Inverse Problems with ML Surrogate Models

Alexander Clarence Berne, Sean Sebastian Darcy, Zijun Deng,
Zongyi Li, Annika Viswesh*

May 2024

1 Introduction

This project will focus on solving inverse problems using machine learning (ML) surrogate models. Inverse problems, which involve deducing unknown parameters or inputs from observed outputs, are common in various scientific and engineering fields. Traditional methods to solve these problems often rely on running complex forward models, which simulate the system to predict outputs given a set of inputs. However, these forward models can be computationally expensive and time-consuming, especially when they need to be run multiple times, as is the case with multi-query methods like Markov Chain Monte Carlo (MCMC) techniques.

The motivation for using ML surrogate models stems from their ability to approximate the behavior of these complex forward models with significantly reduced computational cost. By training a surrogate model on a set of input-output pairs generated from the true forward model, we can create a much faster-running model that can produce predictions with comparable accuracy. This acceleration is particularly advantageous in the context of multi-query methods, where the forward model needs to be evaluated numerous times to explore the parameter space thoroughly.

In MCMC methods, for instance, a large number of forward model evaluations are required to sample from the posterior distribution of the parameters. Each iteration of the MCMC algorithm typically involves proposing a new set of parameters and evaluating the likelihood of the observed data given these parameters using the forward model. When the forward model is computationally intensive, the entire MCMC process can become prohibitively slow. By replacing the forward model with a surrogate model, we can achieve substantial speed-ups, making it feasible to obtain better posterior distribution statistics to solve inverse problems more efficiently.

2 Markov Chain Monte Carlo Approach

We use a standard Markov Chain Monte Carlo algorithm designed for function-valued problem from [2], called pCN. As shown in Algorithm 1, the pCN algorithm is based on Metropolis–Hastings method, where we start with an initial sample $u^{(0)}$ and iterative sample new instances $u^{(k)}$. At each iteration, we either accept or reject the next state $^{(k)}$ based on the ratio of the likelihood of new state $\Phi(v^{(k)})$ and the previous state $\Phi(u^{(k)})$. Specifically, we accept the new instances with probability $\min\{1, \exp(\Phi(u^{(k)}) - \Phi(v^{(k)}))\}$, in this case, $u^{(k+1)} = v^{(k)}$. Otherwise the the previous instance is kept $u^{(k+1)} = u^{(k)}$. The likelihood can be either computed by numerical solver (e.g. RK45 integrator) or machine-learning-based surrogate models. In the following sections, we will compare numerical solvers and ML-based surrogate model for the Lorenz96 system and Darcy flow problems.

*Authors contributed equally. Names are listed in alphabetical order of last names.

Algorithm 1 pCN algorithm

1: Set $k = 0$ and pick $u^{(0)}$.
2: Propose $v^{(k)} = \sqrt{1-\beta^2}u^{(k)} + \beta\xi^{(k)}, \quad \xi^{(k)} \sim \mathcal{N}(0, C)$.
3: Set $u^{(k+1)} = v^{(k)}$ with probability $a(u^{(k)}, v^{(k)}) = \min\{1, \exp(\Phi(u) - \Phi(v))\}$.
4: Set $u^{(k+1)} = u^{(k)}$ otherwise.
5: $k \to k + 1$.

3 Lorenz96

3.1 The Numerical Model

The Lorenz96 model is a dynamical system introduced by Edward Lorenz in 1996, primarily used to study chaotic behavior in the atmosphere. It is a simplified model that captures essential features of atmospheric dynamics and is given by the following set of differential equations:

$$\frac{dx_n}{dt} = (x_{n+1} - x_{n-2})x_{n-1} - x_n + F, \tag{1}$$

for $i = 1, 2, \ldots, N$, where x_i represents the state of the system at the i-th grid point, and F is a constant forcing term. The indices are taken modulo N, meaning that $x_{-1} = x_{N-1}$, $x_0 = x_N$, and $x_{N+1} = x_1$.

The Lorenz96 model is often used with periodic boundary conditions, and its dynamics can be tuned by adjusting the forcing parameter F. For $F \geq 8$, the system exhibits chaotic behavior, making it an excellent testbed for studying predictability and data assimilation in numerical weather prediction. In this project, we will use the Lorenz96 model to study inverse problems. Specifically, we will integrate the Lorenz96 dynamics for 0.2 time units from some initial state and attempt to recover this initial state from a noisy observation of the state at the end of the interval. We will apply both full observations and partial observations (observing every other location) to evaluate the effectiveness of machine learning surrogate models in solving this inverse problem.

3.2 The Surrogate Model

The surrogate model for solving this ODE, which is a combination of finite-size ensemble Kalman filter with a convolutional neural network, is described as follows. Consider the formulation of the following inverse problem:

$$\mathbf{y}_k^{\text{obs}} = \mathcal{H}_k(\mathbf{x}_k) + \epsilon_k^{\text{obs}} \tag{2}$$

where $0 \leq k \leq K$ represents the time index for time t_k, ϵ_k^{obs} represents the observation error, $\mathbf{x}_k \in \mathbb{R}^m$ is a time-stepping representation of an an unknown, continuous process $\mathbf{x}$, and $\mathbf{y}_k^{\text{obs}} \in \mathbb{R}^m$ represents the corresponding multi-dimensional observation vector for $\mathbf{x}_k$ at time t_k. Furthermore, $\mathcal{H}_k : \mathbb{R}^m \to \mathbb{R}^p$ is a known observation operator. We assume that $\epsilon_k^{\text{obs}} \sim \mathcal{N}(0, \mathbf{R}_k)$. We assume i.i.d. noisy distribution and a constant number of observations p.

Define σ^{obs} as the standard deviation with respect to observation error, and $\mathbf{I}_p$ as the $p \times p$ identity matrix. Then, by the assumption that observations are uncorrelated, it is implied that

$$\mathbf{R}_k \equiv (\sigma^{obs})^2\mathbf{I}_p \tag{3}$$

A surrogate model $\mathcal{G}$ was derived to predict the state of process $\mathbf{x}$ at time $k+1$ given the state at time k, with the prediction formulated as

$$\mathbf{x}_{k+1} = \mathcal{G}(\mathbf{x}_k) + \epsilon_k^m \tag{4}$$

where the error of the surrogate model $\mathcal{G}$ is ϵ_k^m. The surrogate model $\mathcal{G}$, in this case, is represented as a convolutional neural network via a parametric function $\mathcal{G}_{\mathbf{W}}(\mathbf{x})$ as follows

$$\mathcal{G}_{\mathbf{W}}(\mathbf{x}_k) := \mathbf{x}_k + f_{nn}(\mathbf{x}_k, \mathbf{W}) \tag{5}$$

where f_{nn} represents a convolutional neural network and $\mathbf{W}$ represents the weights of f_{nn}. The model is trained with loss function defined as follows

$$L(\mathbf{W}) = \sum_{k=0}^{k-N_f-1} \sum_{i=1}^{N_f} \left\| \mathcal{G}_{\mathbf{W}}^{(i)}(\mathbf{x}_k) - \mathbf{x}_{k+i} \right\|_{\mathbf{P}_k^{-1}}^2 \tag{6}$$

where $\mathbf{P}_k^{-1}$ is the surrogate model covariance matrix and is positive semi-definite and symmetric. This matrix allows for each state to receive distinct weights during the optimization of the model. $\mathbf{P}_k^{-1}$ is defined as

$$\|\mathbf{x}\|_{\mathbf{P}_k^{-1}} = \mathbf{x}^T \mathbf{P}_k^{-1} \mathbf{x} \tag{7}$$

In equation (6), N_f is the number of time steps that corresponds to the minimum error between the simulation and target (see Appendix for additional details for the development of the surrogate model).

3.3 Results and Discussion

Figure 1 shows results for the recovery (using the MCMC algorithm 1) of an initial state consisting of 40 parameters that is propagated through 0.2 time units of the Lorenz96 numerical forward model to produce a final observed state. We prescribe noise (sampled from a Gaussian distribution with a standard deviation equal to half (i.e., 0.5) of final parameter values) to observed states. MCMCs use a Metropolis random walk algorithm (refer to section 2) to compute candidate values for each parameter. The likelihood function Φ is defined as a Gaussian L2 norm of the difference between observed and final modelled states. Likelihood function uncertainties equal to the standard deviation of noise assigned to observed states σ^{obs}:

$$\Phi(y) := exp\left(-\frac{||\mathbf{y^{obs}} - \mathbf{y^{mod}}||_2^2}{2(\sigma^{obs})^2} \right) \tag{8}$$

For the case of a fully observed final state (top panel of Figure 1), we find that we are able to closely recover the pattern of ground truth initial states by computing the mean of the posterior distributions of recovered initial parameter values. Recoveries using half-observations (i.e., using every other value for the final 'observed' state) are substantially degraded (i.e., L2 norm = 4.47) relative to recoveries using full observations (i.e., L2 norm = 2.26). Nonetheless, we are able to broadly recover the structure of initial values (i.e., both ground truth and recovered values fall between $\sim$ -2 and 2) in this case.

We repeat our MCMC recovery of a noisy final state using a learned forward model as a surrogate for the Lorenz96 numerical model (see Figure 2). For fully observed final states (top panel), we can recover the pattern of ground truth initial states with only slightly degraded coherence (i.e., L2 norm = 3.16) relative to recoveries using the true numerical model (i.e., L2 norm = 2.26, see top panel of Figure 1). Recoveries using half-observations are also only minimally degraded (i.e., L2 norm = 4.93) relative to similar recoveries using the true numerical model (i.e., L2 norm = 4.47).

4 Darcy Flow

4.1 The Model

The Darcy flow equation is a second-order, linear, elliptic partial differential equation (PDE) that relates the pressure field and the fluid velocity field. It is a fundamental model in hydrogeology and engineering, describing the flow of fluids through porous media, such as groundwater flow, oil reservoir modeling, and environmental engineering. For example, in groundwater flow, the Darcy flow helps to predict the movement of water through aquifers, which is essential for water resource management and contamination assessment. In oil reservoir modeling, it aids in optimizing extraction processes by characterizing the flow of oil, water,

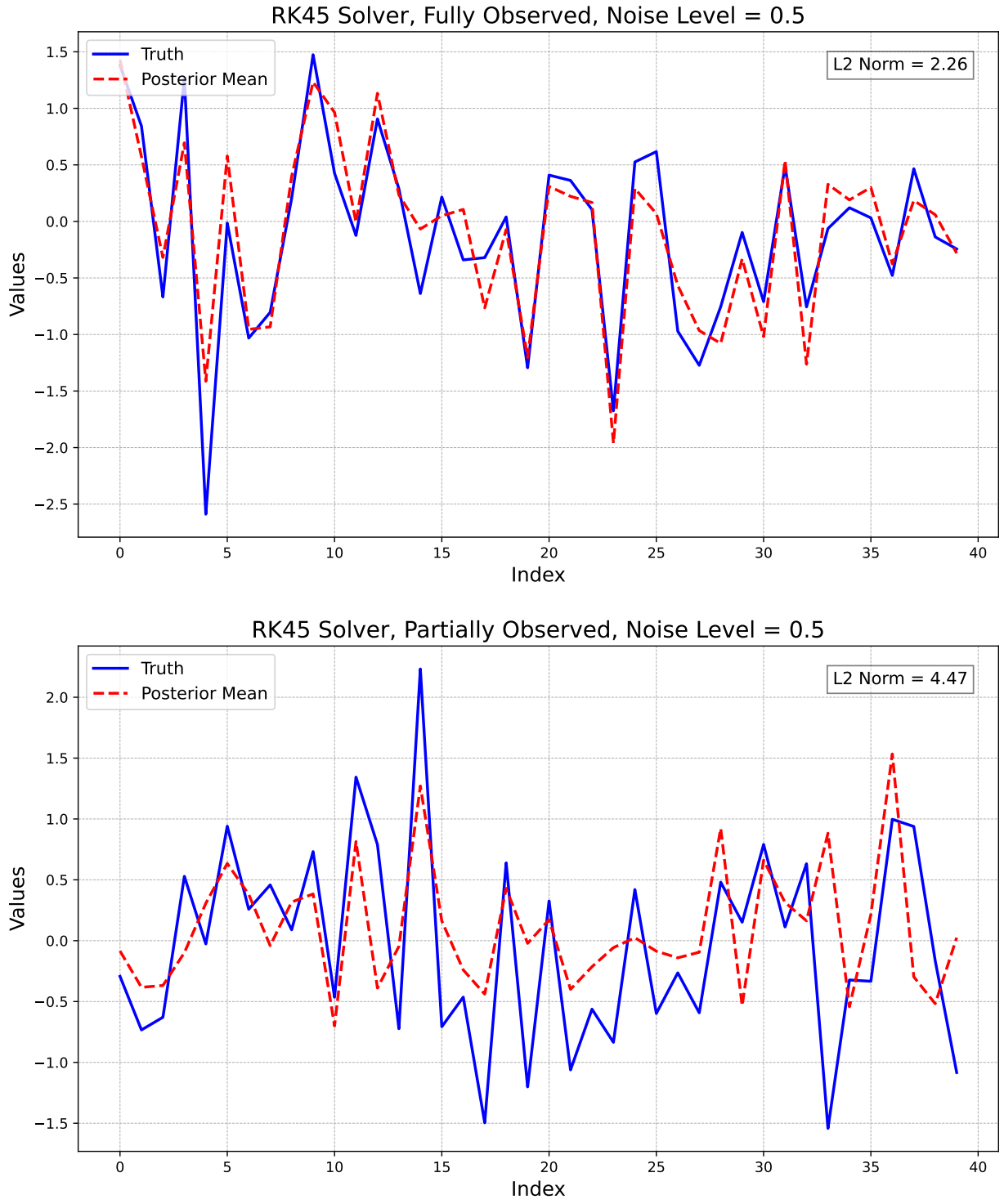

Figure 1: Recovery of 40-dimensional initial states using Lorenz96 forward model using fully observed **(top)** and half-observed **(bottom)** final states. Final states include an added Gaussian noise sampled with standard deviation value equal to 0.5 the standard deviation of final state values). Blue and red lines respectively denote ground truth and posterior mean of inversions of initial states derived from MCMCs. The L2 norm of the difference between recovered and ground truth initial states is shown in the upper right of each plot.

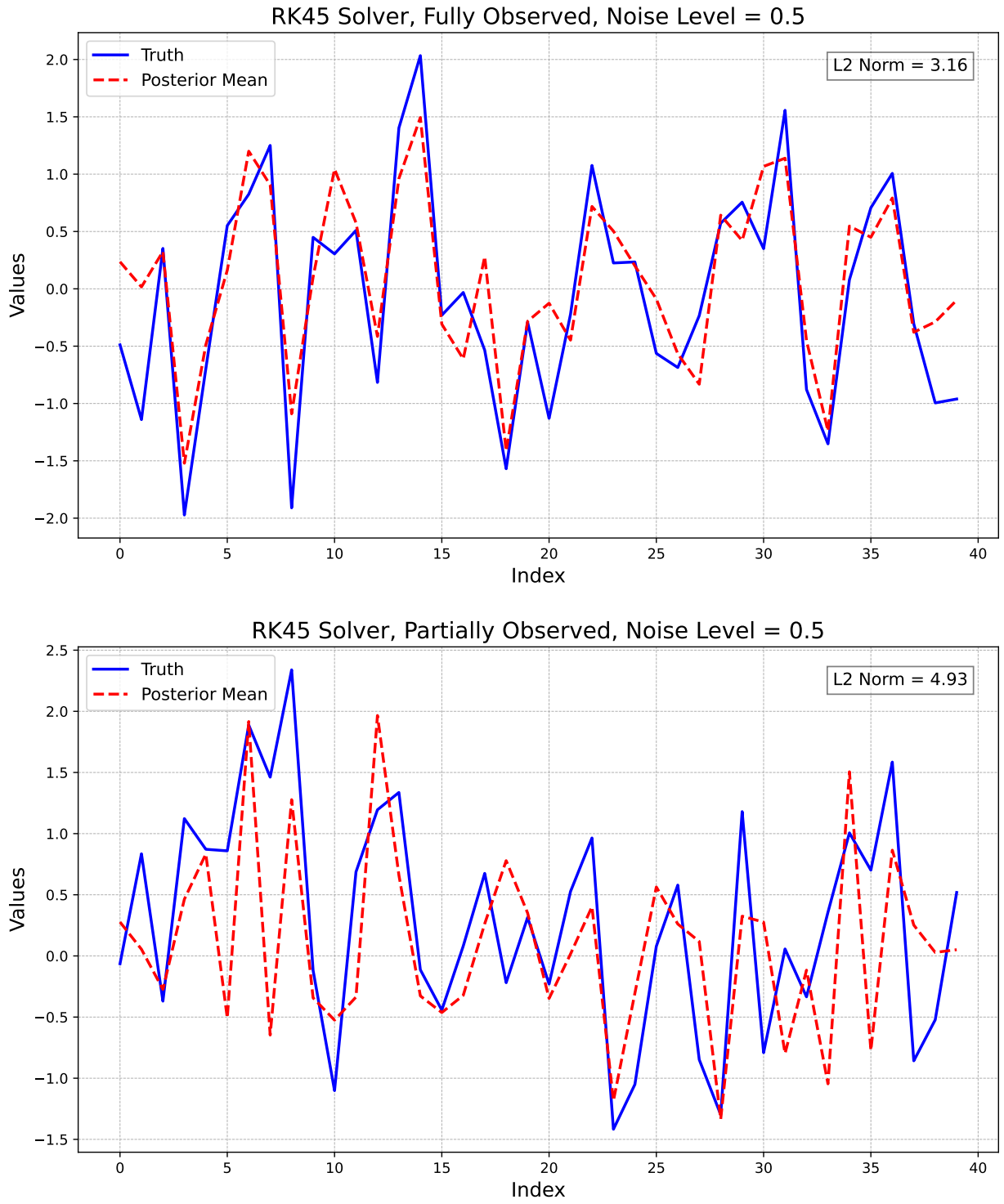

Figure 2: Recovery of 40-dimensional initial state using a learned surrogate model for the forward model with fully observed **(top)** and half-observed **(bottom)** final states. Final states include an added Gaussian noise sampled with standard deviation value equal to 0.5 the standard deviation of final state values). Blue and red lines respectively denote ground truth and posterior mean of inversions of initial states derived from MCMCs. The L2 norm of the difference between recovered and ground truth initial states is shown in the upper right of each plot.

and gas through reservoir rocks. In environmental engineering, the equation is used to model the spread of pollutants through soil and groundwater, providing critical information for remediation efforts. The Darcy We express the Darcy equation as a second order, linear, elliptic PDE, following [3],

$$\begin{aligned} -\nabla \cdot (a(x)\nabla u(x)) &= f(x) \qquad x \in (0,1)^2 \\ u(x) &= 0 \qquad\quad x \in \partial(0,1)^2 \end{aligned} \tag{9}$$

with a Dirichlet boundary where $a \in L^\infty$ is the diffusion coefficient (i.e. permeability), $f \in L^2$ is a fixed forcing function, and $u \in H^1$ is the solution function (i.e. pressure). We are interested in learning the operator mapping the diffusion coefficient to the solution,

$$\begin{aligned} \mathcal{G} : L^\infty &\to H^1 \\ a &\mapsto u \ . \end{aligned} \tag{10}$$

Traditional methods for solving inverse problems associated with Darcy flow can be computationally expensive, as they require numerous evaluations of the forward model. This is where surrogate models, such as the Fourier neural operator (FNO) [3], come in. These machine learning-based models provide a faster alternative by approximating the solutions of the Darcy flow, thereby enabling more efficient and feasible solutions to inverse problems.

4.2 Methods

In this project, we will use the Darcy flow to solve an inverse problem of obtaining the permeability field given the pressure field. Specifically, we will start with a ground truth permeability field and add noise to the resulting pressure field. Using the noisy pressure data, we will apply machine learning surrogate models within an MCMC framework to recover the permeability field. The results will be compared with the ground truth to evaluate the accuracy and efficiency of the surrogate models.

4.2.1 Learning operator

FNO [3] learns a mapping between two infinite-dimensional spaces using a finite collection of observed input-output pairs. Let $D \subset \mathbb{R}^d$ be a bounded, open set, and $\mathcal{A} = \mathcal{A}(D;\mathbb{R}^{d_a})$ and $\mathcal{U} = \mathcal{U}(D;\mathbb{R}^{d_u})$ be separable Banach spaces of functions. Let $G^\dagger : \mathcal{A} \to \mathcal{U}$ be a typically non-linear map to be studied. Denote observations as $\{a_j, u_j\}_{j=1}^N$, where $a_j \sim \mu$ is an i.i.d. sequence from the probability measure μ on $\mathcal{A}$, and $u_j = G^\dagger(a_j)$ is noisy. The goal is to approximate $G^\dagger$ by constructing a parametric map

$$G : \mathcal{A} \times \Theta \to \mathcal{U} \quad \text{or equivalently,} \quad G_\theta : \mathcal{A} \to \mathcal{U}, \quad \theta \in \Theta, \tag{11}$$

where Θ is a finite-dimensional parameter space. $\theta^\dagger \in \Theta$ is chosen such that $G(\cdot, \theta^\dagger) = G_{\theta^\dagger} \approx G^\dagger$.
This framework is natural for learning in infinite-dimensional spaces by defining a cost functional $C : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ and seeking the minimizer of the problem

$$\min_{\theta \in \Theta} \mathbb{E}_{a\sim\mu}[C(G(a,\theta), G^\dagger(a))]$$

The neural operator uses an iterative architecture $v_0 \mapsto v_1 \mapsto \ldots \mapsto v_T$, where each v_j (for $j = 0, 1, \ldots, T-1$) takes values in $\mathbb{R}^{d_u}$. The input $a \in \mathcal{A}$ is initially transformed into a higher-dimensional representation $v_0(x) = P(a(x))$ using a shallow fully-connected neural network P. The output $u(x) = Q(v_T(x))$ is obtained by projecting v_T via the local transformation $Q : \mathbb{R}^{d_v} \to \mathbb{R}^{d_u}$. Each update $v_t \mapsto v_{t+1}$ combines a non-local integral operator $\mathcal{K}$ and a local nonlinear activation function σ.

Definition 1 (Iterative updates): The update to the representation $v_t \mapsto v_{t+1}$ is defined as:

$$v_{t+1}(x) := \sigma\left(W v_t(x) + (\mathcal{K}(a;\phi)v_t)(x)\right), \quad \forall x \in D \tag{12}$$

where $\mathcal{K} : \mathcal{A} \times \Theta_\mathcal{K} \to \mathcal{L}(\mathcal{U}(D;\mathbb{R}^{d_v}), \mathcal{U}(D;\mathbb{R}^{d_v}))$ maps to bounded linear operators on $\mathcal{U}(D;\mathbb{R}^{d_v})$ and is parameterized by $\phi \in \Theta_\mathcal{K}$. $W : \mathbb{R}^{d_v} \to \mathbb{R}^{d_v}$ is a linear transformation, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a component-wise nonlinear activation function. $\mathcal{K}(a;\phi)$ is chosen as a kernel integral transformation parameterized by a neural network.

4.2.2 Fourier Neural Operator

FNO replaces the kernel integral operator by a convolution operator defined in Fourier space. Let $\mathcal{F}$ denote the Fourier transform of a function $f : D \to \mathbb{R}^{d_v}$ and $\mathcal{F}^{-1}$ being its inverse, then for $j = 1, \ldots, d_v$:

$$(\mathcal{F}f)_j(k) = \int_D f_j(x) e^{-2\pi i \langle x,k \rangle} dx, \quad (\mathcal{F}^{-1}f)_j(x) = \int_D f_j(k) e^{2\pi i \langle x,k \rangle} dk, \tag{13}$$

where $i = \sqrt{-1}$. Applying the convolution theorem,

$$(\mathcal{K}(a;\phi)v_t)(x) = \mathcal{F}^{-1}\left(\mathcal{F}(\kappa_\phi) \cdot \mathcal{F}(v_t)\right)(x), \quad \forall x \in D. \tag{14}$$

Therefore, FNO proposed to directly parameterize κ_ϕ in Fourier space.

Definition 2 (Fourier integral operator $\mathcal{K}$) Define the Fourier integral operator

$$(\mathcal{K}(\phi)v_t)(x) = \mathcal{F}^{-1}\left(R_\phi \cdot (\mathcal{F}v_t)\right)(x), \quad \forall x \in D \tag{15}$$

where R_ϕ is the Fourier transform of a periodic function $\kappa : \bar{D} \to \mathbb{R}^{d_v \times d_v}$ parameterized by $\phi \in \Theta_{\mathcal{K}}$.

For frequency mode $k \in D$, we have $(\mathcal{F}v_t)(k) \in \mathbb{C}^{d_v}$ and $R_\phi(k) \in \mathbb{C}^{d_v \times d_v}$. Assuming κ is periodic, we work with discrete modes $k \in \mathbb{Z}^d$ and truncate the Fourier series at $k_{\max} = |Z_{k_{\max}}| = \{k \in \mathbb{Z}^d : |k_j| \leq k_{\max,j}, \text{ for } j = 1, \ldots, d\}$, and thus R_ϕ is parameterized as a complex-valued $(k_{\max} \times d_v \times d_v)$-tensor of truncated Fourier modes, imposing conjugate symmetry for the real-valued κ. The set $Z_{k_{\max}}$ is chosen for efficient implementation.

In implementation, R is a $(s_1 \times \cdots \times s_d \times d_v \times d_v)$-tensor, allowing efficient parallel matrix-vector multiplication.

4.3 Discussion

4.3.1 Gaussian Random Field

The input of the Darcy problem is a function in the L^∞ space. Using a pixel-wise Gaussian distribution would result in pixel-wise noise. As discussed in [3], the input data are sampled from a Gaussian random field $\mathcal{N}(0, (I + 9\Delta)^2)$. We use the same Gaussian random field for MCMC.

Technically, the Gaussian random field $\mathcal{N}(0, (I+9\Delta)^2)$ is implemented by sampling Fourier coefficients and scaling them accordingly with the wavenumber. For periodic problems, we apply the Fast Fourier Transform to convert the Fourier coefficients to a spatial function. For non-periodic problems like Darcy, we use the Discrete Cosine Transform.

4.3.2 Truncation

The Darcy input function is piecewise constant, where 0 represents one type of medium and 1 represents the other. In [3], the sample from the Gaussian random field $\hat{a} \sim \mathcal{N}(0, (I + 9\Delta)^2)$ is truncated such that $a(x) = 1$ where $\hat{a} \geq 0$ and $a(x) = 0$ otherwise. After truncation, the input is no longer Gaussian. However, in MCMC, we can only sample Gaussians. Therefore, we track the posterior distribution without truncation but apply truncation to (1) the ML model and (2) to the final prediction.

Since the provided model is trained on piecewise linear inputs, it is crucial to apply the same truncation in the MCMC algorithm. Thus, whenever we call the model, we truncate the input. Finally, after computing the posterior mean, we apply truncation to obtain the final prediction, which is truncated to zero and one.

4.4 Results

As shown in Figure 3, the MCMC methods with the ML surrogate model can approximately recover the input function. The first row of Figure 3 shows the ground truth input and noisy observation. The

second row shows the prediction of the MCMC algorithm and the corresponding output. Comparing the ground truth input and prediction, the methods recover the overall shape where the top left and bottom right corners are dark ($a = 0$) while the right half is light ($a = 1$). The model also correctly recovers the hole on the left. However, the model does not accurately capture the interface (boundary between the two media). The MCMC prediction has a smoother interface, while the ground truth interface is steeper. Additionally, the prediction did not capture the small dark piece at the top right.

Applying the ML surrogate model to the predicted input, we observe that the corresponding predicted output in Figure 3 (bottom right) is also close to the observed output (top right). Overall, the shape is correct, with two yellow regions at the top left and bottom left, but the boundary of the bottom left region is slightly inaccurate.

5 Conclusions

This project has demonstrated the effectiveness of machine learning (ML) surrogate models for solving inverse problems, specifically within the context of the Lorenz96 and Darcy flow. By leveraging the computational efficiency of ML surrogate models, our approach demonstrates the potential to accelerate the process of obtaining solutions to inverse problems that traditionally require numerous evaluations of computationally expensive forward models.

Through the case studies of the Lorenz96 model, we successfully recovered initial states from noisy observations using both full and partial observation scenarios. The results indicate that ML surrogate models can provide accurate approximations, thereby facilitating the efficient application of Markov Chain Monte Carlo (MCMC) techniques. The comparison between the numerical and ML-based Lorenz96 models also highlighted the potential of ML surrogates to maintain high accuracy for solutions.

For the inverse problem involving the Darcy flow, we demonstrated a capacity to recover permeability fields from noisy pressure field observations. The application of the Darcy ML surrogate likelihood within the MCMC framework yielded posterior distributions that closely matched the ground truth permeability fields. This validates the robustness and reliability of ML surrogate models in complex fluid dynamics applications.

Overall, the findings from this project underscore the potential of ML surrogate models to treat inverse problems across various scientific and engineering domains. By integrating these models into existing frameworks, researchers can achieve faster and more efficient solutions without compromising on accuracy. This approach not only enhances the practicality of multi-query methods. Future work could focus on enhancing training processes, incorporating more sophisticated ML algorithms, and extending the application of learned models to other types of inverse problems.

6 Code

The code used to generate results for this work can be found in the following Github Repository

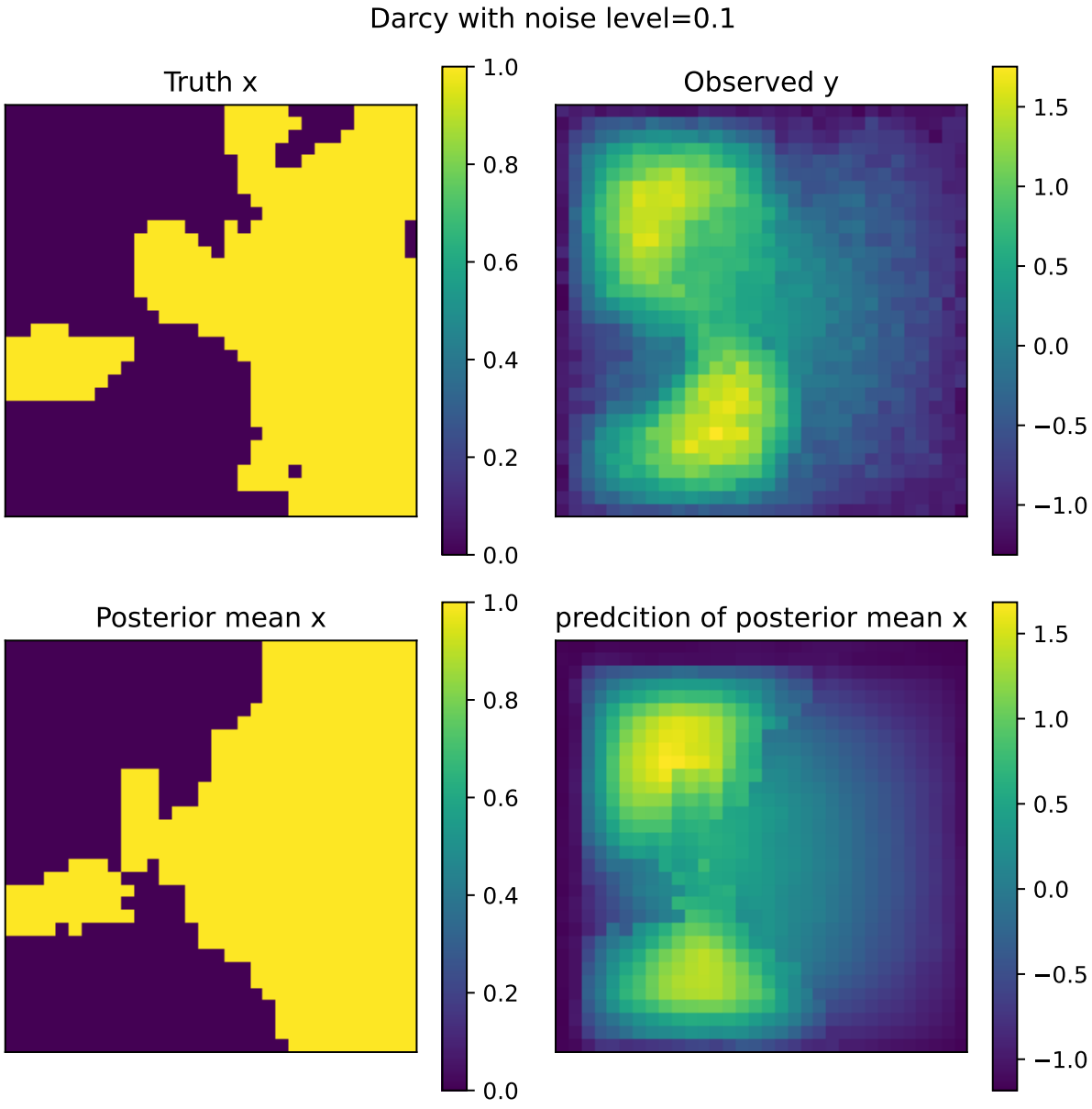

Figure 3: **Top left**: The ground truth input function, which is piecewise constant, with 0 representing one type of medium and 1 representing the other. **Top right**: The observed solution function, with added noise of 0.1 standard deviation. **Bottom left**: The truncated posterior mean of the MCMC algorithm (10000 steps). **Bottom right**: Applying the ML model to the predicted input, recovering the output.

References

[1] Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the lorenz 96 model. *Journal of computational science*, 44:101171, 2020.

[2] Simon L Cotter, Gareth O Roberts, Andrew M Stuart, and David White. Mcmc methods for functions: modifying old algorithms to make them faster. 2013.

[3] 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.

7 Appendix

A data-assimilation approach is taken in combination with the above convolutional neural network to construct the surrogate. In particular, the finite-size ensemble Kalman filter technique (EnKF-N) was used to construct the operator $\mathcal{G}$. The forecast matrix at time t_k is described as follows

$$\boldsymbol{X}_k^f \equiv \left[\boldsymbol{x}_{k,a}^f, \ldots, \boldsymbol{x}_{k,p}^f, \ldots, \boldsymbol{x}_{k,N}^f\right] \in \mathbb{R}^{m\times N} \tag{16}$$

The analysis matrix at time t_k is described as follows

$$\boldsymbol{X}_k^a \equiv \left[\boldsymbol{x}_{k,1}^a, \ldots, \boldsymbol{x}_{k,p}^a, \ldots, \boldsymbol{x}_{k,N}^a\right] \in \mathbb{R}^{m\times N} \tag{17}$$

where each element of the forecast matrix $\boldsymbol{X}_k^f$ is computed in the following manner

$$\boldsymbol{x}_{k,p}^f = \mathcal{G}(\boldsymbol{x}_{k-1,p}^a) + \boldsymbol{\epsilon}_{k,p}^m \tag{18}$$

where $\boldsymbol{\epsilon}_{k,p}^m$ represents the model noise at time t and for member p.
The entire surrogate model combines both the EnKF-N process and the convolutional neural network to solve the ODE represented by the Lorenz-96 model. The surrogate model process is as such. First, a set of weights $\mathbf{W}$ is initialized and updated through a cyclic series of steps. Each cycle consists of first, fixing $\mathbf{W}$ and running the EnKF-N process to obtain $\boldsymbol{x}_{1:K}^m$, using $\boldsymbol{y}^{obs}$, and then fixing $\boldsymbol{x}_{1:K}^m$ from the prior step and running the convolutional neural network to obtain the new series of weights $\boldsymbol{W}$. [1]. These series of steps is repeated until, the weights $\boldsymbol{W}$ have converged.