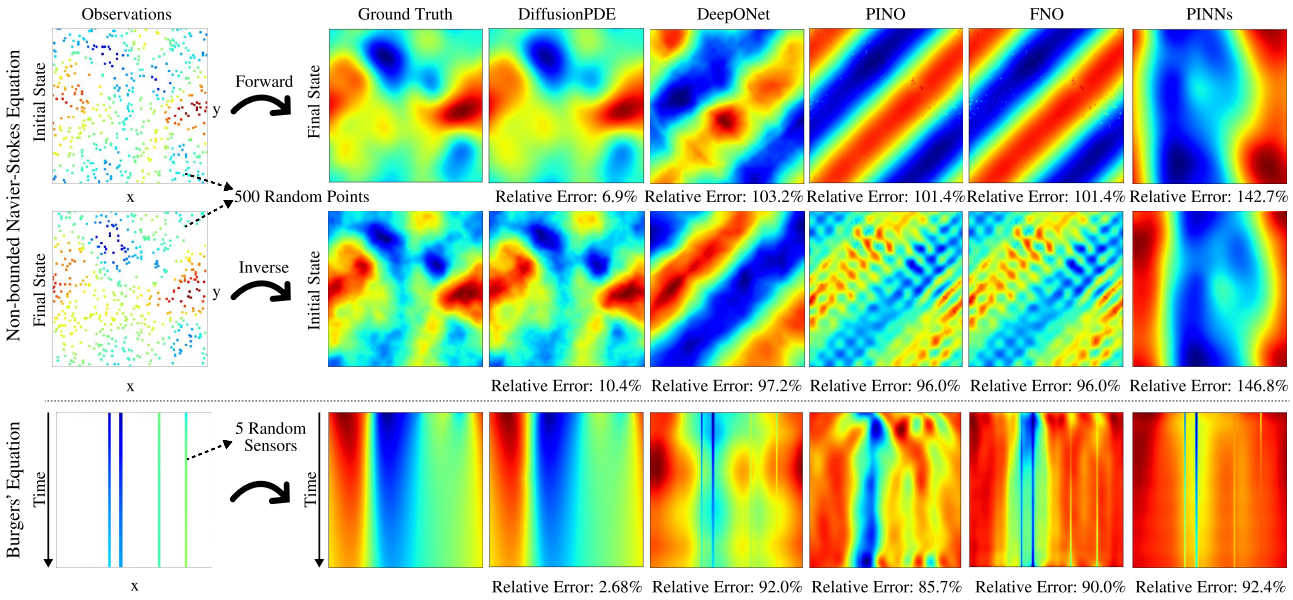


Figure 4. We compare DiffusionPDE with state-of-the-art neural PDE solvers (Li et al., 2020; 2021; Lu et al., 2021; Raissi et al., 2019). In the forward Navier-Stokes problem, we give 500 sparse observations of the initial state to solve for the final state. In the inverse set-up, we take observations of the final state and solve for the initial. For the Burgers' equation, we use 5 sensors throughout all time steps and want to recover the solution at all time steps. Note that we train on neighboring snapshot pairs for the baselines in order to add continuous observations of the Burgers' equation. Results show that existing methods do not support PDE solving under sparse observations, and we believe they are not easily extendable to do so. We refer readers to the supplementary for a complete set of visual results.

port. GraphPDE uses a trained latent space model and a bounded forward GNN model to solve the inverse problem with sparse sensors and thus is incompatible with unbounded Navier-Stokes. We create bounded meshes using our bounded grids to train the GNN model and train the latent prior with $v_{0:T}$ for GraphPDE.

4.4. Main Evaluation Results

We respectively address the forward problem and the inverse problem with sparse observations of $\boldsymbol{a}$ or $\mathbf{u}$. For the forward problem, we randomly select coefficients (initial states) as sparse observations and then compare the predicted solutions (final states) with the ground truth. Specifically, we select 500 out of 128×128 points, approximately 3%, on the coefficients of Darcy Flow, Poisson equation, Helmholtz equation, and the initial state of the non-bounded Navier-Stokes equation. For the bounded Navier-Stokes equation, we use 1% observed points beside the boundary of the cylinder in 2D. Similarly, for the inverse problem, we randomly sample points on solutions (final states) as sparse observations, using the same number of observed points as in the forward model for each PDE.

We show the relative errors of all methods regarding both forward and inverse problems in Table 1. Since the coefficients of Darcy Flow are binary, we evaluate the error rates of our prediction. Non-binary data is evaluated using mean pixel-wise relative error. We report error numbers averaged across 1,000 random scenes and observations for each PDE. DiffusionPDE outperforms other methods including PINO (Li et al., 2021), DeepONet (Lu et al., 2021), PINNs (Raissi et al., 2019), and FNO (Li et al., 2020) for both directions with sparse observations, demonstrating the novelty and uniqueness of our approach. For the inverse problems of the Poisson and Helmholtz equations, DiffusionPDE exhibits higher error rates due to the insufficient constraints within the coefficient space, produced from random fields. In Fig. 4, we visualize the results for solving both the forward and inverse problem of the non-bounded Navier-Stokes. We refer to the *supplementary* for additional visual results. While other methods may produce partially correct results, DiffusionPDE outperforms them and can recover results very close to the ground truth.

For the inverse problem of the bounded Navier-Stokes equation, we further compare DiffusionPDE with GraphPDE, as illustrated in Fig. 5. Our findings reveal that DiffusionPDE surpasses GraphPDE (Zhao et al., 2022) in accuracy, reducing the relative error from 12.0% to 2.7% with only 1% observed points.

We further show whether DiffusionPDE can jointly recover both $\boldsymbol{a}$ and $\mathbf{u}$ by analyzing the retrieved $\boldsymbol{a}$ and $\mathbf{u}$ with sparse observations on different sides as well as on both sides. In Fig. 2, we recover the coefficients and solutions of Darcy Flow by randomly observing 500 points on only coefficient space, only space solution space, and both. Both coefficients and solutions can be recovered with low errors for

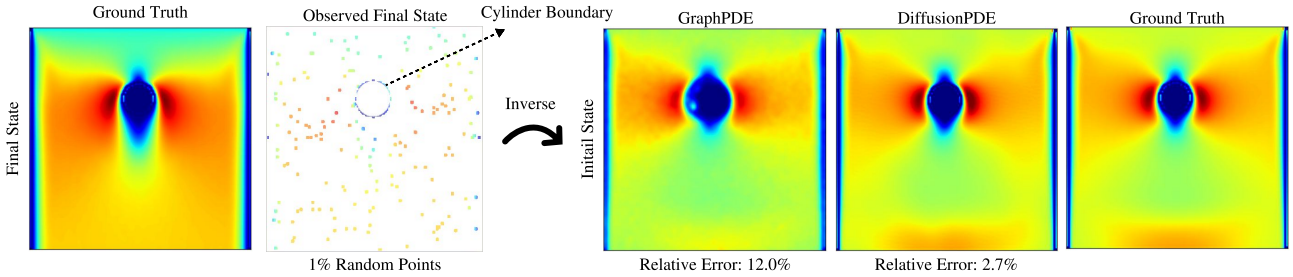


Figure 5. We compare GraphPDE (Zhao et al., 2022) and our method for solving the inverse bounded Navier-Stokes equation. Given the boundary conditions and 1% observations of the final vorticity field, we solve the initial vorticity field. We set the fliuds to flow in from the top, with boundary conditions at the edges and a middle cylinder. While GraphPDE can recover the overall pattern of the initial state, it suffers from noise when the fluid passes the cylinder and misses the high vorticities at the bottom.

each situation. We therefore conclude that DiffusionPDE can solve the forward problem and the inverse problem simultaneously with sparse observations at any side without retraining our network.

4.5. Recovering Solutions Throughout a Time Interval

We demonstrate that DiffusionPDE is capable of retrieving all time steps throughout the time interval $[0, T]$ from continuous observations on sparse sensors. To evaluate its ability to recover $u_{0:T}$ with sparse sensors, we study the 1D dynamic Burgers' equation, where DiffusionPDE learns the distribution of $u_{0:T}$ using a 2D diffusion model. To apply continuous observation on PINO, DeepONet, FNO, and PINNs, we train them on neighboring snapshot pairs. Our experiment results in a test relative error of 2.68%, depicted in Fig. 4, which is significantly lower than other methods.

4.6. Additional Analysis

We examine the effects of different components of our algorithm such as PDE loss and observation samplings. We strongly encourage readers to view the supplementary for more details of these analyses as well as additional experiments.

PDE Loss. To verify the role of the PDE guidance loss of Eq. 8 during the denoising process, we visualize the errors of recovered $\boldsymbol{a}$ and $\mathbf{u}$ of Helmholtz equation with or without PDE loss. Here, we run our DPS algorithm with 500 sparse observed points on both the coefficient $\boldsymbol{a}$ and solution $\mathbf{u}$ and study the effect of the additional PDE loss guidance. The relative error of $\mathbf{u}$ reduces from 9.3% to 0.6%, and the relative error of $\boldsymbol{a}$ reduces from 13.2% to 9.4%. Therefore, we conclude that PDE guidance helps smooth the prediction and improve the accuracy.

Number of Observations. We examine the results of DiffusionPDE in solving forward and inverse problems when there are 100, 300, 500, and 1000 random observations on $\boldsymbol{a}$, $\mathbf{u}$, or both $\boldsymbol{a}$ and $\mathbf{u}$. The error of DiffusionPDE decreases as the number of sparse observations increases. DiffusionPDE is capable of recovering both $\boldsymbol{a}$ and $\mathbf{u}$ with errors $1\% \sim 10\%$ with approximately 6% observation points at any side for most PDE families. DiffusionPDE becomes insensitive to the number of observations and can solve the problems well once more than 3% of the points are observed.

Observation Sampling Pattern We show that DiffusionPDE is robust to different sampling patterns of the sparse observations, including grid and non-uniformly concentrated patterns. Note that even when conditioned on the full observations, our approach performs on par with the current best methods, likely due to the inherent resilience of our guided diffusion algorithm.

5. Conclusion and Future Work

In this work, we develop DiffusionPDE, a diffusion-based PDE solver that addresses the challenge of solving PDEs from partial observations by filling in missing information using generative priors. We formulate a diffusion model that learns the joint distribution of the coefficient (or initial state) space and the solution (or final state) space. During the sampling process, DiffusionPDE can flexibly generate plausible data by guiding its denoising with sparse measurements and PDE constraints. Our new approach leads to significant improvements over existing state-of-the-art methods, advancing toward a general PDE-solving framework that leverages the power of generative models.

Several promising directions for future research have emerged from this work. Currently, DiffusionPDE is limited to solving slices of 2D dynamic PDEs; extending its capabilities to cover full time intervals of these equations presents a significant opportunity. Moreover, the model's struggle with accuracy in spaces that lack constraints is another critical area for exploration. DiffusionPDE also suffers from a slow sampling procedure, and a faster solution might be desired.

Impact Statement

Our work, which concentrates on solving both forward and inverse problems of partial differential equations (PDEs) with sparse observations, promises to significantly advance scientific and engineering disciplines that rely on these computations. The potential negative impacts are minimal, as this research primarily aims to provide more accurate and efficient computational methods without adverse effects on existing systems or practices.

References

Aliabadi, F. M. Boundary element methods. In *Encyclopedia of continuum mechanics*, pp. 182–193. Springer, 2020.

Bonev, B., Kurth, T., Hundt, C., Pathak, J., Baust, M., Kashinath, K., and Anandkumar, A. Spherical fourier neural operators: Learning stable dynamics on the sphere. In *International conference on machine learning*, pp. 2806–2823. PMLR, 2023.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021a.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021b.

Chen, P. Y., Xiang, J., Cho, D. H., Chang, Y., Pershing, G., Maia, H. T., Chiaramonte, M. M., Carlberg, K., and Grinspun, E. Crom: Continuous reduced-order modeling of pdes using implicit neural representations. *arXiv preprint arXiv:2206.02607*, 2022.

Cho, M., Jadamba, B., Kahler, R., Khan, A., and Sama, M. First-order and second-order adjoint methods for the inverse problem of identifying non-linear parameters in pdes. *Industrial Mathematics and Complex Systems: Emerging Mathematical Models, Methods and Algorithms*, pp. 147–163, 2017.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022a.

Chung, H., Sim, B., Ryu, D., and Ye, J. C. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022b.

de Hoop, M. V., Lassas, M., and Wong, C. A. Deep learning architectures for nonlinear operator functions and nonlinear inverse problems. *Mathematical Statistics and Learning*, 4(1):1–86, 2022.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Eivazi, H., Tahani, M., Schlatter, P., and Vinuesa, R. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Physics of Fluids*, 34(7), 2022.

Evans, L. C. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

Fox, C. and Nicholls, G. Statistical estimation of the parameters of a pde. *Can. appl. Math. Quater*, 10:277–810, 2001.

Harrach, B. An introduction to finite element methods for inverse coefficient problems in elliptic pdes. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 123 (3):183–210, 2021.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J., and Durand, F. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Iakovlev, V., Heinonen, M., and Lähdesmäki, H. Learning continuous-time pdes from sparse data with graph neural networks. *arXiv preprint arXiv:2006.08956*, 2020.

Idelsohn, S. R., Onate, E., Calvo, N., and Del Pin, F. The meshless finite element method. *International Journal for Numerical Methods in Engineering*, 58(6):893–912, 2003.

Jacobsen, C., Zhuang, Y., and Duraisamy, K. Cocogen: Physically-consistent and conditioned score-based generative models for forward and inverse problems. *arXiv preprint arXiv:2312.10527*, 2023.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022.

Kumar, K. and Choi, Y. Accelerating particle and fluid simulations with differentiable graph networks for solving forward and inverse problems. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, pp. 60–65, 2023.

Lehmann, F., Gatti, F., Bertin, M., and Clouteau, D. Fourier neural operator surrogate model to predict 3d seismic waves propagation. *arXiv preprint arXiv:2304.10242*, 2023.

Li, B., Wang, H., Feng, S., Yang, X., and Lin, Y. Solving seismic wave equations on variable velocity models with fourier neural operator. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–18, 2023a.

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

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 2021.

Li, Z., Peng, W., Yuan, Z., and Wang, J. Fourier neural operator approach to large eddy simulation of three-dimensional turbulence. *Theoretical and Applied Mechanics Letters*, 12(6):100389, 2022.

Li, Z., Yang, G., Deng, X., De Sa, C., Hariharan, B., and Marschner, S. Neural caches for monte carlo partial differential equation solvers. In *SIGGRAPH Asia 2023 Conference Papers*, pp. 1–10, 2023b.

Li, Z., Kovachki, N., Choy, C., Li, B., Kossaifi, J., Otta, S., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36, 2024.

Lienen, M., Lüdke, D., Hansen-Palmus, J., and Günnemann, S. From zero to turbulence: Generative modeling for 3d flow simulation. In *The Twelfth International Conference on Learning Representations*, 2023.

Lippe, P., Veeling, B., Perdikaris, P., Turner, R., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36, 2024.

Long, D. and Zhe, S. Invertible fourier neural operators for tackling both forward and inverse problems. *arXiv preprint arXiv:2402.11722*, 2024.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Mao, Z., Jagtap, A. D., and Karniadakis, G. E. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

Misyris, G. S., Venzke, A., and Chatzivasileiadis, S. Physics-informed neural networks for power systems. In *2020 IEEE power & energy society general meeting (PESGM)*, pp. 1–5. IEEE, 2020.

Molinaro, R., Yang, Y., Engquist, B., and Mishra, S. Neural inverse operators for solving pde inverse problems. *arXiv preprint arXiv:2301.11167*, 2023.

Müller, T., Rousselle, F., Novák, J., and Keller, A. Real-time neural radiance caching for path tracing. *arXiv preprint arXiv:2106.12372*, 2021.

Nam, H. C., Berner, J., and Anandkumar, A. Solving poisson equations using neural walk-on-spheres. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, 2024.

Omori, K. and Kotera, J. Overview of pdes and their regulation. *Circulation research*, 100(3):309–327, 2007.

Pakravan, S., Mistani, P. A., Aragon-Calvo, M. A., and Gibou, F. Solving inverse-pde problems with physics-aware neural networks. *Journal of Computational Physics*, 440: 110414, 2021.

Peng, W., Yuan, Z., Li, Z., and Wang, J. Linear attention coupled fourier neural operator for simulation of three-dimensional turbulence. *Physics of Fluids*, 35(1), 2023.

Po, R., Yifan, W., Golyanik, V., Aberman, K., Barron, J. T., Bermano, A. H., Chan, E. R., Dekel, T., Holynski, A., Kanazawa, A., et al. State of the art on diffusion models for visual computing. *arXiv preprint arXiv:2310.07204*, 2023.

Quarteroni, A. and Valli, A. *Numerical approximation of partial differential equations*, volume 23. Springer Science & Business Media, 2008.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Serrano, L., Le Boudec, L., Kassaï Koupaï, A., Wang, T. X., Yin, Y., Vittaut, J.-N., and Gallinari, P. Operator learning with neural fields: Tackling pdes on general geometries. *Advances in Neural Information Processing Systems*, 36, 2024.

Shu, D., Li, Z., and Farimani, A. B. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972, 2023.

Ŝolín, P. *Partial differential equations and the finite element method*. John Wiley & Sons, 2005.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Strauss, W. A. *Partial differential equations: An introduction*. John Wiley & Sons, 2007.

van Leeuwen, T. and Herrmann, F. J. A penalty method for pde-constrained optimization in inverse problems. *Inverse Problems*, 32(1):015007, 2015.

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Xu, S., Huang, Y., Pan, J., Ma, Z., and Chai, J. Inversion-free image editing with natural language. *arXiv preprint arXiv:2312.04965*, 2023.

Yang, G. and Sommer, S. A denoising diffusion model for fluid field prediction. *arXiv preprint arXiv:2301.11661*, 2023.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

Zhao, Q., Lindell, D. B., and Wetzstein, G. Learning to solve pde-constrained inverse problems with graph networks. *arXiv preprint arXiv:2206.00711*, 2022.

Appendix

A. Overview

In this supplementary material, we provide detailed additional content that complements the main paper. Section B elaborates on the implementation of our data generation process. Section C includes the detailed implementation of our sampling progress, while Section D delves into the reductions in error achieved by integrating PDE loss. Section E includes comprehensive visual results for both forward and inverse computations using sparse observations, not featured in the main text. In Section F, we discuss results from full observation scenarios for all methods. Section G justifies our choice of training the baselines on complete observation data. To evaluate the stability of DiffusionPDE, Section H includes the standard deviation of DiffusionPDE, and Section I examines the robustness of our DiffusionPDE model against random noise or different observation locations. Section J explores the effects of varying observation numbers on result accuracy, providing a deeper understanding of our model's performance under different conditions. Finally, Section K studies the performance of DiffusionPDE across different resolutions.

B. Data Generation Details

We generate 50,000 samples for each PDE and all diffusion models are trained on Nvidia A40 GPUs.

B.1. Static PDEs

We derived the methods of data generation for static PDEs from (Li et al., 2020). We first generate Gaussian random fields on $(0,1)^2$ so that $\mu \sim \mathcal{N}(0, (-\Delta + 9\mathbf{I})^{-2})$. For Darcy Flow, we let $a = f(\mu)$ so that:

$$\begin{cases} a(x) = 12, & \text{if } \mu(x) \geq 0 \\ a(x) = 3, & \text{if } \mu(x) < 0 \end{cases}$$

For the Poisson equation and Helmholtz equation, we let $a = \mu$ as the coefficients. We then use second-order finite difference schemes to solve the solution u and enforce the no-slip boundary condition for solutions by multiplying a mollifier $\sin(\pi x_1)\sin(\pi x_2)$ for point $x = (x_1, x_2) \in (0,1)^2$. Both a and u have resolutions of 128×128.

B.2. Non-bounded Navier-Stokes Equation

We derived the method to generate non-bounded Navier-Stokes equation from (Li et al., 2020). The initial condition w_0 is generated by Gaussian random field $\mathcal{N}(0, 7^{1.5}(-\Delta + 49\mathbf{I})^{-2.5})$. The forcing function follows the fixed pattern for point (x_1, x_2):

$$q(x) = \frac{1}{10}(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$$

We then use the pseudo-spectral method to solve the Navier-Stokes equations in the stream-function formulation. We transform the equations into the spectral domain using Fourier transforms, solving the vorticity equation in the spectral domain, and then using inverse Fourier transforms to compute nonlinear terms in physical space. We simulate for 1 second with 10 timesteps, and w_t has a resolution of 128×128.

B.3. Bounded Navier-Stokes Equation

We use Difftaichi (Hu et al., 2019) to generate data for the bounded Navier-Stokes equation. Specifically, we apply the Marker-and-Cell (MAC) method by solving a pressure-Poisson equation to enforce incompressibility and iterating through predictor and corrector steps to update the velocity and pressure fields. The grid is of the resolution 128×128 and the center of the cylinder is at a random location in $[30, 60] \times [30, 90]$ with a random radius in $[5, 20]$. The fluid flows into the grid from the upper boundary with a random initial vertical velocity in $[0.5, 3]$. We simulate for 1 second with 10 timesteps and study steps 4 to 8 when the turbulence is passing the cylinder.

B.4. Burgers' Equation

We derived the method to generate Burgers' equation from (Li et al., 2020). The initial condition u_0 is generated by Gaussian random field $\mathcal{N}(0, 625(-\Delta + 25\mathbf{I})^{-2})$. We solve the PDE with a spectral method and simulate 1 second with 127 additional timesteps. The final $u_{0:T}$ space has a resolution of 128×128.

C. Guided Sampling Details

For experiments with sparse observations or sensors, we find that DiffusionPDE performs the best when weights ζ are selected as shown in Table 2. During the initial 80% of iterations in the sampling process, guidance is exclusively provided by the observation loss $\mathcal{L}_{obs}$. Subsequently, after 80% of the iterations have been completed, we introduce the PDE loss $\mathcal{L}_{pde}$, and reduce the weighting factor ζ_{obs} for the observation loss, by a factor of 10. This adjustment shifts the primary guiding influence to the PDE loss, thereby aligning the diffusion model more closely with the dynamics governed by the partial differential equations.

Table 2. The weights assigned to the PDE loss and the observation loss vary depending on whether the observations pertain to the coefficients (or initial states) a or to the solutions (or final states) u.

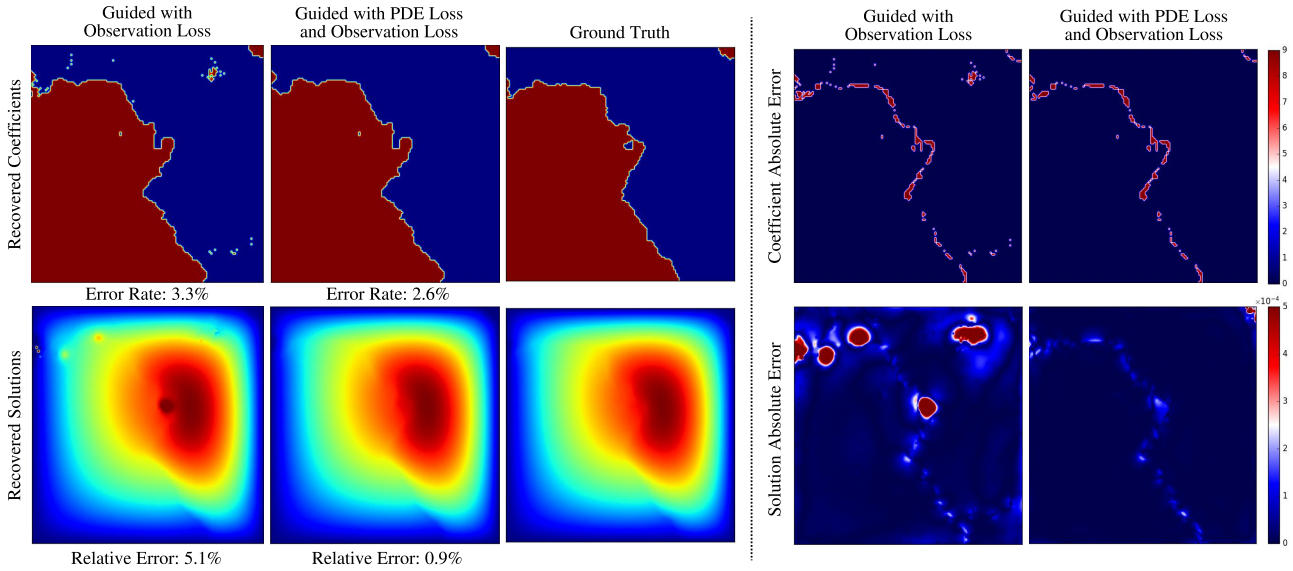
		Darcy Flow	Poisson	Helmholtz	Non-bounded Navier-Stokes	Bounded Navier-Stokes	Burgers' equation
ζ_{obs}	a	2.5×10^3	4×10^2	2×10^2	5×10^2	2.5×10^2	3.2×10^2
	u	10^6	2×10^4	3×10^4	5×10^2	2.5×10^2	-
ζ_{pde}		10^3	10^2	10^2	10^2	10^2	10^2

D. Improvement in Prediction through PDE Loss Term

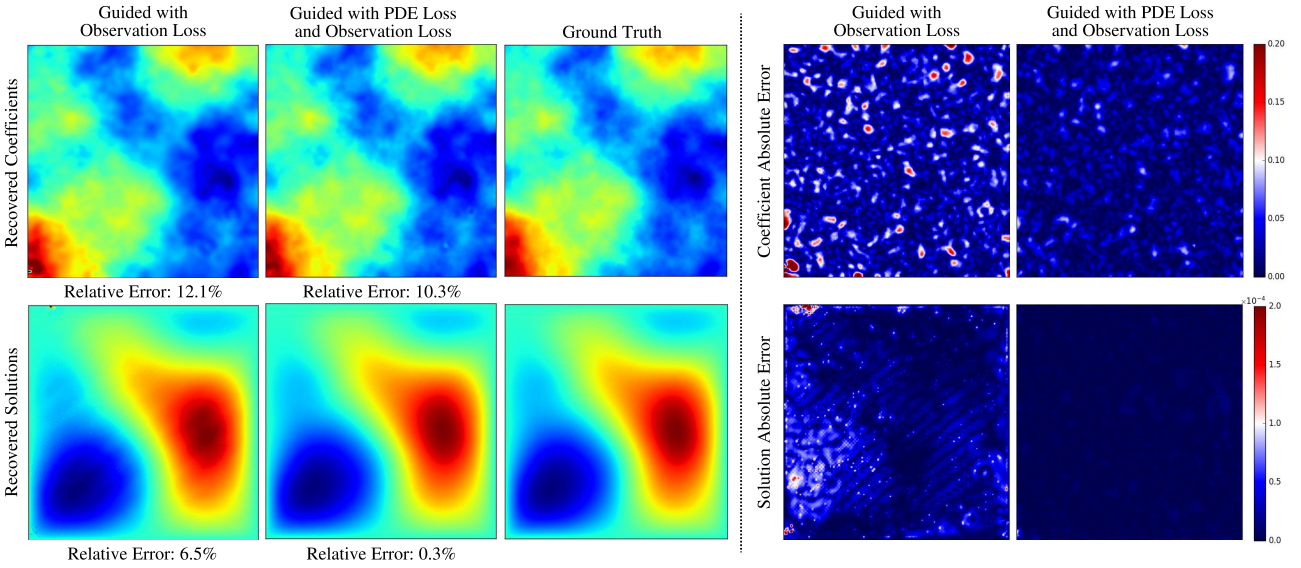
DiffusionPDE performs better when we apply the PDE loss term $\mathcal{L}_{pde}$ in addition to the observation loss term $\mathcal{L}_{obs}$ as guidance, as shown in Table 3. The errors in both the coefficients (initial states) a and the solutions (final states) u significantly decrease. We also visualize the recovered a and u and corresponding absolute errors of Darcy Flow, Poisson equation, and Helmholtz equation in Fig. 6. It is demonstrated that the prediction becomes more accurate with the combined guidance of PDE loss and observation loss than with only observation loss.

Table 3. DiffusionPDE' prediction errors of coefficients (initial states) a and solutions (final states) u with sparse observation on both a and u, guided by different loss functions.

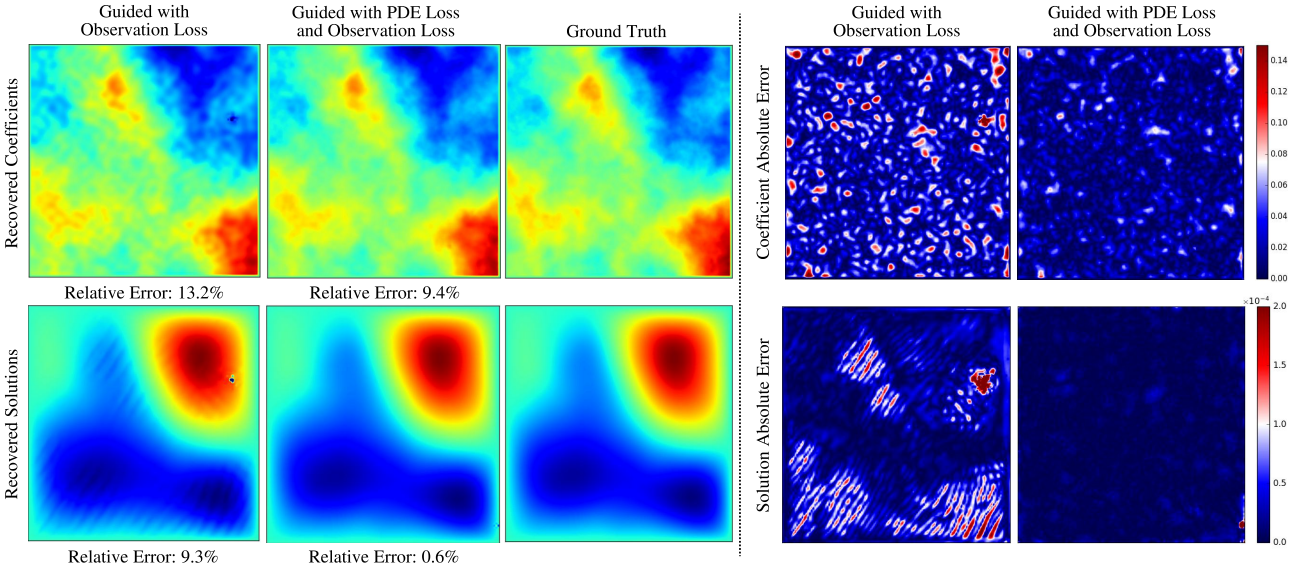
Loss Function	Side	Darcy Flow	Poisson	Helmholtz	Non-bounded Navier-Stokes	Bounded Navier-Stokes
$\mathcal{L}_{obs}$	a	4.6%	12.1%	13.2%	8.2%	6.4%
	u	4.8%	6.5%	9.3%	7.6%	3.3%
$\mathcal{L}_{obs} + \mathcal{L}_{pde}$	a	3.4%	10.3%	9.4%	4.9%	1.7%
	u	1.7%	0.3%	0.6%	0.6%	1.4%



(a) Recovered coefficients, solutions, and corresponding absolute errors of Darcy Flow.



(b) Recovered coefficients, solutions, and corresponding absolute errors of Poisson equation.



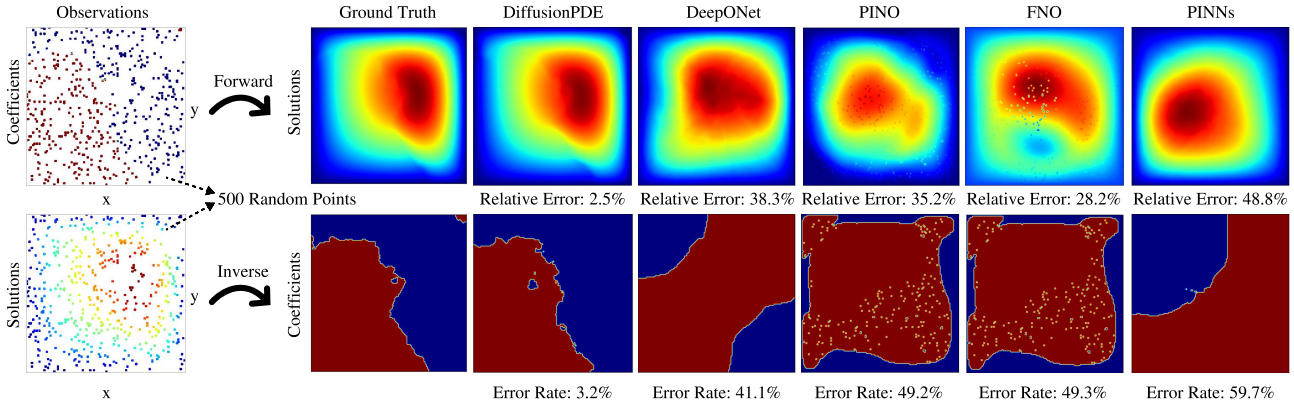
(c) Recovered coefficients, solutions, and corresponding absolute errors of Helmholtz equation.

Figure 6. Recovered coefficients, solutions, and their corresponding visualized absolute errors for various PDE families.

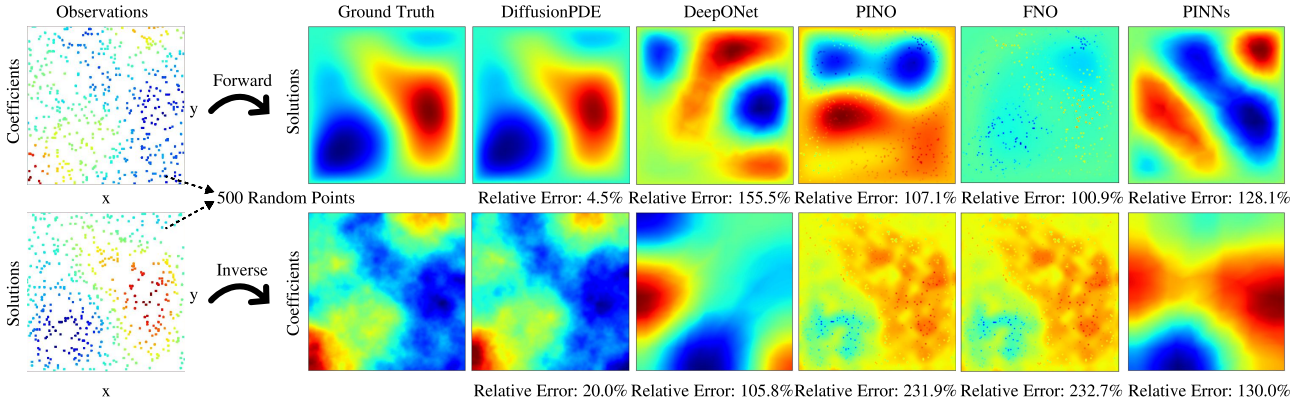
E. Additional Results on All PDEs with Sparse Observation

We present the recovered results of another Burgers' equation in Fig. 8. DiffusionPDE outperforms all other methods with 5 sensors for continuous observation. We also present the recovered results for both the forward and inverse problems of all other PDEs with sparse observations, as shown in Fig. 7. Specifically, we solve the forward and inverse problems for the Darcy Flow, Poisson equation, Helmholtz equation, and non-bounded Navier-Stokes equation using 500 random points observed in either the solution space or the coefficient space. Additionally, for the bounded Navier-Stokes equation, we observe 1% of the points in the velocity field. Our findings indicate that DiffusionPDE outperforms all other methods, providing the most accurate solutions.

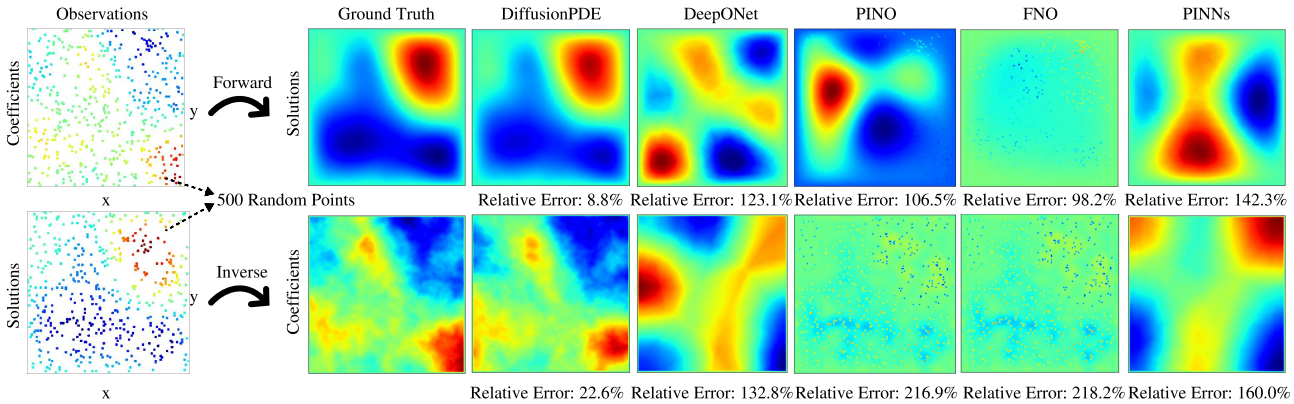
Additional Data Setting for Darcy Flow To further demonstrate the generalization capability of our model, we conducted additional tests on different data settings for Darcy Flow. In Fig. 9, we solve the forward and inverse problems of Darcy Flow with 500 observation points, adjusting the binary values of a to 20 and 16 instead of the original 12 and 3 in Section B. Our results indicate that DiffusionPDE performs equally well under these varied data settings, showcasing its robustness and adaptability.



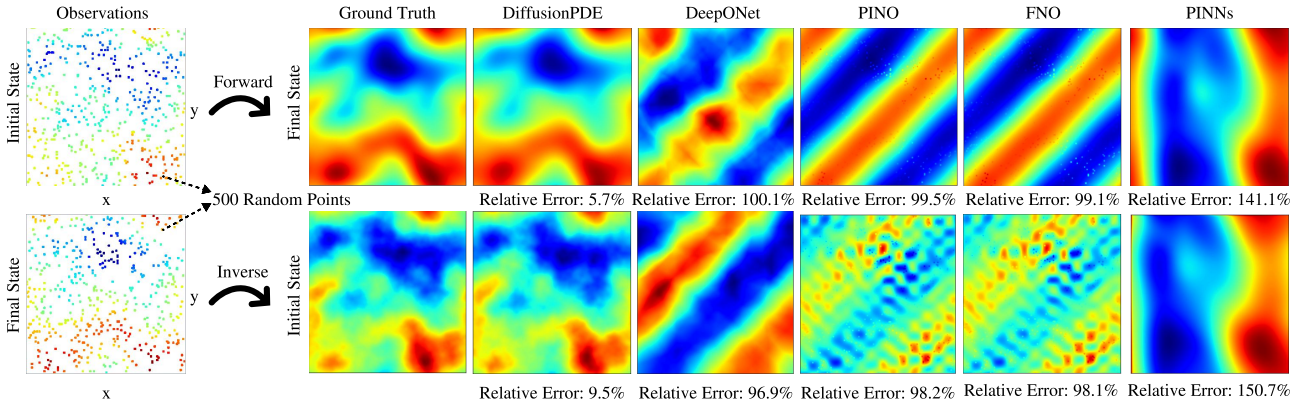
(a) Forward and inverse results of Darcy Flow recovered by 500 observation points.



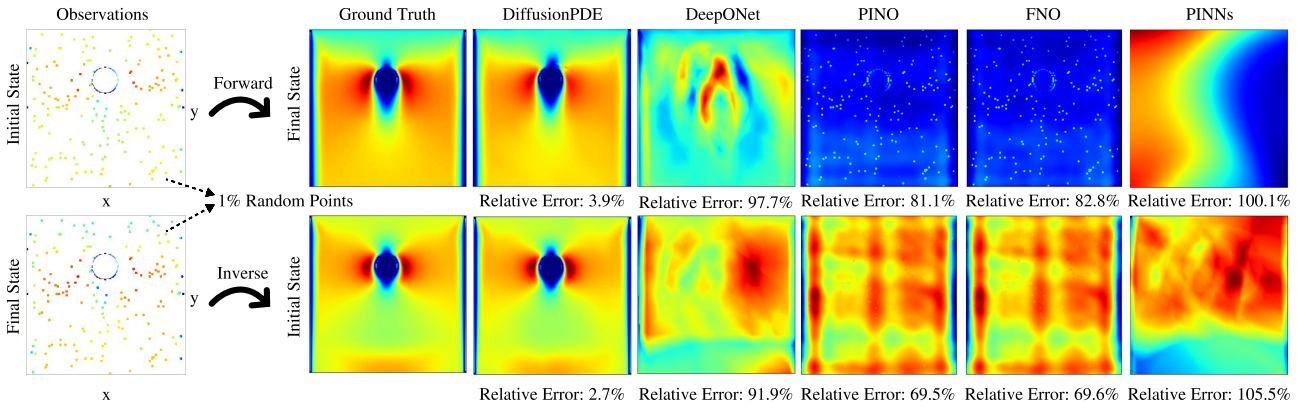
(b) Forward and inverse results of Poisson equation recovered by 500 observation points.



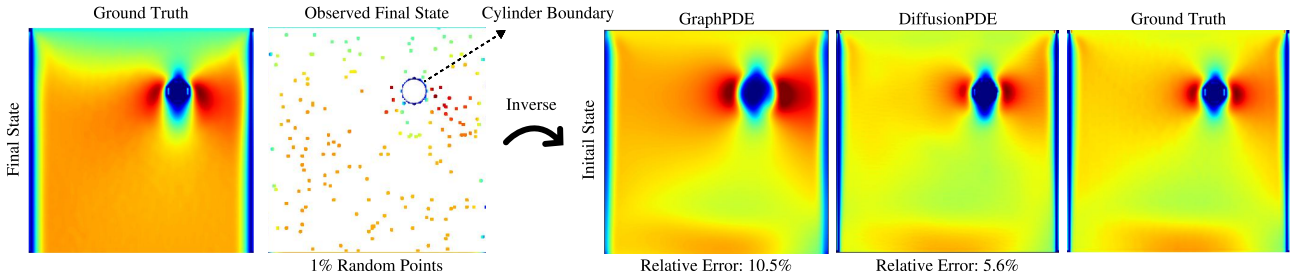
(c) Forward and inverse results of Helmholtz equation recovered by 500 observation points.



(d) Forward and inverse results of another non-bounded Navier-Stokes equation recovered by 500 observation points.



(e) Forward and inverse results of bounded Navier-Stokes equation recovered by 1% observation points.



(f) Inverse results of DiffusionPDE and GraphPDE of another bounded Navier-Stokes equation recovered by 1% observation points and the known boundary of the cylinder.

Figure 7. Results of forward and inverse problems for different PDE families with sparse observation.

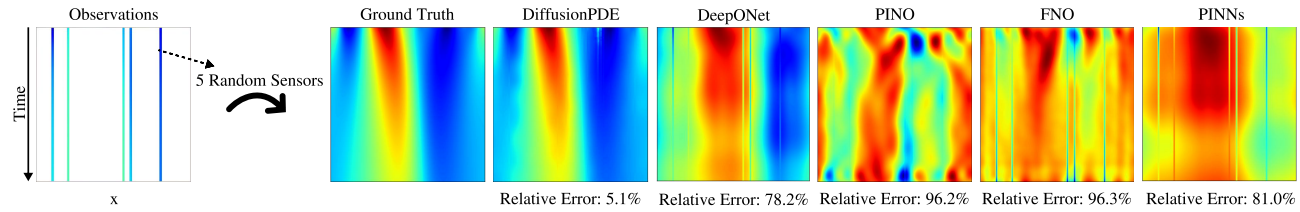


Figure 8. Results of another Burgers' equation recovered by 5 sensors throughout the time interval.

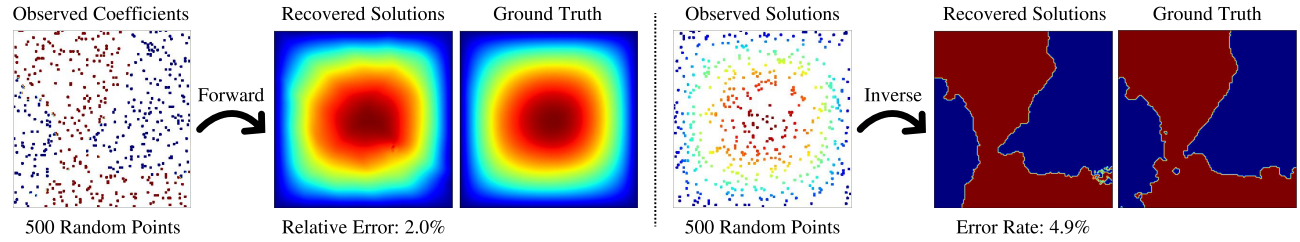


Figure 9. Forward and inverse results of Darcy Flow recovered by 500 observation points under a different data setting.

F. Solving Forward and Inverse Problems with Full Observation

We have also included the errors of all methods when solving both the forward and inverse problems with full observation, as displayed in Table 4. In general, DiffusionPDE and PINO outperform all other methods, and DiffusionPDE performs the best for all static PDEs. DiffusionPDE is capable of solving both forward and inverse problems with errors of less than 10% for all classes of discussed PDEs and is comparable to the state-of-the-art.

Table 4. Relative errors of solutions (or final states) and coefficients (or initial states) when solving forward and inverse problems with full observations. Error rates are used for the inverse problem of Darcy Flow.

		DiffusionPDE	PINO	DeepONet	PINNs	FNO
Darcy Flow	Forward	**2.2%**	4.0%	12.3%	15.4%	5.3%
	Inverse	**2.0%**	2.1%	8.4%	10.1%	5.6%
Poisson	Forward	**2.7%**	3.7%	14.3%	16.1%	8.2%
	Inverse	**9.8%**	10.2%	29.0%	33.1%	13.6%
Helmholtz	Forward	**2.3%**	4.9%	17.8%	18.1%	11.1%
	Inverse	**4.0%**	4.9%	28.1%	29.2%	5.0%
Non-bounded Navier-Stokes	Forward	6.1%	**1.1%**	25.6%	27.3%	2.3%
	Inverse	8.6%	**6.8%**	19.6%	27.8%	6.8%
Bounded Navier-Stokes	Forward	**1.7%**	1.9%	13.3%	18.6%	2.0%
	Inverse	**1.4%**	2.9%	6.1%	7.6%	3.0%

G. Training Baselines Methods on Partial Inputs

For our main experiments, we opt to train the baseline models (PINO, DeepONet, PINNs, FNO) on full observations for several compelling reasons: First, physics-informed models such as PINNs and PINO are unable to effectively compute the PDE loss when only sparse observations are available. Second, other models like DeepONet and FNO perform poorly with sparse observations. For instance, training the DeepONet model on 500 uniformly random points for each training sample in the context of the forward problem of Darcy Flow leads to testing outcomes that are consistently similar, as illustrated in Fig. 10, regardless of the testing input. This pattern suggests that the model tends to generate a generalized solution that minimizes the average error across all potential solutions rather than converging based on specific samples. Furthermore, the partial-input-trained model exhibits poor generalization when faced with a different distribution of observations from training, indicating that it lacks flexibility—a critical attribute of our DiffusionPDE.

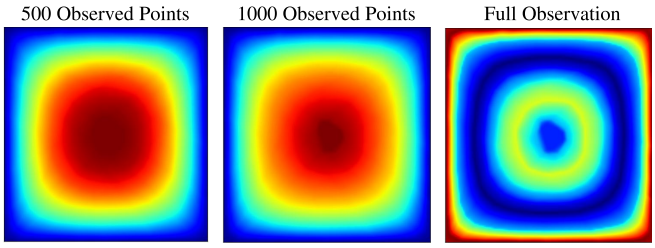


Figure 10. Predicted solutions obtained using the DeepONet model trained with 500 observation points across different numbers of observation points.

H. Standard Deviation of DiffusionPDE Experiment Results

We further assess the statistical significance of our DiffusionPDE by analyzing the standard deviations for forward and inverse problems under conditions of 500 sparse observation points and full observation, respectively, as detailed in Table 5. We evaluate our model using test sets comprising 1,000 samples for each PDE. Our findings confirm that full observation enhances the stability of the results, a predictable outcome as variability diminishes with an increase in observation points. The standard deviations are notably higher for more complex PDEs, such as the inverse problems of the Poisson and Helmholtz equations, reflecting the inherent challenges associated with these computations. Overall, DiffusionPDE demonstrates considerable stability, evidenced by relatively low standard deviations across various tests.

I. Robustness of DiffusionPDE

We find that DiffusionPDE is robust against sparse noisy observation. In Fig. 11, we add Gaussian noise to the 500 observed points of Darcy Flow coefficients. Our DiffusionPDE can maintain a relative error of around 10% with a 15% noise level

Table 5. Standard deviation of DiffusionPDE when solving forward and inverse problems with sparse or full observations regarding different kinds of PDEs.

		Sparse Observations	Full Observations
Darcy Flow	Forward	$2.5 \pm 0.7\%$	$2.2 \pm 0.1\%$
	Inverse	$3.2 \pm 0.9\%$	$2.0 \pm 0.1\%$
Poisson	Forward	$4.5 \pm 0.9\%$	$2.7 \pm 0.1\%$
	Inverse	$20.0 \pm 1.8\%$	$9.8 \pm 0.7\%$
Helmholtz	Forward	$8.8 \pm 1.0\%$	$2.3 \pm 0.1\%$
	Inverse	$22.6 \pm 1.7\%$	$4.0 \pm 0.6\%$
Non-bounded Navier-Stokes	Forward	$6.9 \pm 0.9\%$	$6.1 \pm 0.2\%$
	Inverse	$10.4 \pm 1.0\%$	$8.6 \pm 0.3\%$
Bounded Navier-Stokes	Forward	$3.9 \pm 0.2\%$	$1.7 \pm 0.1\%$
	Inverse	$2.7 \pm 0.2\%$	$1.4 \pm 0.1\%$

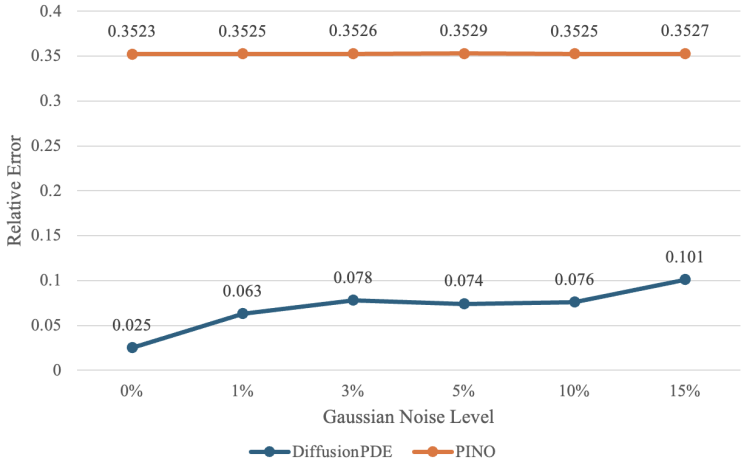


Figure 11. Relative errors of recovered Darcy Flow solutions with sparse noisy observation.

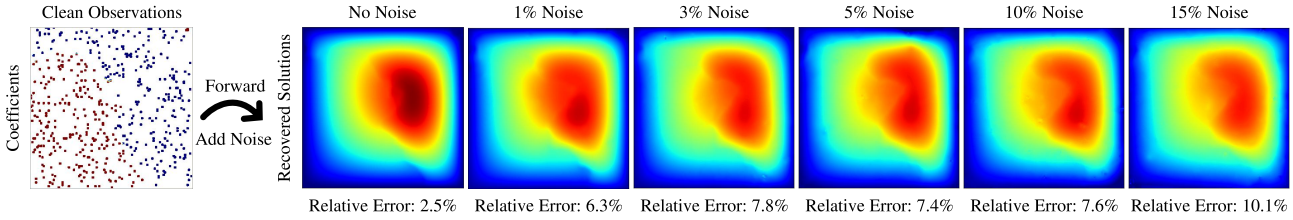


Figure 12. Recovered solutions for Darcy Flow with noisy observations.

concerning the forward problem, and the recovered solutions are shown in Fig. 12. Baseline methods such as PINO also exhibit robustness against random noise under sparse observation conditions; this is attributed to their limited applicability to sparse observation problems, leading them to address the problem in a more randomized manner.

Robustness on Sampling Patterns Moreover, as mentioned in the main document, we investigate the robustness of DiffusionPDE on different sampling patterns of the observation points. Here, we address the forward problem of Darcy Flow using 500 observed coefficient points, which are non-uniformly concentrated on the left and right sides or are regularly distributed across the grid, as depicted in Fig. 13. Our results demonstrate that DiffusionPDE flexibly solves problems with arbitrary sparse observation locations within the spatial domain, without re-training the neural network model.

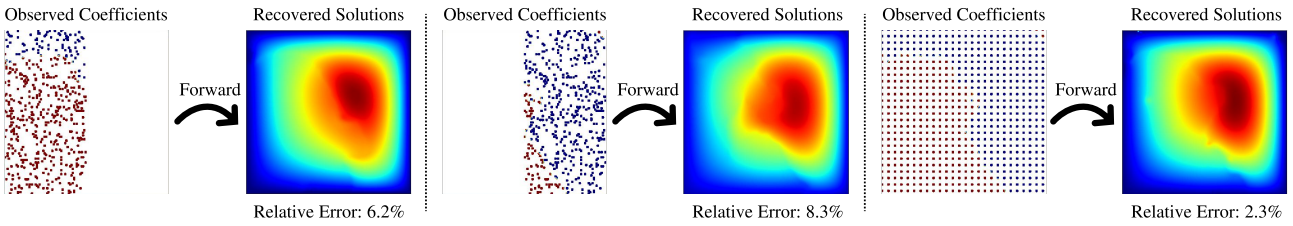
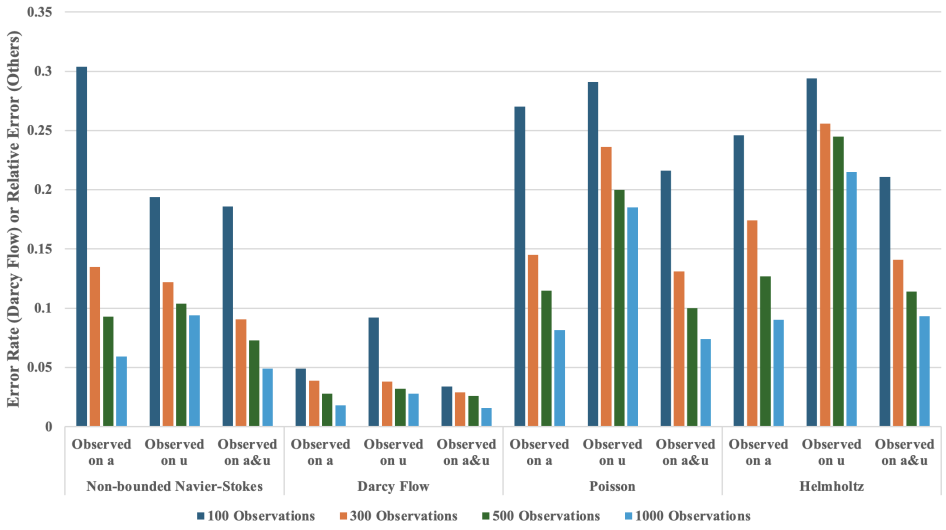


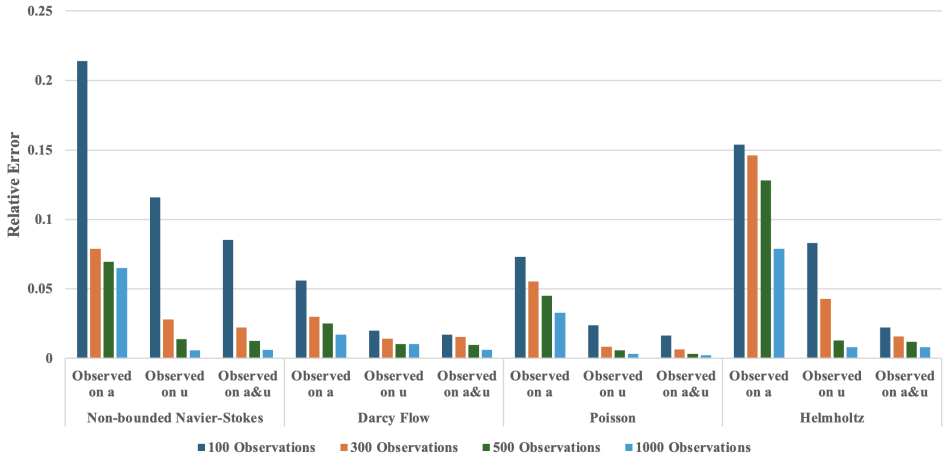
Figure 13. Recovered solutions for Darcy Flow with observations sampled using non-uniform distributions.

J. Solving Forward and Inverse Problems with Different Numbers of Observations

We also investigate how our DiffusionPDE handles varying degrees of sparse observation. Experiments are conducted on the Darcy Flow, Poisson equation, Helmholtz equation, and non-bounded Navier-Stokes equation. We examine the results of DiffusionPDE in solving forward and inverse problems when there are 100, 300, 500, and 1000 random observations on a, u, or both a and u, as shown in Fig. 14.



(a) Error rates for Darcy Flow and relative errors for other PDEs of recovered coefficients or initial states a.



(b) Relative errors of recovered solutions or final states u.

Figure 14. Error rate or relative error of both coefficients (or initial states) a and solutions (or final states) u with different numbers of observations.

We have observed that the error of DiffusionPDE decreases as the number of sparse observations increases. Overall, we recover u better than a. DiffusionPDE can recover u with approximately 2% observation points at any side pretty well. DiffusionPDE is also capable of recovering both a and u with errors $1\% \sim 10\%$ with approximately 6% observation points at any side for most PDE families. We also conclude that our DiffusionPDE becomes insensitive to the number of observations once more than 3% of the points are observed.

K. Solving Forward and Inverse Problems across Varied Resolutions

To evaluate the generalizability of DiffusionPDE, we implemented the model on various resolutions, including 64×64 and 256×256, while maintaining the same percentage of observed points. For resolutions of 64×64, 128×128, and 256×256, we observe 125, 500, and 2000 points on a or u respectively, which are approximately 3% for each resolution. Overall, DiffusionPDE is capable of handling different resolutions effectively. For instance, Table 6 presents the forward relative errors of the solution u and inverse error rates of the coefficient a for the Darcy Flow, demonstrating that DiffusionPDE performs consistently well with similar error rates across various resolutions.

Table 6. Forward relative errors and inverse error rates of Darcy Flow across different resolutions.

Resolution	Forward Relative Error	Inverse Error Rate
64×64	2.9%	4.3%
128×128	2.5%	3.2%
256×256	3.1%	4.1%