

# Bayesian multiscale deep generative model for the solution of high-dimensional inverse problems

Yingzhi Xia<sup>a,b,c,d</sup>, Nicholas Zabaras<sup>b,\*</sup>

<sup>a</sup>*School of Information Science and Technology, ShanghaiTech University, Shanghai, China*

<sup>b</sup>*Scientific Computing and Artificial Intelligence (SCAI) Laboratory, University of Notre Dame, 311 Cushing Hall, Notre Dame, IN 46556, USA*

<sup>c</sup>*Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China*

<sup>d</sup>*University of Chinese Academy of Sciences, Beijing, China*

---

## Abstract

Estimation of spatially-varying parameters for computationally expensive forward models governed by partial differential equations is addressed. A novel multiscale Bayesian inference approach is introduced based on deep probabilistic generative models. Such generative models provide a flexible representation by inferring on each scale a low-dimensional latent encoding while allowing hierarchical parameter generation from coarse- to fine-scales. Combining the multiscale generative model with Markov Chain Monte Carlo (MCMC), inference across scales is achieved enabling us to efficiently obtain posterior parameter samples at various scales. The estimation of coarse-scale parameters using a low-dimensional latent embedding captures global and notable parameter features using an inexpensive but inaccurate solver. MCMC sampling of the fine-scale parameters is enabled by utilizing the posterior information in the immediate coarser-scale. In this way, the global features are identified in the coarse-scale with inference of low-dimensional variables and inexpensive forward computation, and the local features are refined and corrected in the fine-scale. The developed method is demonstrated with two types of permeability estimation for flow in heterogeneous media. One is a Gaussian random field (GRF) with uncertain length scales, and the other is channelized permeability with the two regions defined by different GRFs. The ob-

---

\*Corresponding author

*Email addresses:* xiayzh@shanghaitech.edu.cn (Yingzhi Xia), nzabaras@gmail.com (Nicholas Zabaras)

*URL:* <https://www.zabaras.com/> (Nicholas Zabaras)

tained results indicate that the method allows high-dimensional parameter estimation while exhibiting stability, efficiency and accuracy.

*Keywords:* Bayesian Inference, Inverse Problems, Deep Generative Model, High-dimensionality, Multiscale Estimation, Markov Chain Monte Carlo

---

## 1. Introduction

Inverse problems are important but challenging in many fields like geophysics, medical imaging, groundwater flows, and other. They address the estimation of model parameters from partial and noisy observations [1]. Two approaches for addressing inverse problems are typically employed. The deterministic methods convert parameter identification to an optimization problem that involves minimizing the misfit between model predictions and observations. Since limited observations are insufficient to identify the underlying parameters, regularization methods [2, 3, 4] are used to address this ill-posed problem. On the other hand, Bayesian inference approaches play a fundamental role in inverse problems allowing us to quantify the uncertainty of the solution and providing natural regularization via prior knowledge [5]. They treat parameters as random variables to highlight the uncertainty in their estimation. The non-uniqueness of the solution is addressed by computing the posterior of the parameters rather than a single point estimate. Variational inference (VI) and Monte Carlo (MC) methods are two main approximation methods to deal with the computation of the intractable posterior distribution. VI [6, 7, 8, 9] is easy to implement but limited by the family of variational distributions. Most Bayesian approaches emphasize Markov Chain Monte Carlo (MCMC) methods that aim to generate samples from the posterior distribution that subsequently are used to produce statistics of the quantities of interest.

MCMC methods have the appealing property that they are asymptotically exact. Thus, many previous works have studied the MCMC method or its variants for Bayesian inverse problems (BIPs). However, there are two main difficulties for these methods. First, the dimensionality of the spatially-varying parameters can be high (e.g. equal to the number of grid points) leading to the so called curse of dimensionality. Second, these sampling-based approaches require multiple evaluations of the forward model (likelihood evaluation). Each evaluation involves a full forward simulation, which is computationally prohibiting for many

practical problems governed by partial differential equations (PDEs).

For the first problem identified above, given prior information, parameterization methods are often used to provide a low-dimensional embedding of the unknown spatially-varying parameter. The common method in BIPs is the truncated Karhunen-Loève expansion (KLE) for the estimation of Gaussian random fields (GRFs) [10, 11], where inference is performed over a small number of expansion coefficients. With limitations and strong assumptions on the mean and covariance functions, the KLE cannot reflect the true prior information, and is not a good choice for fields with nontrivial correlation structure. To address this, sparse grid interpolation [12, 13] and wavelet-based [14] methods have been proposed. However, such methods still have difficulties in the parameterization of complex parameters such as multi-modal or non-Gaussian random fields [15, 16].

Deep generative models (DGM) [17, 18, 19] provide a good choice for parameterization. DGMs are much more flexible and scalable, where the prior information is naturally incorporated into the training data without strong assumptions. Once the DGM is trained, one can sample latent variables from a low-dimensional simple distribution (like a Gaussian), and then generate the spatially-varying parameter using the pre-trained neural network. Many recent studies integrated the DGM-based parameterization method with various inference methods to tackle non-Gaussian parameter estimation problems, including conditional invertible neural networks [20], variational autoencoder (VAE) with MCMC or ensemble smoother [15, 21], generative adversarial network (GAN) with MCMC or Metropolis-adjusted Langevin algorithm (MALA) [22, 23], adversarial autoencoder (AAE) with iterative local updating ensemble smoother (ILUES) [16] and so on.

A potential remedy of the requirement of MCMC methods for multiple calls to the forward model solver is to build a surrogate forward model, such as polynomial chaos [24, 25], Gaussian process [26, 27], or deep neural networks [28, 29, 30]. However, the surrogate model often introduces epistemic uncertainty that will result in broadening of the posterior for parameter estimation [31]. Furthermore, it is still a difficult task to construct an accurate surrogate for forward models with high-dimensional input using limited data. To reduce the computational burden of the simulation, multiscale [32, 33, 34] and multi-fidelity methods [35, 36] have been applied to accelerate the Bayesian computation without sacri-

ficating accuracy. The two-stage MCMC [37] designed a preconditioned Metropolis-Hastings algorithm to improve the acceptance rate in the fine-scale model. Inspired by the multilevel Monte Carlo, Multilevel MCMC methods [38, 39, 40, 41] are proposed for BIPs to accelerate the estimation of the posterior distribution. All these methods are indeed promising for BIPs by leveraging the advantages of the accurate fine-scale model and the efficiency of the coarse-scale model.

In this work, we propose a multiscale deep generative model (MDGM) exploiting the multiscale nature of the parameter of interest. This extends existing DGMs and allows us to generate parameters on various scales with different discretization/resolution. Since GANs are notorious on training stability and mode collapse, and flow-based models [42, 43] require an identical-dimensional latent space to the parameter, we derive the MDGM based on VAE. In the MDGM, we design a specific latent space that includes two latent variables, a low-dimensional latent variable that controls global and salient features and a higher-dimensional latent variable that defines local and detailed features. Utilizing the hierarchical representation of the parameter and latent spaces, the multiscale inference is performed in the low-dimensional latent space rather than the original parameter space. This allows us to explore the posterior of the parameter from coarse- to fine-scales with a significant computational saving. Once most of the salient features are identified in the coarse-scale using a computationally inexpensive coarse-solver, the fine-scale estimation requires only few fine-scale simulations to refine the coarse-scale parameter estimation.

The main contributions of this work are summarized as follows. (1) Based on the vanilla VAE, we extend and derive the MDGM, which can generate spatial parameters at various scales with an appropriately designed latent space. (2) The proposed multiscale inference method performs efficiently inference across scales based on the MDGM. (3) A flexible scheme allows efficient estimation of rough/global parameter features with coarse-scale inference and parameter refinement with fine-scale inference. (4) The proposed method is demonstrated in Gaussian and non-Gaussian inversion tasks.

The rest of the paper is organized as follows. Section 2 provides the definition of the inverse problem and addresses the limitations of standard Bayesian approaches for distributed parameter estimation. Section 3.1 introduces the generation of the multiscale training

datasets. The big picture of the multiscale estimation problem using hierarchical generative models is addressed in Section 3.2. The one-scale and multiscale generative models are derived in Sections 3.3 and 3.4, respectively. The Bayesian inversion using the multiscale generative model is discussed in Sections 3.5 and 3.6. Section 4 presents the results of various numerical examples in the estimation of Gaussian and channelized permeability in porous media flows and Section 5 summarizes this work.

## 2. Problem Definition

### 2.1. Bayesian inverse problems

In this section, we introduce the inverse problems of interest and briefly discuss the limitations of standard Bayesian inference approaches to inverse problems. We consider a spatially-varying parameter  $\mathbf{x}(\mathbf{s})$  usually represented as a random field  $\mathbf{x}(\mathbf{s}, \omega)$ , where  $\mathbf{s}$  is spatial location in the domain  $\mathcal{S}$  and  $\omega$  is a random event in the sample space  $\Omega$ . This random field is discretized by a random vector  $\mathbf{x} \in \mathbb{R}^M$  using standard finite element or finite difference discretization approaches. In our inverse problem setting,  $\mathbf{x}(\mathbf{s})$ , will be considered as our primary quantity of interest.

Let us consider a physical system governed by PDEs in a given spatial domain. We assume  $\mathbf{x}(\mathbf{s}, \omega)$  to be an input parameter (e.g. material property) of this model. Of interest to this work are distributed properties with multiscale features. The forward model concerning this physical system is usually considered as a function  $\mathcal{F} : \mathbb{R}^M \rightarrow \mathbb{R}^D$ , which maps the unknown parameters  $\mathbf{x}$  to the observable output  $\mathcal{D}_{obs} \in \mathbb{R}^D$  with a measurement noise  $\boldsymbol{\xi} \in \mathbb{R}^D$ :

$$\mathcal{D}_{obs} = \mathcal{F}(\mathbf{x}) + \boldsymbol{\xi}. \quad (1)$$

The inverse problem is to infer the unknown parameters  $\mathbf{x}$  based on these noisy data  $\mathcal{D}_{obs}$ . In the particular problem we will focus in Section 4, our goal is to estimate the permeability field in a porous media flow using pressure measurements.

Without prior information about the measurement system and/or model evaluation, we assume that  $\boldsymbol{\xi}$  is a zero-mean Gaussian noise with covariance matrix  $\boldsymbol{\Sigma}$ , i.e.,  $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ . Since often  $dim(\mathbb{R}^D) \ll dim(\mathbb{R}^M)$ , the inverse problem is highly ill-posed and identification

of the parameter  $\mathbf{x}$  is highly-sensitive to this noise. The Bayesian paradigm [5] provides a general and natural way to treat the unknown parameter  $\mathbf{x}$  as random variable to highlight the uncertainty in the inference process. Given the observation data  $\mathcal{D}_{obs}$ , one calculates the posterior probability  $\pi(\mathbf{x}|\mathcal{D}_{obs})$  via Bayes' formula as follows:

$$\pi(\mathbf{x}|\mathcal{D}_{obs}) = \frac{\mathcal{L}_e(\mathcal{D}_{obs}|\mathbf{x})\pi(\mathbf{x})}{\int \mathcal{L}_e(\mathcal{D}_{obs}|\mathbf{x})\pi(\mathbf{x})d\mathbf{x}}, \quad (2)$$

where  $\pi(\mathbf{x})$  is the prior distribution, and  $\mathcal{L}_e(\mathcal{D}_{obs}|\mathbf{x})$  is the likelihood function which evaluates the discrepancy between the forward predictions and observations. For the assumed case of Gaussian noise, we can define the likelihood function as

$$\mathcal{L}_e(\mathcal{D}_{obs}|\mathbf{x}) \propto \exp\left(-\frac{1}{2}(\mathcal{D}_{obs} - \mathcal{F}(\mathbf{x}))^T \Sigma^{-1}(\mathcal{D}_{obs} - \mathcal{F}(\mathbf{x}))\right). \quad (3)$$

As the parameter  $\mathbf{x}$  of interest is high-dimensional, the normalization constant in Eq. (2) involves computing a high-dimensional integral that is often an intractable process. Thus approximate inference for the posterior  $\pi(\mathbf{x}|\mathcal{D}_{obs})$  is performed using the unnormalized density, i.e.,

$$\pi(\mathbf{x}|\mathcal{D}_{obs}) \propto \mathcal{L}_e(\mathcal{D}_{obs}|\mathbf{x})\pi(\mathbf{x}). \quad (4)$$

## 2.2. Multiscale inference with MDGM

Without a closed-form expression, the posterior distribution in Eq. (4) must be computed numerically. To this end, MCMC [10, 44] or other approximation methods like Ensemble Kalman filter (EnKF) [16, 30] are often employed. However, there are still two main difficulties for these methods. MCMC and EnKF implementations will often fail to directly approximate the posterior of the high-dimensional spatially-varying parameter  $\mathbf{x}$ . Moreover, for complex parameters (e.g. channelized permeability), the prior information cannot be easily cast as an explicit probability distribution. However, one often has access to a historical dataset  $\mathbf{X} \equiv \{\mathbf{x}^{(i)}\}_{i=1}^N$  [16, 45, 46] where  $\mathbf{x}^{(i)}$  can be seen as samples from the underlying prior distribution  $\pi(\mathbf{x})$ . One could use  $\mathbf{X}$  to approximate  $\pi(\mathbf{x})$  with its empirical measure. However, in this work, we will use this dataset to approximate the prior distribution with a

generative model as follows:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}, \quad (5)$$

where  $p(\mathbf{z})$  is a simple distribution (e.g. Gaussian) for the latent variable  $\mathbf{z} \in \mathbb{R}^d$ , and  $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$  is a generative model parameterized by  $\boldsymbol{\theta}$  (decoder). In a DGM like a VAE, one can choose a Gaussian distribution  $\mathcal{N}(\mathbf{0}, \mathbf{I})$  for  $p(\mathbf{z})$ , and  $\mathcal{N}(\mu_{\boldsymbol{\theta}}(\mathbf{z}), \sigma^2\mathbf{I})$  for  $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ , where  $\mu_{\boldsymbol{\theta}}(\mathbf{z})$  is the output of the decoder neural network, and  $\sigma$  is a hyperparameter that does not depend on the latent variable  $\mathbf{z}$ . It is common practice in the literature [15, 16, 47, 48] to ignore the noise and approximate the density  $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$  with the point estimate  $\mu_{\boldsymbol{\theta}}(\mathbf{z})$ . Once the model is trained, we can thus define a mapping from the latent space to the original parameter space, i.e.  $\mathbf{x} = \mu_{\boldsymbol{\theta}}(\mathbf{z})$ .

The parameters  $\boldsymbol{\theta}$  of the generative model  $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$  can be computed using the given training dataset  $\mathbf{X}$  by minimizing the Kullback–Leibler (KL) divergence  $D_{KL}(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta}))$  [49], where  $\mathbf{x}^{(i)} \stackrel{i.i.d}{\sim} \pi(\mathbf{x})$ . This leads to the equivalent problem of maximizing the marginal log-likelihood:

$$\begin{aligned} \log p(\mathbf{X}|\boldsymbol{\theta}) &= \sum_{i=1}^N \log p(\mathbf{x}^{(i)}|\boldsymbol{\theta}) \\ &= \sum_{i=1}^N \log \int p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})p(\mathbf{z}^{(i)})d\mathbf{z}^{(i)}. \end{aligned} \quad (6)$$

The above marginalization is potentially very difficult to compute involving an intractable integration. Using Expectation-Maximization is also intractable as that will require the posterior  $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$  that is also computationally intractable. The marginal likelihood for the dataset  $\{\mathbf{x}^{(i)}\}_{i=1}^N$  can be reformulated using a variational density  $q_{\phi}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$  (encoder) parameterized by  $\phi$ . The details of these calculations will be given in Section 3.3.

To approximate the posterior distribution in Eq. (4) using the MCMC method, we are interested to generate realizations sampled from the underlying prior distribution  $\pi(\mathbf{x})$ . To sample realizations from  $\pi(\mathbf{x})$  using the generative model, one can sample  $\mathbf{z}_i$  from the simple and low-dimensional distribution  $p(\mathbf{z})$  and then obtain the realization  $\mathbf{x}^{(i)}$  using the decoder model  $\mu_{\boldsymbol{\theta}}(\mathbf{z})$ . Alternatively, instead of approximating the posterior  $\pi(\mathbf{x}|\mathcal{D}_{obs})$  in Eq. (4), one can instead evaluate the low-dimensional posterior  $p(\mathbf{z}|\mathcal{D}_{obs})$  using the following

unnormalized density:

$$p(\mathbf{z}|\mathcal{D}_{obs}) \propto \mathcal{L}_e(\mathcal{D}_{obs}|\mathbf{z})p(\mathbf{z}), \quad (7)$$

where  $p(\mathbf{z})$  is an explicit distribution, and the likelihood can be evaluated using the decoder model  $\mu_{\theta}(\mathbf{z})$  and the forward model  $\mathcal{F}(\mathbf{x})$ . The evaluation of the posterior of the low-dimensional latent variable  $\mathbf{z}$  using MCMC or EnKF is computationally tractable [15, 16].

To further improve the efficiency of the inference process, we will introduce a multiscale version of the above highlighted generative model to perform inference in each scale  $l = 1, 2, \dots, L$  from the coarsest-scale ( $l = 1$ ) to the desired finest-scale ( $l = L$ ). This multiscale scheme based on the MDGM contains a hierarchical simple distribution  $p(\mathbf{z}_l)$  at each scale  $l$  and a conditional distribution  $p_{\theta_l}(\mathbf{x}_l|\mathbf{z}_l)$  that can generate the spatially-varying parameter  $\mathbf{x}$  in each scale. Correspondingly, one can assess the posterior  $p(\mathbf{z}_l|\mathcal{D}_{obs})$  using MCMC with  $p(\mathbf{z}_l)$ ,  $\mu_{\theta_l}(\mathbf{z}_l)$ , and  $\mathcal{F}_l(\mathbf{x}_l)$ . The details of this multiscale model are given next.

### 3. Methodology

#### 3.1. Multiscale dataset

Our physical systems of interest are governed by a system of PDEs, and the spatially-varying property of interest is a material property appearing e.g. in the constitutive equations. The forward problem defines the well-posed solution of the PDEs (with some boundary conditions) given appropriate material properties. Such problems are often solved in a discretized fashion with finite element or finite difference or spectral approximations for different levels of discretization of the spatial domain  $\mathcal{S}$ . In this work, we are interested in a hierarchical parameterization of the spatially-varying parameters  $\mathbf{x}_l$  with different spatial discretization or resolutions at each scale  $l$ . If the forward model is performed in the 2-D space, the parameter random fields  $\mathbf{x}_l$  at the  $l$ -th scale are treated as images, e.g.  $\mathbf{x}_l \in \mathbb{R}^{H_l \times W_l}$  ( $M_l = H_l \times W_l$ ), where  $H_l, W_l$  denote the number of the pixels in the horizontal and vertical directions, respectively.

For notational convenience, we assume that the finest scale parameters  $\mathbf{x}_L$  represent our “true parameter model”. The noisy observations  $\mathcal{D}_{obs}$  in our numerical studies are taken from

this discretization level. In the inverse problem of interest, our task is to compute  $\mathbf{x}_L$  given a finite number of observations. For the solution of this inverse problem, prior knowledge can provide useful information for  $\mathbf{x}_L$  before any observations. As prior information for our model, we assume that we are given a dataset  $\mathbf{X} = \{\mathbf{x}_L^{(i)}\}_{i=1}^N$ . To obtain images for training the generative model at different discretization levels, we will need to obtain a multiscale training dataset.

This can be accomplished by upscaling the fine-scale training dataset [50, 51, 52]. For example, with an upscaling (deterministic) operator  $\mathcal{U} : \mathbb{R}^{M_l} \rightarrow \mathbb{R}^{M_{l-1}}$  ( $M_{l-1} \ll M_l$ ), where  $M_l$  and  $M_{l-1}$  are the dimensions of  $\mathbf{x}_l$  and  $\mathbf{x}_{l-1}$ , respectively. The dataset  $\{\mathbf{x}_{l-1}^{(i)}\}_{i=1}^N$  in the coarse-scale ( $l-1$ ) is obtained by

$$\mathbf{x}_{l-1}^{(i)} = \mathcal{U}(\mathbf{x}_l^{(i)}). \quad (8)$$

The datasets  $\{\mathbf{x}_l^{(i)}\}_{i=1}^N$ ,  $l = 1, 2, \dots, L$  in different scales are obtained by adopting recursively  $\mathcal{U}$  in Eq. (8) starting with the finest-scale  $l = L$ . we assume  $\mathbf{x}_l^{(i)}$  is sampled from  $l$ -th scale underlying prior distribution  $\pi_l(\mathbf{x}_l)$ . The operator  $\mathcal{U}$  used in this paper is deterministic, which leads to a one-to-one correspondence between the elements in  $\{\mathbf{x}_1^{(i)}, \dots, \mathbf{x}_{L-1}^{(i)}, \mathbf{x}_L^{(i)}\}_{i=1}^N$ . One can choose different operators  $\mathcal{U}$  depending on the particular parameter of interest. In this paper, we employ the arithmetic average<sup>1</sup>:

$$\mathbf{x}_{l-1}(e) = \frac{1}{n_e} \sum_{i=1}^{n_e} \mathbf{x}_l(e_i), \quad (9)$$

where  $n_e$  denotes the number of elements in the fine-scale  $l$  corresponding to one element in the coarse-scale ( $l-1$ ). The value at the coarse-grid element  $e$  is the mean of the values in the spatially corresponding elements  $e_i$  in the fine-scale. Spatial correspondence between two adjacent scales with 50% coarsening in each direction is illustrated in Fig. 1.

An example illustrating this deterministic upscaling for channelized permeability using Eq. (9) is given in Fig. 2. The coarse-scale image provides a blurry representation of the fine-scale image but overall its features are consistent with those of the fine-scale image. It can be noticed that the coarse-scale image manifests itself with a checkerboard pattern that

---

<sup>1</sup><http://www.epgeology.com/static-modeling-f39/how-upscale-permeability-t6045.html>

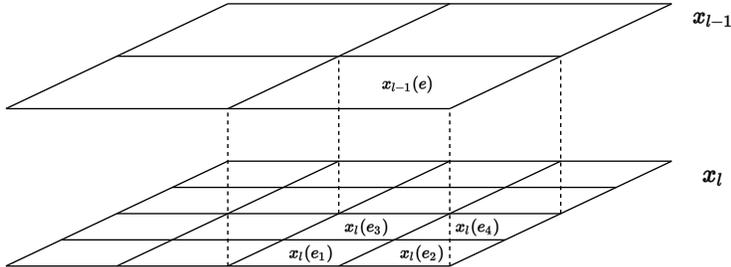


Figure 1: Illustration of the spatial correspondence in deterministic upscaling [32]. The parameter  $\mathbf{x}_{l-1}(e)$  in the coarse-scale element  $e$  is equal to  $\mathcal{U}(\mathbf{x}_l(e_i))$ , where  $e_i$  are the spatially corresponding fine-scale elements to the coarse-element  $e$ . The number of fine- to coarse-elements in each direction is proportional to  $\frac{h_{l-1}}{h_l}$ , where  $h_{l-1}$ ,  $h_l$  are the mesh sizes in the coarse- and fine-scales, respectively.

misses a lots of local information.

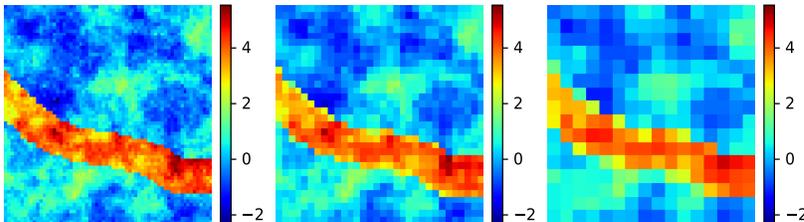


Figure 2: Upscaling channelized log-permeability samples (3 scales) using the upscaling technique in Fig. 1 and Eq. (9). From left to right: (a) original and the finest-grid  $\mathbf{x}_3 \in \mathbb{R}^{64 \times 64}$  realization with resolution  $64 \times 64$ , (b) the coarser-scale  $\mathbf{x}_2$  with resolution  $32 \times 32$  after applying the operator in Eq. (9) with  $n_e = 4$  on  $\mathbf{x}_3$ , (c) the coarsest-scale  $\mathbf{x}_1$  with resolution  $16 \times 16$  after applying the operator in Eq. (9) with  $n_e = 4$  on  $\mathbf{x}_2$ .

With the scales of interest pre-determined and the training dataset defined at each scale, we are ready to train the MDGM and perform inference on each scale using the proposed hierarchical multiscale framework.

### 3.2. Model specification

Given the training dataset  $\{\mathbf{x}^{(i)}\}_{i=1}^N$ , we seek to learn a DGM that can approximate  $\pi(\mathbf{x})$  with Eq. (5). The DGM involves the original parameter space and the latent space and the mappings between these spaces. For any  $\mathbf{x}$  sampled from the underlying distribution  $\pi(\mathbf{x})$ ,

the corresponding  $\mathbf{z}$  is sampled from the conditional distribution  $q_\phi(\mathbf{z}|\mathbf{x})$ , where  $q_\phi(\mathbf{z}|\mathbf{x})$  is called the recognition model or probabilistic encoder model, and  $\phi$  are its model parameters. In the reverse direction, one can sample a  $\mathbf{z}$  from a simple and low-dimensional distribution  $p(\mathbf{z})$ , and obtain the corresponding  $\mathbf{x}$  from the generative model or probabilistic decoder model  $p_\theta(\mathbf{x}|\mathbf{z})$ .

**Definition 3.1** (Probabilistic Generative Model). *Given a set of training input data  $\{\mathbf{x}^{(i)}\}_{i=1}^N$ , where  $\mathbf{x}^{(i)} \sim \pi(\mathbf{x})$ , select an appropriate distribution  $p(\mathbf{z})$  for the latent variable  $\mathbf{z}$  and learn the models  $p_\theta(\mathbf{x}|\mathbf{z})$  and  $q_\phi(\mathbf{z}|\mathbf{x})$ , respectively, such that  $\pi(\mathbf{x})$  can be approximated using Eq. (5), where  $\theta$  and  $\phi$  denote the parameters of the generative and recognition models, respectively.*

For high-dimensional inversion tasks, direct inference in the fine-scale is prohibited due to the computational cost of the forward model. In addition, inference of the latent parameters  $\mathbf{z}$  that lead to good estimates of  $\mathbf{x}$  through a generative model requires a high-dimensional  $\mathbf{z}$ . This will lead to long exploration costs for MCMC and requires a large number of forward model evaluations. To this end, given the multiscale dataset as discussed in Section 3.1, we propose a multiscale scheme for posterior estimation by introducing a hierarchy of generative models from coarse- to fine-scales. In this scheme, the coarse-scale generative models have low-dimensional latent spaces. This together with inexpensive forward model evaluations in the coarse-scales would allow MCMC to explore the posterior in coarse-scales with much reduced cost. The computational savings can be even higher if the latent representation on a given scale utilizes the latent information that was inferred in the immediately coarser-scale.

In our construct, the latent variables  $\mathbf{z}_l$  at level  $l$  of the hierarchy are given as  $\mathbf{z}_l = (\mathbf{z}_{l-1}, \mathbf{z}_l^*)$ , where the latent variables  $\mathbf{z}_{l-1}$  and  $\mathbf{z}_l^*$  are encoded from the coarse-scale parameter  $\mathbf{x}_{l-1}$  and the fine-scale parameter  $\mathbf{x}_l$ , respectively. The latent variable  $\mathbf{z}_{l-1}$  can generate  $\mathbf{x}_{l-1}$  through the generative model at scale  $(l-1)$ . It also impacts the generation of  $\mathbf{x}_l$  in the fine-scale  $l$  by way of dominating its salient features since  $\mathbf{z}_{l-1}$  captured the information from the immediately coarser-scale. In this setting with  $\mathbf{z}_{l-1}$  encoded from  $\mathbf{x}_{l-1}$ , it is anticipated that  $\mathbf{x}_l$  generated by  $p_{\theta_l}(\mathbf{x}_l|\mathbf{z}_{l-1}, \mathbf{z}_l^*)$  would sustain most of the features of  $\mathbf{x}_{l-1}$ . We extend the Definition 3.1 to a multiscale scenario as follows.

**Definition 3.2** (Multiscale Deep Generative Model). *Given a set of training input data  $\{\mathbf{x}_l^{(i)}\}_{i=1}^N, l = 1, 2, \dots, L$ , select appropriate distributions  $p(\mathbf{z}_l)$ , where  $\mathbf{z}_l = (\mathbf{z}_{l-1}, \mathbf{z}_l^*)$  are the latent variables at scale  $l$ , and learn the models  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1), p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*), \dots, p_{\theta_l}(\mathbf{x}_l|\mathbf{z}_{l-1}, \mathbf{z}_l^*)$  and  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1), q_{\phi_2}(\mathbf{z}_1, \mathbf{z}_2^*|\mathbf{x}_2), \dots, q_{\phi_l}(\mathbf{z}_{l-1}, \mathbf{z}_l^*|\mathbf{x}_l)$  recursively, such that  $\pi_l(\mathbf{x}_l)$  can be approximated using Eq. (5) in each scale. Here,  $\theta_l$  and  $\phi_l$  denote the parameters of the generative and recognition models at scale  $l$ , respectively.*

Once the MDGM is established, the parameters  $\mathbf{x}_l$  are encoded by the latent variable  $\mathbf{z}_l$  so that one can perform inference of  $\mathbf{z}_l$ . Given the prior distribution  $p(\mathbf{z}_l)$ , the decoder model  $\mu_{\theta}(\mathbf{z}_l)$ , the forward model  $\mathcal{F}_l$ , and observations  $\mathcal{D}_{obs}$ , inference of the posterior of  $\mathbf{z}_l$  is performed as follows:

$$p(\mathbf{z}_l|\mathcal{D}_{obs}) \propto \mathcal{L}_e(\mathcal{D}_{obs}|\mathbf{z}_l)p(\mathbf{z}_l). \quad (10)$$

Note that MCMC converges and captures prominent and valuable features quickly in the coarse-scale  $(l - 1)$  since  $\mathbf{z}_{l-1}$  is low-dimensional and the forward model  $\mathcal{F}_{l-1}$  is less expensive in comparison to  $\mathcal{F}_l$ . For an efficient Bayesian inference at each scale  $l$ , we are interested in using the posterior distribution at coarse-scale  $(l - 1)$  to provide an informative prior information or improve sampling efficiency in the next finer-scale. This avoids relying completely on inference in a high-dimensional latent space where direct computation of fine-scale details would increase the model complexity. The purpose of inference on fine-scale is to correct the details rather than run long exploration for capturing all appropriate features. A related idea was implemented earlier using hierarchical structured sparse grids in [13]. In summary, the inverse problem is divided into a multiscale posterior estimation, with the inference of parameters proceeding from coarse- to fine-scale. The definition of the multiscale posterior estimation problem is given next.

**Definition 3.3** (Multiscale Posterior Estimation). *Given observations  $\mathcal{D}_{obs}$ , forward models  $\mathcal{F}_l$ , probabilistic encoder model  $p_{\phi_l}(\mathbf{z}_l|\mathbf{x}_l)$ , decoder models  $\mu_{\theta}(\mathbf{z}_l)$ , and prior distributions  $p(\mathbf{z}_l)$  on different scales  $l$  ( $l = 1, 2, \dots, L$ ), explore the posterior distribution  $p(\mathbf{z}_l|\mathcal{D}_{obs})$  in Eq. (10) recursively from coarse- to fine-scales by using MCMC or other posterior modeling techniques.*

### 3.3. Probabilistic generative model

The MDGM is used to generate parameters with different discretization/resolution. We construct such a model based on the variational autoencoder (VAE). The coarsest generative model that involves a single-scale ( $l = 1$ ) is vanilla VAE. It employs the dataset  $\{\mathbf{x}_1^{(i)}\}_{i=1}^N$  (generated as discussed in Section 3.1) sampled from the underlying distribution  $\pi(\mathbf{x}_1)$ , i.e.  $\mathbf{x}_1^{(i)} \stackrel{i.i.d}{\sim} \pi(\mathbf{x}_1)$ . We consider below the probabilistic generative model on a single-scale before deriving the multiscale formulation in Section 3.4. For simplicity of the notation, we drop the subscript 1 in the equations below even though this model will be used in the scale  $l = 1$ .

Given the training dataset  $\{\mathbf{x}^{(i)}\}_{i=1}^N$ , one can introduce a variational family  $q_\phi(\mathbf{z}|\mathbf{x})$  to convert the intractable computation of maximizing the marginal log-likelihood in Eq. (6) into an optimization problem, where  $\phi$  denotes the model parameters. It can be written as follows:

$$\begin{aligned}
 \log p(\mathbf{X}|\boldsymbol{\theta}) &= \sum_{i=1}^N \log p(\mathbf{x}^{(i)}|\boldsymbol{\theta}) \\
 &= \sum_{i=1}^N \log \int p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})p_\theta(\mathbf{z}^{(i)}) d\mathbf{z}^{(i)} \\
 &= \sum_{i=1}^N \log \int q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)}) \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})p_\theta(\mathbf{z}^{(i)})}{q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})} d\mathbf{z}^{(i)} \\
 &\geq \sum_{i=1}^N \underbrace{\int q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)}) \log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})p_\theta(\mathbf{z}^{(i)})}{q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})} d\mathbf{z}^{(i)}}_{\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)})}, \tag{11}
 \end{aligned}$$

where the last step is the application of Jensen's inequality. The above lower bound is called the variational lower bound. For a given training dataset, one can maximize the variational lower bound rather than the marginal log-likelihood. This is an optimization problem with respect to model parameter  $\boldsymbol{\theta}$  and  $\phi$ . The variational lower bound in Eq. (11) can be written as follows,

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) &= \sum_{i=1}^N \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})} [\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})] \\
&= \sum_{i=1}^N \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})} [\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})] - \sum_{i=1}^N D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}^{(i)})). \quad (12)
\end{aligned}$$

The minimization of the  $-\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$  balances the optimization of both the recognition and generative models. Thus the recognition model parameters  $\boldsymbol{\phi}$  are learned jointly with the generative model parameters  $\boldsymbol{\theta}$  [18]. The graphical model is shown in Fig. 3.

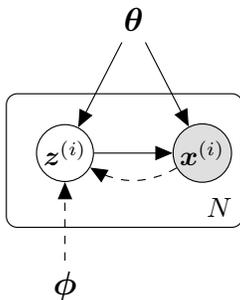


Figure 3: The directed graphical model for the probabilistic model [18]. The latent variable  $\mathbf{z}^{(i)}$  of each configuration  $\mathbf{x}^{(i)}$  is obtained by the probabilistic recognition model  $q_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$ . The variational approximation is indicated with dashed edges and the generative model  $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})$  with solid edges.

To evaluate  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$  in Eq. (12), there are three probability distributions to be identified. As mentioned in Definition 3.1, we shall select appropriate simple distributions for the latent variables  $\mathbf{z}$ . For example, in this paper, we let  $p(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . The  $\mathbb{E}_{\mathbf{z}^{(i)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})} [\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})]$  in Eq. (12) is the expected log-likelihood. It encourages the reconstructed data  $\hat{\mathbf{x}}$  by the decoder to approximate the original data  $\mathbf{x}$ . We assume  $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$  is modeled by a Gaussian distribution  $\mathcal{N}(\mu(\mathbf{z}), \sigma^2 \mathbf{I})$ , where the mean  $\mu(\mathbf{z})$  is the output of a decoder neural network and  $\sigma$  is a constant hyperparameter that does not depend on the decoder so that it can be ignored during optimization. Let  $\hat{\Sigma} = \sigma^2 \mathbf{I}$ , the first term in Eq. (12) then takes the form based on the minibatches:

$$\begin{aligned}
& \sum_{i=1}^N \mathbb{E}_{\mathbf{z}^{(i)} \sim q_{\phi}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})} (\log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})) \\
= & \sum_{i=1}^N \mathbb{E}_{\mathbf{z}^{(i)} \sim q_{\phi}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})} \left[ -\frac{1}{2} (\mathbf{x}^{(i)} - \mu(\mathbf{z}^{(i)}))^T \hat{\Sigma}^{-1} (\mathbf{x}^{(i)} - \mu(\mathbf{z}^{(i)})) \right] + \text{constant} \\
\propto & -\sum_{i=1}^N \mathbb{E}_{\mathbf{z}^{(i)} \sim q_{\phi}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})} \left[ (\mathbf{x}^{(i)} - \mu(\mathbf{z}^{(i)}))^T (\mathbf{x}^{(i)} - \mu(\mathbf{z}^{(i)})) \right] \\
\approx & -\frac{N}{n} \frac{1}{m} \sum_{i=1}^n \sum_{j=1}^m \|\mathbf{x}^{(i)} - \mu(\mathbf{z}^{(i,j)})\|^2, \quad \mathbf{z}^{(i,j)} \sim q_{\phi}(\mathbf{z}^{(i,j)}|\mathbf{x}^{(i)}), \tag{13}
\end{aligned}$$

where  $n$  denotes the number of training samples of  $\mathbf{x}$  (also referred to as the batch size in the training of deep neural networks). For each epoch, there are  $\frac{N}{n}$  minibatches, each batch uniformly sampled from the dataset  $\{\mathbf{x}^{(i)}\}_{i=1}^N$ .  $m$  is the number of  $\mathbf{z}$  samples from  $q_{\phi}(\mathbf{z}|\mathbf{x})$  for an expectation approximation using the Monte Carlo method. One can also refer to Eq. (13) as the reconstruction error.

For the  $D_{KL}(q_{\phi}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})||p(\mathbf{z}^{(i)}))$  in Eq. (12),  $q_{\phi}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$  is taken as a Gaussian distribution, where the mean  $\hat{\mu}(\mathbf{x})$  and the variance  $\hat{\sigma}^2(\mathbf{x})$  are outputs of the encoder network. The KL-divergence can be analytically computed when both distributions are Gaussian [18]. The KL-divergence works as an objective function for the optimization problem with respect to the encoder parameters  $\phi$ . Based on the minibatches, it can be written as:

$$\sum_{i=1}^N D_{KL}(q_{\phi}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})||p(\mathbf{z})) = \frac{N}{2n} \sum_{i=1}^n \sum_{k=1}^d (\hat{\mu}_k^2(\mathbf{x}^{(i)}) + \hat{\sigma}_k^2(\mathbf{x}^{(i)}) - \log \hat{\sigma}_k^2(\mathbf{x}^{(i)}) - 1), \tag{14}$$

where  $\hat{\mu}_k(\mathbf{x})$  and  $\hat{\sigma}_k(\mathbf{x})$  denote the  $k$ -th element of the mean and standard deviation, respectively, which are outputs of the encoder network. The results of Eqs. (13) and (14) summarize the objective function for optimization of the encoder and decoder neural networks.

Inspired from  $\beta$ -VAE [53], we slightly modify the underlying loss function. This modification of the VAE results by adding an extra hyperparameter  $\beta$  to the KL divergence. This hyperparameter can constrict the capacity of the latent bottleneck and encourage a disentangled representation. We would like the individual dimensions of the latent variable  $\mathbf{z}$  to be interpretable or to correspond to some features of parameters thus disentangling

the true variation of data [53, 54]. A good interpretability and factorized representation for the latent variable will facilitate the feature exploration in MCMC. Note that [55] has shown that  $\beta$ -VAE is an important method in disentangled representation for learning the compositional and hierarchical visual concept. Choosing an appropriate hyperparameter  $\tilde{\beta}$  for loss function is important in MDGM. We can define the loss function  $\tilde{\mathcal{L}} \equiv -\frac{n}{N}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$  in each training iteration, the loss function for the single-scale probabilistic generative model as follows:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{\tilde{\beta}}{2} \sum_{i=1}^n \sum_{k=1}^d (\hat{\mu}_k^2(\mathbf{x}^{(i)}) + \hat{\sigma}_k^2(\mathbf{x}^{(i)}) - \log \hat{\sigma}_k^2(\mathbf{x}^{(i)}) - 1) + \frac{1}{m} \sum_{i=1}^n \sum_{j=1}^m \|\mathbf{x}^{(i)} - \mu(\mathbf{z}^{(i,j)})\|^2. \quad (15)$$

**Remark 1.** *In the vanilla VAE, the objective function considers the reconstruction error and the KL-divergence to be of equivalent importance. The hyperparameter  $\beta$  is introduced to break this balance. More specifically, high values of  $\beta$  put more emphasis on the latent space approximation than on the reconstruction, expediting the learning of notable feature variations but bringing blurred minutiae. For example, in the channelized permeability experiment, we noted that high  $\beta$  is conducive to capturing the continuous channel structures while losing much fidelity in local variations.*

To optimize the parameters  $\boldsymbol{\theta}$  and  $\boldsymbol{\phi}$  in neural networks, one could employ stochastic gradient descent (SGD) or other gradient-based optimization algorithms related to back propagation. Note that the second term in Eq. (15) is an expectation approximation using the Monte Carlo method that samples  $\mathbf{z}^{(i,j)}$  from  $q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,j)}|\mathbf{x}^{(i)})$ . But the expectation computation involving sampling with a high variance will reflect on the gradient estimation, which leads to an unfavorable influence on optimization. To make it trainable and back propagate the gradient correctly, the reparameterization trick is introduced. The latent variables  $\mathbf{z}$  are expressed by a differentiable transformation  $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$  with respect to an auxiliary independent random variable  $\boldsymbol{\epsilon}$ . In the Gaussian distribution case, we let  $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ , sampling  $\mathbf{z}$  via such a  $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ :

$$\mathbf{z} = \hat{\boldsymbol{\mu}}(\mathbf{x}) + \hat{\boldsymbol{\sigma}}(\mathbf{x}) \odot \boldsymbol{\epsilon}, \quad (16)$$

where  $\odot$  refers to an element-wise product. The illustration of forward and back propagation computation with the reparameterization trick is shown in Fig. 4.

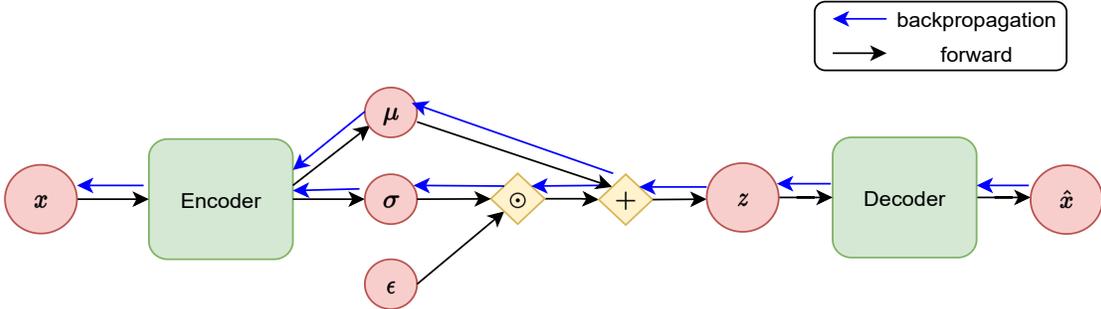


Figure 4: Illustration of the network architecture. The black arrows denote forward computation in the encoder/decoder networks and the blue arrows indicate the feasible implementation of back propagation under the reparameterization trick.

**Remark 2.** *The reparameterization trick makes the random variable  $\mathbf{z}$  to only depend on two deterministic variables by introducing an auxiliary random variable  $\epsilon$  sampled from the standard Gaussian distribution. It scales  $\epsilon$  by the variance  $\hat{\sigma}(\mathbf{x})$  and shifts it by the mean  $\hat{\mu}(\mathbf{x})$ . The operators  $+$  and  $\odot$  in Eq. (16) are differentiable, which makes the gradient computation achievable. Numerical experiments indicate that  $m$  in Eq. (15) can be set to 1 when  $n$  is large enough.*

As discussed in Definition 3.1, we constructed the probabilistic models  $p_{\theta}(\mathbf{x}|\mathbf{z})$  and  $q_{\phi}(\mathbf{z}|\mathbf{x})$  to sample  $\mathbf{x}$  given its corresponding latent variable  $\mathbf{z}$  and to map the parameters  $\mathbf{x}$  to the latent space, respectively. Based on the objective function in Eq. (15) and neural network in Fig. 4 (detailed architecture see Appendix A), one can optimize the network parameters  $\phi$  and  $\theta$ . The implementation procedure is summarized in Algorithm 1.

This model only involves a single-scale parameter representation learning, while MDGM is a multi-stage and recursive training procedure from coarse to fine, i.e. the training output in the coarse-scale model is the input to the next finer-scale model. The coarsest-scale probabilistic models  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  and  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$  are outputs of Algorithm 1. The generative model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  is used for the estimation of the posterior  $\pi(\mathbf{x}_1|\mathcal{D}_{obs})$  in the coarsest-scale by standard MCMC (see Section 3.5). The recognition model  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$  is the input (as a pre-trained model) to the second-scale ( $l = 2$ ) generative model training (see Section 3.4).

---

**Algorithm 1** Training probabilistic generative model

---

**Input:** Dataset  $\{\mathbf{x}^{(i)}\}_{i=1}^N$ , training epoch  $E$ , batch size  $n$ , learning rate  $\eta$ , hyperparameter  $\tilde{\beta}$ ,  $m = 1$ .

- 1: Initialize  $\phi, \theta \leftarrow$  Initialize parameters
- 2: **while**  $epoch < E$  **do**
- 3:    $\mathbf{x}^n \leftarrow$  Sample minibatch  $n$  datapoints from  $\{\mathbf{x}^{(i)}\}_{i=1}^N$
- 4:    $\epsilon^n \leftarrow$  Sample  $n$  noise from Gaussian distribution  $\mathcal{N}(0, I)$
- 5:    $\mathbf{z}^n \leftarrow$  Compute by encoder network with Eq. (16)
- 6:    $\nabla_{\theta} \tilde{\mathcal{L}}, \nabla_{\phi} \tilde{\mathcal{L}} \leftarrow$  Calculate gradients of  $\tilde{\mathcal{L}}(\theta, \phi; \mathbf{z}^n, \mathbf{x}^n, \tilde{\beta}, m)$  in Eq. (15)
- 7:    $\theta = \theta - \eta \nabla_{\theta} \tilde{\mathcal{L}} \leftarrow$  update  $\theta$  using gradient-based optimization algorithm (e.g. SGD or Adam)
- 8:    $\phi = \phi - \eta \nabla_{\phi} \tilde{\mathcal{L}} \leftarrow$  update  $\phi$  using gradient-based optimization algorithm (e.g. SGD or Adam)
- 9: **end while**

**Output:** probabilistic encoder  $q_{\phi}(\mathbf{z}|\mathbf{x})$ , probabilistic decoder  $p_{\theta}(\mathbf{x}|\mathbf{z})$ .

---

### 3.4. Multiscale deep generative model (MDGM)

In Section 3.3, we presented the single-scale parameter generative model in a probabilistic perspective and discussed how to use deep neural networks for its implementation. In this section, we enhance this model to a multiscale framework. Based on the parameter data generation procedure in Section 3.1, we start from the training data in the finest-scale and coarse grain them recursively to represent the training data in the coarsest-scale. We demonstrate the MDGM by explaining how to train it in the  $l$ -th scale as an example based on the assumption that we obtained the pre-trained encoder and decoder networks in the  $(l - 1)$ -th scale.

To construct the connection among various scales, we design a special latent space for the finer-scale. In particular, its latent variable  $\mathbf{z}_l$  inherits the latent variable  $\mathbf{z}_{l-1}$  of the previous scale and augments it with an additional latent variable  $\mathbf{z}_l^*$ . These two variables are independent. This is in principle similar to the Bayesian approach followed in [13] where a hierarchical sparse grid approximation was used to represent an unknown parameter field

at different scales.

As we discussed in Definition 3.2, we also need to train the recognition model  $q_\phi(\mathbf{z}_l|\mathbf{x}_l)$  and generative model  $p_\theta(\mathbf{x}_l|\mathbf{z}_l)$  in scale  $l$  with the same objective function as in Eq. (6). The difference with standard generative models is that the recognition model is divided into two parts in order to encode the parameter  $\mathbf{x}$  with the independent latent variables  $\mathbf{z}_{l-1}$  and  $\mathbf{z}_l^*$ . As shown in Fig. 5, the encoder network includes a pre-trained encoder network and an augmented encoder network.  $q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_l)$  is computed by adopting the upscaling operator over  $\mathbf{x}_l$  firstly and then using the previous scale encoder network. It can be modeled by

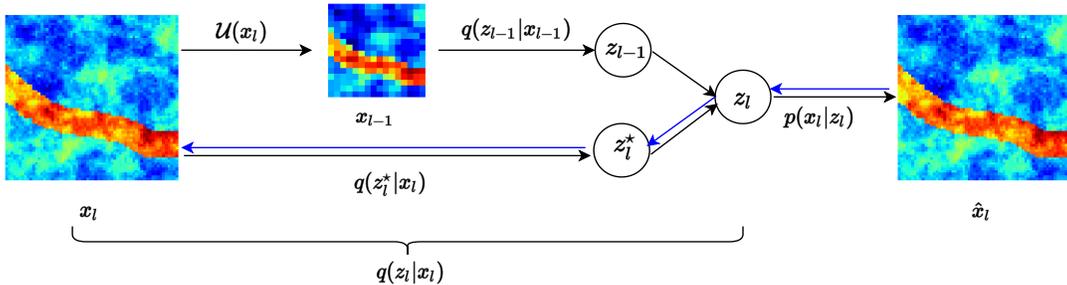


Figure 5: Schematic illustration of the probabilistic generative model in the  $l$ -th scale. Black arrows above denote the forward computation, and blue arrows in reverse direction denote the back propagation in gradient-based optimization.  $q(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})$  as an input is a pre-trained network, whereas the model parameters in  $q(\mathbf{z}_l^*|\mathbf{x}_l)$  and  $p(\mathbf{x}_l|\mathbf{z}_l)$  need to be learned. For details on how to concatenate  $\mathbf{z}_{l-1}$  with  $\mathbf{z}_l^*$  to obtain the latent variable  $\mathbf{z}_l$  refer to Appendix B.

$$q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_l) = \int q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})\pi(\mathbf{x}_{l-1}|\mathbf{x}_l)d\mathbf{x}_{l-1}. \quad (17)$$

Since the upscaling operator  $\mathcal{U}$  is deterministic, we can write the following:

$$q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_l) = q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_{l-1}). \quad (18)$$

As in the single-scale probabilistic generative model in the last section, the variational lower bound  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x})$  consists of two parts as described in Eq. (12), i.e.  $\mathbb{E}_{q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})]$  and  $D_{KL}(q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}^{(i)}))$ . The first part in the  $l$ -th scale is formulated as the reconstruction error introduced in Eq. (13), which involves the decoder model  $p_{\theta_l}(\mathbf{x}_l|\mathbf{z}_l)$  and its model parameter  $\boldsymbol{\theta}_l$ . The second part is much distinct with  $p(\mathbf{z}_l)$  being the target distribution in the KL-divergence. It is an isotropic multivariate Gaussian distribution as well,

but it can be viewed as the joint distribution of  $(\mathbf{z}_{l-1}, \mathbf{z}_l^*)$ , where  $\mathbf{z}_{l-1}$ ,  $\mathbf{z}_l^*$  are independent. Thus  $p(\mathbf{z}_{l-1})$  and  $p(\mathbf{z}_l^*)$  are also isotropic multivariate Gaussian distributions. Combining with Eq. (18), we rewrite the KL-divergence as follows:

$$\begin{aligned}
D_{KL}(q_\phi(\mathbf{z}_l|\mathbf{x}_l)\|p(\mathbf{z}_l)) &= \int q_\phi(\mathbf{z}_l|\mathbf{x}_l) \log \frac{q_\phi(\mathbf{z}_l|\mathbf{x}_l)}{p(\mathbf{z}_l)} d\mathbf{z}_l \\
&= \int \int q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_l) q_\phi(\mathbf{z}_l^*|\mathbf{x}_l) \log \frac{q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_l) q_\phi(\mathbf{z}_l^*|\mathbf{x}_l)}{p(\mathbf{z}_{l-1}) p(\mathbf{z}_l^*)} d\mathbf{z}_{l-1} d\mathbf{z}_l^* \\
&= \int \int q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_{l-1}) q_\phi(\mathbf{z}_l^*|\mathbf{x}_l) \left[ \log \frac{q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})}{p(\mathbf{z}_{l-1})} + \log \frac{q_\phi(\mathbf{z}_l^*|\mathbf{x}_l)}{p(\mathbf{z}_l^*)} \right] d\mathbf{z}_{l-1} d\mathbf{z}_l^* \\
&= \int q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_{l-1}) \log \frac{q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})}{p(\mathbf{z}_{l-1})} d\mathbf{z}_{l-1} \int q_\phi(\mathbf{z}_l^*|\mathbf{x}_l) d\mathbf{z}_l^* \\
&+ \int q_\phi(\mathbf{z}_l^*|\mathbf{x}_l) \log \frac{q_\phi(\mathbf{z}_l^*|\mathbf{x}_l)}{p(\mathbf{z}_l^*)} d\mathbf{z}_l^* \int q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_{l-1}) d\mathbf{z}_{l-1} \\
&= D_{KL}(q_\phi(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})\|p(\mathbf{z}_{l-1})) + D_{KL}(q_\phi(\mathbf{z}_l^*|\mathbf{x}_l)\|p(\mathbf{z}_l^*)), \tag{19}
\end{aligned}$$

where  $\phi$  in the  $l$ -th scale is  $\phi_l$  containing  $\phi_{l-1}$  and  $\phi_l^*$ , where  $\phi_{l-1}$  denotes the parameters of the encoder network  $q(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})$  in the  $(l-1)$ -th scale and  $\phi_l^*$  denotes the parameters of the augmented encoder network  $q(\mathbf{z}_l^*|\mathbf{x}_l)$  in the  $l$ -th scale. Recall that we need to employ the pre-trained encoder network  $q_{\phi_{l-1}}(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})$  directly in the  $l$ -th scale training because it should ensure that the finer-scale generative model shares common latent variables  $\mathbf{z}_{l-1}$  with the coarser-scale, so

$$D_{KL}(q_\phi(\mathbf{z}_l|\mathbf{x}_l)\|p(\mathbf{z}_l)) = D_{KL}(q_{\phi_l^*}(\mathbf{z}_l^*|\mathbf{x}_l)\|p(\mathbf{z}_l^*)) + \text{constant}. \tag{20}$$

We also use gradient-based optimization algorithm to train the MDGM where the back propagation is only applied in the decoder network and augmented encoder network as blue arrows illustrated in Fig. 5. We can formulate the loss function for each training iteration like Eq. (15) as follows:

$$\begin{aligned}
\tilde{\mathcal{L}}(\boldsymbol{\theta}_l, \phi_l^*) &= \frac{\tilde{\beta}}{2} \sum_{i=1}^n \sum_{k=1}^{d_l^*} \left( \hat{\mu}_{ik}^2(\mathbf{x}_l^{(i)}) + \hat{\sigma}_{ik}^2(\mathbf{x}_l^{(i)}) - \log \hat{\sigma}_{ik}^2(\mathbf{x}_l^{(i)}) - 1 \right) \\
&+ \frac{1}{m} \sum_{i=1}^n \sum_{j=1}^m \left\| \mathbf{x}_l^{(i)} - \mu_l(\mathbf{z}_l^{(i,j)}) \right\|^2, \tag{21}
\end{aligned}$$

where  $\hat{\mu}_{lk}(\cdot)$  and  $\hat{\sigma}_{lk}(\cdot)$  denote the  $k$ -th element in the mean and standard deviation, respectively, which are outputs of the  $l$ -th scale augmented encoder network, and  $\mu_l(\cdot)$  is the output of the  $l$ -th scale decoder network. Here,  $m$  is still set to be 1 when the batch size  $n$  is large enough.  $d_l^*$  is the dimension of the latent variable  $\mathbf{z}_l^*$ . The hyperparameter  $\tilde{\beta}$  is determined by balancing the disentanglement of the parameter features and reconstruction.

In summary, assume that we obtained the probabilistic encoder and decoder models in the previous  $(l - 1)$  level. We can then train the augmented encoder network and decoder networks using the objective function in Eq. (21) so that the probabilistic encoder  $q_{\phi_l}(\mathbf{z}_l|\mathbf{x}_l)$  and the probabilistic decoder  $p_{\theta_l}(\mathbf{x}_l|\mathbf{z}_l)$  in the  $l$ -th scale are acquired. For details see Algorithm 2. The architectures of the augmented encoder and decoder networks are the same as those in the encoder and decoder networks for the single-scale model (refer to Appendix A).

We have introduced so far the probabilistic generative model in a single-scale in the last section and the concept of generative modeling across two scales in this section. Since these unsupervised learning models only need the unlabeled data, we only provide the finest-scale dataset  $\{\mathbf{x}_L^{(i)}\}_{i=1}^N$  sampled from the underlying prior distribution  $\pi(\mathbf{x}_L)$  and adopt the deterministic upscaling operator  $\mathcal{U}$  to generate the dataset in the scale of interest. The training of the generative models proceeds from the coarsest- to the finest-scale. The coarsest-scale generative model is trained by Algorithm 1. From the second- to the finest-scale, the generative model is trained recursively by Algorithm 2. Once the generative models are trained in the scales of interest, they can be integrated with MCMC to estimate the posterior of the parameters in each scale.

**Remark 3.** *The training of the  $l$ -th scale generative model employs the  $(l - 1)$ -th scale encoder network, where  $\mathbf{z}_{l-1}$  is sampled from  $q_{\phi_{l-1}}(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})$  using the reparameterization trick. A  $\mathbf{z}_{l-1}$  with high-variance will impact the training stability and convergence inflicting big noise in the latent variable  $\mathbf{z}_l$ . We address this problem in the training procedure by replacing  $\mathbf{z}_{l-1}$  with its mean  $\hat{\mu}_{l-1}(\mathbf{x}_{l-1})$ .*

### 3.5. Bayesian inversion with the DGM

In Bayesian inversion, the Metropolis-Hastings (MH) algorithm [56, 57] is a popular MCMC method to approximate the posterior distribution [58]. As discussed in Definition 3.3,

---

**Algorithm 2** Training of the multiscale probabilistic generative model in the  $l$ -th scale

---

**Input:**  $\{\mathbf{x}_i\}_{i=1}^N$ , pre-trained encoder  $q_{\phi_{l-1}}(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})$ , training epoch  $E$ , batch size  $n$ , learning rate  $\eta$ , hyperparameter  $\tilde{\beta}$ ,  $m = 1$

- 1: Initialize  $\phi_l, \theta_l^* \leftarrow$  Initialize parameters
  - 2: **while**  $epoch < E$  **do**
  - 3:    $\mathbf{x}^n \leftarrow$  Sample minibatch  $M$  datapoints from  $\{\mathbf{x}_i\}_{i=1}^N$
  - 4:    $\mathbf{z}_{l-1}^n \leftarrow$  Compute by  $q_{\phi_{l-1}}(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})$  for each data
  - 5:    $\epsilon \leftarrow$  Sample  $n$  noise variables from the Gaussian distribution  $p(\epsilon)$
  - 6:    $\mathbf{z}_l^n \leftarrow$  Compute  $\mathbf{z}_l^*$  by the augmented encoder network in Eq. (16), and concatenate with  $\mathbf{z}_{l-1}$  for each data
  - 7:    $\nabla_{\theta_l^*} \tilde{\mathcal{L}}, \nabla_{\phi_l} \tilde{\mathcal{L}} \leftarrow$  Calculate gradients of  $\tilde{\mathcal{L}}(\theta_l^*, \phi_l; \mathbf{z}_l^n, \mathbf{x}^n, \tilde{\beta}, m)$  in Eq. (21) with respect to  $\theta_l^*$  and  $\phi_l$
  - 8:    $\theta_l^* = \theta - \eta \nabla_{\theta_l^*} \tilde{\mathcal{L}} \leftarrow$  using gradient-based optimization algorithm (e.g. SGD or Adam)
  - 9:    $\phi_l = \phi - \eta \nabla_{\phi_l} \tilde{\mathcal{L}} \leftarrow$  using gradient-based optimization algorithm (e.g. SGD or Adam)
  - 10: **end while**
  - 11:  $q_{\phi_l}(\mathbf{z}_l|\mathbf{x}_l) \leftarrow$  combine the pre-trained encoder  $q_{\phi_{l-1}}(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})$  with  $q_{\phi_l^*}(\mathbf{z}_l^*|\mathbf{x}_l)$
- Output:** probabilistic encoder  $q_{\phi_l}(\mathbf{z}_l|\mathbf{x}_l)$ , probabilistic decoder  $p_{\theta_l}(\mathbf{x}_l|\mathbf{z}_l)$ .
- 

we design a multiscale scheme aimed to realize the high-dimensional parameter estimation in inverse problems where the posterior distribution is approximated recursively. We utilize the previous coarser-scale estimation information as we proceed with inference in the next immediate finer-scale. Note that the coarsest-scale inference only involves single-scale information in which the parameterization employs the DGM given in Algorithm 1.

For the coarsest-scale posterior distribution, we are interested in combining the standard MH algorithm with the single-scale generative model to estimate the parameters  $\mathbf{x}_1 \in \mathbb{R}^{M_1}$  using the low-dimensional latent variable  $\mathbf{z}_1 \in \mathbb{R}^{d_1}$ , where  $M_1$  is the dimension of the coarsest-scale parameter  $\mathbf{x}_1$  that also determines the spatial discretization of the coarsest-scale forward model  $\mathcal{F}_1$ , and  $d_1$  is the dimension of the coarsest-scale latent space. MCMC based on the pre-trained model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  allows us to explore the posterior approximation of  $\mathbf{z}_1$ . Recall that the model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  is a Gaussian distribution with mean  $\mu_{\theta_1}(\mathbf{z}_1)$  and standard

derivation  $\sigma$ , where  $\mu_{\theta_1}(\mathbf{z}_1)$  is the output of the decoder network in Algorithm 1 and  $\sigma$  is a hyperparameter that determines the noise level. In the test procedure, we can thus define  $\mathbf{x}_1 = \mu_{\theta_1}(\mathbf{z}_1)$  as a mapping from the latent space to the original parameter space. Then one can sample in the continuous latent space to produce various parameters  $\mathbf{x}_1$  using this mapping. In this way, the problem of inferring  $\mathbf{x}_1$  becomes equivalent to the one of inferring  $\mathbf{z}_1$ .

Given the observed data  $\mathcal{D}_{obs}$ , the forward function  $\mathcal{F}_1$ , the decoder model  $\mathbf{x}_1 = \mu_{\theta_1}(\mathbf{z}_1)$ , and the prior distribution  $p(\mathbf{z}_1)$ , the unnormalized posterior distribution in the coarsest-scale is as follows:

$$\pi(\mathbf{z}_1|\mathcal{D}_{obs}) \propto \mathcal{L}_e(\mathcal{D}_{obs}|\mathbf{z}_1)p(\mathbf{z}_1), \quad (22)$$

where the prior distribution  $p(\mathbf{z}_1)$  is  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ .

For the above target distribution in the coarsest-scale, we only need to create a Markov chain with length  $N_{ite}$  for the latent variable  $\mathbf{z}_1$ . The proposal distribution  $\pi_q(\mathbf{z}'_1|\mathbf{z}_1^{(j)})$  can be the simple random walk [59] or a preconditioned Crank-Nicholson (pCN) algorithm [38, 60, 61]. To initialize the first state  $\mathbf{z}_1^{(1)}$  for this Markov chain, one can sample  $\mathbf{z}_1^{(1)}$  from the prior distribution  $p(\mathbf{z}_1)$ . For each  $j \geq 1$ , we first sample a candidate  $\mathbf{z}'_1$  from the proposal distribution  $\pi_q(\cdot|\mathbf{z}_1^{(j)})$  and then reconstruct the spatially-varying parameter  $\mathbf{x}_1$  by the decoder model  $\mathbf{x}_1 = \mu_{\phi_1}(\mathbf{z}_1)$ . One can compute the likelihood function and the Metropolis acceptance ratio  $\alpha$  using the forward model  $\mathcal{F}_1(\mathbf{x}_1)$ . The acceptance ratio  $\alpha$  is defined as in MH algorithm:

$$\alpha = \min \left( 1, \frac{\pi(\mathbf{z}'_1|\mathcal{D}_{obs})\pi_q(\mathbf{z}_1^{(j)}|\mathbf{z}'_1)}{\pi(\mathbf{z}_1^{(j)}|\mathcal{D}_{obs})\pi_q(\mathbf{z}'_1|\mathbf{z}_1^{(j)})} \right). \quad (23)$$

With probability  $\alpha$ , the candidate sample  $\mathbf{z}'_1$  is accepted, otherwise rejected. Specifically, in each iteration, the state includes not only the latent variable  $\mathbf{z}_1$  but also its corresponding spatially-varying parameter  $\mathbf{x}_1$ . The Markov chain typically takes some iterations to reach its stationary distribution, which depends on the target distribution. The first  $n_b$  states of the Markov chain (burn-in stage) are thrown-away to ensure a valid approximation of the target distribution. Once the latent variable  $\mathbf{z}_1$  is estimated, its corresponding parameter

samples  $\{\mathbf{x}_1^{(j)}\}_{j=n_b}^{N_{ite}}$  can be viewed as samples from the posterior distribution  $\pi(\mathbf{x}_1|\mathcal{D}_{obs})$ . A summary of the MH algorithm for Bayesian inverse modeling based on the deep generative model is given in Algorithm 3. This algorithm can be applied to solve any high-dimensional parameter estimation in the single-scale.

In the multiscale context, the coarsest posterior was approximated by Algorithm 3. But there are still two main problems to be resolved by invoking a finer-scale model. One is that the high-resolution parameter estimation that can present richer details is desired; the other is the inaccurate forward model introduces an epistemic uncertainty or model error, which will reflect on the parameter estimation. The model error  $\delta_l$  in the  $l$ -th scale is defined by

$$\delta_l = \mathcal{F}_L(\mathbf{x}_L) - \mathcal{F}_l(\mathbf{x}_l). \quad (24)$$

This model error  $\delta_l$  will be zero if and only if we adopt the “true model”  $\mathcal{F}_L$ . The posterior distribution  $\pi(\mathbf{x}_L|\mathcal{D}_{obs})$  is the ultimate goal of our inverse problem.

### 3.6. Multiscale Bayesian inversion with the MDGM

To eliminate the model error and estimate the fine-scale parameter, we integrate the MDGM with the MH algorithm. We refer to this algorithm as MH-MDGM. Such a procedure involves inference across scales and allows us to generate samples on each scale for estimating the unknown parameter proceeding from the coarsest- to the finest-scale. Leveraging the latent space construction in the MDGM, the difference with the single-scale method is that the fine-scale estimation integrated with the MDGM method can inherit the coarse estimation resulting in a highly-efficient sampling procedure. This approach is applied from the second-coarsest to the finest-scale where inference in each scale proceeds recursively utilizing information at the previous coarser-scale.

For any  $2 \leq l \leq L$ , let us suppose that the approximate posterior distribution  $\pi(\mathbf{z}_{l-1}|\mathcal{D}_{obs})$  in the  $(l-1)$ -th scale has been obtained. We thus have obtained the posterior samples  $\{\mathbf{z}_{l-1}^{(i)}\}_{i=n_{l-1}}^{N_{l-1}}$  and their corresponding parameter samples  $\{\mathbf{x}_{l-1}^{(i)}\}_{i=n_{l-1}}^{N_{l-1}}$ . As defined in Eqs. (7) and (22), we are concerned about the finer-scale  $\pi(\mathbf{z}_l|\mathcal{D}_{obs})$  posterior estimation. Using the MCMC method, one can sample  $\mathbf{z}_l$  to generate the parameter  $\mathbf{x}_l$  by the decoder model  $\mathbf{x}_l = \mu_{\theta_l}(\mathbf{z}_l)$  in MDGM, and then solve the forward model  $\mathcal{F}_l(\mathbf{x}_l)$  to evaluate the likelihood

---

**Algorithm 3** The Metropolis-Hastings with Deep Generative Model (MH-DGM) Algorithm for the first-scale posterior distribution approximation

---

**Input:** unnormalized distribution  $\pi(\mathbf{z}|\mathcal{D}_{obs})$ , proposal distribution  $\pi_q(\cdot)$ , iteration number

$N_{ite}$ , decoder model  $\mu(\mathbf{z})$ , burn-in length  $n_b$

1: Initialize  $\mathbf{z}^{(1)}, \mathbf{z}^{(1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

2: **for**  $j = 1 : N$  **do**

3:     Draw  $\mathbf{z}' \sim \pi_q(\cdot|\mathbf{z}^{(j)})$

4:     Compute the spatially-varying parameter  $\mathbf{x}' = \mu(\mathbf{z}')$  by the decoder model

5:     Compute the likelihood function by solving the forward model  $\mathcal{F}(\mathbf{x}')$

6:     Compute the acceptance ratio

$$\alpha = \min \left( 1, \frac{\pi(\mathbf{z}'|\mathcal{D}_{obs})\pi_q(\mathbf{z}^{(j)}|\mathbf{z}')}{\pi(\mathbf{z}^{(j)}|\mathcal{D}_{obs})\pi_q(\mathbf{z}'|\mathbf{z}^{(j)})} \right)$$

7:     Draw  $\rho$  from uniform distribution  $\mathcal{U}[0, 1]$

8:     **if**  $\rho < \alpha$  **then**

9:         Let  $\mathbf{z}^{(j+1)} = \mathbf{z}', \mathbf{x}^{(j+1)} = \mathbf{x}'$

10:     **else**

11:         Let  $\mathbf{z}^{(j+1)} = \mathbf{z}^j, \mathbf{x}^{(j+1)} = \mathbf{x}^{(j)}$

12:     **end if**

13: **end for**

**Output:** posterior samples  $\{\mathbf{x}^{(i)}\}_{i=n_b}^{N_{ite}}$

---

function and the acceptance ratio.

Suppose that the  $j$ -th state of the Markov chain in the  $l$ -th scale is  $\mathbf{z}_l^{(j)}$ . The acceptance ratio of the reject/accept scheme for the  $(j+1)$ -th state in the MH-MDGM is still the same as that in the single-scale MH algorithm, i.e.

$$\alpha = \min \left( 1, \frac{\pi(\mathbf{z}'_l|\mathcal{D}_{obs})\pi_q(\mathbf{z}_l^{(j)}|\mathbf{z}'_l)}{\pi(\mathbf{z}_l^{(j)}|\mathcal{D}_{obs})\pi_q(\mathbf{z}'_l|\mathbf{z}_l^{(j)})} \right), \quad (25)$$

where the proposal  $\pi_q(\mathbf{z}'_l|\mathbf{z}_l^{(j)})$  is a particular distribution since  $\mathbf{z}_l = (\mathbf{z}_{l-1}, \mathbf{z}_l^*)$  contains two independent random variables, which have different connotation in the MH-MDGM. The

proposal is taken to have a factorized form as follows [38, 62]:

$$\pi_q(\mathbf{z}'_l | \mathbf{z}_l^{(j)}) = \pi_q(\mathbf{z}'_{l-1} | \mathbf{z}_{l-1}^{(j)}, \mathbf{z}_l^{*(j)}) \pi_q(\mathbf{z}'_l | \mathbf{z}'_{l-1}, \mathbf{z}_l^{*(j)}). \quad (26)$$

In the MDGM,  $\mathbf{z}_l^*$  is sampled from an isotropic Gaussian distribution, which is independent of  $\mathbf{z}_{l-1}$ . The proposal distribution for  $\mathbf{z}_l^{*(j)}$  can be simplified as

$$\pi_q(\mathbf{z}'_l | \mathbf{z}_{l-1}^{(j)}, \mathbf{z}_l^{*(j)}) = \pi_q(\mathbf{z}'_l | \mathbf{z}_l^{*(j)}). \quad (27)$$

This can be a simple random walk or the pCN algorithm. Exploring  $\mathbf{z}_l^*$  provides additional information to that obtained at the  $(l-1)$ -scale allowing us to enrich the obtained local details and correct global features. One can set a big step size for  $\mathbf{z}_l^*$  in the proposal distribution to explore and detect various local patterns. In the MH-MDGM, the initial state  $\mathbf{z}_l^*$  can be sampled from the prior distribution  $p(\mathbf{z}_l^*)$ .

Based on the independence assumption, one can define a proposal distribution in the  $l$ -th scale for the latent variable  $\mathbf{z}_{l-1}$  that connects adjacent scales as follows:

$$\pi_q(\mathbf{z}'_{l-1} | \mathbf{z}_{l-1}^{(j)}, \mathbf{z}_l^{*(j)}) = \pi_q(\mathbf{z}'_{l-1} | \mathbf{z}_{l-1}^{(j)}). \quad (28)$$

Once  $\mathbf{z}_{l-1}$  has been estimated in the previous-scale, some multiscale estimation methods like [13] will not correct and estimate it again in the finer-scale. However, note that the model error that resulted from the coarse-scale solver can impact the posterior estimation. We treat the latent variable  $\mathbf{z}_{l-1}$  as a random variable in the  $l$ -th scale estimation and update it invoking the accurate forward model. To exploit the  $(l-1)$ -th scale posterior result, we assign an initial state for  $\mathbf{z}_{l-1}$  in the  $l$ -th scale estimation, which can inherit the  $(l-1)$ -th scale Markov state, greatly reducing the time to obtain the stationary distribution. The obvious approach is to use the last state in the  $(l-1)$ -th scale, while another reasonable choice is the mean of the  $(l-1)$ -th posterior estimation. Note that we use a set of posterior samples  $\{\mathbf{x}_{l-1}^{(i)}\}_{i=n_{l-1}}^{N_{l-1}}$  to approximate the posterior distribution  $\pi(\mathbf{x}_{l-1} | \mathcal{D}_{obs})$ , where  $n_{l-1}$ ,  $N_{l-1}$  denote the lengths of burn-in and Markov chain, respectively.  $\mathbf{z}_{l-1}$  encodes the coarse-scale estimation information via the encoder network  $\pi_{\theta_{l-1}}(\mathbf{z}_{l-1} | \mathbf{x}_{l-1})$ . We can assign the initial state  $\mathbf{z}_{l-1}^{(1)}$  in the  $l$ -th scale estimation to be  $\mathbf{z}_{l-1}^{(1)} = \arg \max \pi_{\theta_{l-1}}(\mathbf{z}_{l-1} | \bar{\mathbf{x}}_{l-1})$ , where  $\bar{\mathbf{x}}_{l-1}$  is the

mean of the posterior samples  $\{\mathbf{x}_{l-1}^{(i)}\}_{i=n_{l-1}}^{N_{l-1}}$ . Since the latent variable  $\mathbf{z}_{l-1}$  has been estimated and can impact the crucial global features of the parameter  $\mathbf{x}_l$  generation using the decoder model  $\mathbf{x}_l = \mu_{\theta_l}(\mathbf{z}_l)$ , it should be assigned a small step size in the proposal distribution.

Combining Eq. (28) with Eq. (27), we obtain the conclusion in Proposition 1. The MH-MDGM algorithm for the  $l$ -th scale posterior distribution approximation is presented in Algorithm 4.

**Remark 4.** *In the  $(l - 1)$ -th scale posterior approximation, the Markov chain generates samples  $\{\mathbf{z}_{l-1}^{(i)}\}_{i=n_{l-1}}^{N_{l-1}}$  and their corresponding parameter samples  $\{\mathbf{x}_{l-1}^{(i)}\}_{i=n_{l-1}}^{N_{l-1}}$ . One cannot directly use the posterior samples  $\mathbf{z}_{l-1}^{(i)}$  in the  $l$ -th fine-scale inference. The reason is that these samples are only related to the decoder model  $\mu_{\theta_{l-1}}(\mathbf{z}_{l-1})$ , which can generate parameters  $\mathbf{x}_{l-1}$  to evaluate the likelihood function for the coarse-scale posterior approximation. Note that the training of the MDGM in the  $l$ -th scale only uses  $\mathbf{x}_l$  and  $\mathbf{x}_{l-1}$  rather than the latent variables. Message passing across scales depends on the encoder network  $\pi_{\theta_{l-1}}(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})$ . In order to generate  $\mathbf{x}_l$  for the fine-scale inference, we need to use  $\{\mathbf{x}_{l-1}^{(i)}\}_{i=n_{l-1}}^{N_{l-1}}$  as the approximate  $(l - 1)$ -th scale posterior result and then encode them into the latent variables  $\mathbf{z}_{l-1}$ , rather than directly using the samples  $\{\mathbf{z}_{l-1}^{(i)}\}_{i=n_{l-1}}^{N_{l-1}}$ .*

**Proposition 1.** *For any  $2 \leq l \leq L$ , the acceptance ratio  $\alpha(\mathbf{z}'_l, \mathbf{z}_l^{(j)})$  for the  $(j + 1)$ -th state in the  $l$ -th scale MH-MDGM algorithm with the factorized proposal is*

$$\alpha(\mathbf{z}'_l, \mathbf{z}_l^{(j)}) = \min \left( 1, \frac{\pi(\mathbf{z}'_l|\mathcal{D}_{obs})\pi_q(\mathbf{z}_{l-1}^{(j)}|\mathbf{z}'_{l-1})\pi_q(\mathbf{z}_l^{*(j)}|\mathbf{z}'_l)}{\pi(\mathbf{z}_l^{(j)}|\mathcal{D}_{obs})\pi_q(\mathbf{z}'_{l-1}|\mathbf{z}_{l-1}^{(j)})\pi_q(\mathbf{z}'_l|\mathbf{z}_l^{*(j)})} \right), \quad (29)$$

which satisfies the detailed balance condition.

*Proof.* The transition kernel  $K(\mathbf{z}'_l, \mathbf{z}_l^{(j)})$  with the proposal distribution  $\pi_q(\mathbf{z}'_l|\mathbf{z}_l^{(j)})$  in the MH algorithm is:

$$K(\mathbf{z}'_l, \mathbf{z}_l^{(j)}) = \pi_q(\mathbf{z}'_l|\mathbf{z}_l^{(j)})\alpha(\mathbf{z}'_l, \mathbf{z}_l^{(j)}) + \delta(\mathbf{z}'_l, \mathbf{z}_l^{(j)}) \int \pi_q(\mathbf{z}_l|\mathbf{z}_l^{(j)})(1 - \alpha(\mathbf{z}_l, \mathbf{z}_l^{(j)}))d\mathbf{z}_l, \quad (30)$$

where  $\delta(\mathbf{z}'_l, \mathbf{z}_l^{(j)})$  is the Dirac delta function. When  $\mathbf{z}'_l = \mathbf{z}_l^{(j)}$ , it is obvious that the below

detailed balance condition is satisfied:

$$\pi(\mathbf{z}_l^{(j)}|\mathcal{D}_{obs})K(\mathbf{z}'_l, \mathbf{z}_l^{(j)}) = \pi(\mathbf{z}'_l|\mathcal{D}_{obs})K(\mathbf{z}_l^{(j)}, \mathbf{z}'_l). \quad (31)$$

When  $\mathbf{z}'_l \neq \mathbf{z}_l^{(j)}$ ,  $\delta(\mathbf{z}'_l, \mathbf{z}_l^{(j)}) = 0$ , we obtain  $K(\mathbf{z}'_l, \mathbf{z}_l^{(j)}) = \pi_q(\mathbf{z}'_l|\mathbf{z}_l^{(j)})\alpha(\mathbf{z}'_l, \mathbf{z}_l^{(j)})$ . Based on Eqs. (27) and (28), the proposal distribution can be written as  $\pi_q(\mathbf{z}'_l|\mathbf{z}_l^{(j)}) = \pi_q(\mathbf{z}'_{l-1}|\mathbf{z}_{l-1}^{(j)})\pi_q(\mathbf{z}'_l|\mathbf{z}'_{l-1})$ .

Then we can write the following:

$$\begin{aligned} & \pi(\mathbf{z}_l^{(j)}|\mathcal{D}_{obs})K(\mathbf{z}'_l, \mathbf{z}_l^{(j)}) \\ = & \pi(\mathbf{z}_l^{(j)}|\mathcal{D}_{obs})\pi_q(\mathbf{z}'_{l-1}|\mathbf{z}_{l-1}^{(j)})\pi_q(\mathbf{z}'_l|\mathbf{z}'_{l-1}) \min \left( 1, \frac{\pi(\mathbf{z}'_l|\mathcal{D}_{obs})\pi_q(\mathbf{z}_{l-1}^{(j)}|\mathbf{z}'_{l-1})\pi_q(\mathbf{z}_l^{*(j)}|\mathbf{z}'_l)}{\pi(\mathbf{z}_l^{(j)}|\mathcal{D}_{obs})\pi_q(\mathbf{z}'_{l-1}|\mathbf{z}_{l-1}^{(j)})\pi_q(\mathbf{z}'_l|\mathbf{z}'_{l-1})} \right) \\ = & \min \left( \pi(\mathbf{z}_l^{(j)}|\mathcal{D}_{obs})\pi_q(\mathbf{z}'_{l-1}|\mathbf{z}_{l-1}^{(j)})\pi_q(\mathbf{z}'_l|\mathbf{z}'_{l-1}), \pi(\mathbf{z}'_l|\mathcal{D}_{obs})\pi_q(\mathbf{z}_{l-1}^{(j)}|\mathbf{z}'_{l-1})\pi_q(\mathbf{z}_l^{*(j)}|\mathbf{z}'_l) \right) \\ = & \pi(\mathbf{z}'_l|\mathcal{D}_{obs})\pi_q(\mathbf{z}'_{l-1}|\mathbf{z}_{l-1}^{(j)})\pi_q(\mathbf{z}'_l|\mathbf{z}'_{l-1}) \min \left( 1, \frac{\pi(\mathbf{z}_l^{(j)}|\mathcal{D}_{obs})\pi_q(\mathbf{z}_{l-1}^{(j)}|\mathbf{z}'_{l-1})\pi_q(\mathbf{z}_l^{*(j)}|\mathbf{z}'_l)}{\pi(\mathbf{z}'_l|\mathcal{D}_{obs})\pi_q(\mathbf{z}'_{l-1}|\mathbf{z}_{l-1}^{(j)})\pi_q(\mathbf{z}'_l|\mathbf{z}'_{l-1})} \right) \\ = & \pi(\mathbf{z}'_l|\mathcal{D}_{obs})K(\mathbf{z}_l^{(j)}, \mathbf{z}'_l), \end{aligned} \quad (32)$$

and the detailed balance condition is satisfied.  $\square$

#### 4. Numerical Examples

The code and data for reproducing all examples reported in this paper can be found at <https://github.com/zabaras/MH-MDGM>. In this section, we discuss and compare the performance of the proposed multiscale method with that of the single-scale inference method. Our focus is on the estimation of the permeability field in a single phase, steady-state Darcy flow. For any permeability field  $\mathbf{K}$  on a 2D unit square domain  $\mathcal{S} = [0, 1]^2$ , the pressure field  $p$  and velocity field  $\mathbf{v}$  are governed by the following equations:

$$\mathbf{v}(\mathbf{s}) = -\mathbf{K}(\mathbf{s})\nabla p(\mathbf{s}), \quad \mathbf{s} \in \mathcal{S}, \quad (33)$$

$$\nabla \cdot \mathbf{v}(\mathbf{s}) = f(\mathbf{s}), \quad \mathbf{s} \in \mathcal{S}, \quad (34)$$

with boundary conditions

$$\mathbf{v}(\mathbf{s}) \cdot \hat{\mathbf{n}} = 0, \quad \mathbf{s} \in \Gamma_N, \quad (35)$$

$$p(\mathbf{s}) = 0, \quad \mathbf{s} \in \Gamma_D, \quad (36)$$

---

**Algorithm 4** The Metropolis-Hastings with Multiscale Deep Generative Model (MH-MDGM) Algorithm for the  $l$ -th scale posterior distribution approximation

---

**Input:** unnormalized distribution  $\pi(\mathbf{z}_l|\mathcal{D}_{obs})$ , proposal distribution  $\pi_q(\cdot|\mathbf{z}_{l-1})$  and  $\pi_q(\cdot|\mathbf{z}_l^*)$ , decoder model  $\mu_{\theta_l}(\mathbf{z}_l)$ , recognition model  $\pi_{\theta_{l-1}}(\mathbf{z}_{l-1}|\mathbf{x}_{l-1})$ ,  $(l-1)$ -th scale posterior samples  $\{\mathbf{x}_{l-1}^{(j)}\}_{j=n_{l-1}}^{N_{l-1}}$ , iteration number  $N_l$ , burn-in length  $n_l$ .

- 1: Compute the mean of  $(l-1)$ -th scale posterior estimation  $\bar{\mathbf{x}}_{l-1}$
- 2: Initialize  $\mathbf{z}_{l-1}^{(1)}, \mathbf{z}_{l-1}^{(1)} = \arg \max \pi_{\theta_{l-1}}(\mathbf{z}_{l-1}|\bar{\mathbf{x}}_{l-1})$ .
- 3: Initialize  $\mathbf{z}_l^{*(1)}, \mathbf{z}_l^{*(1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ .
- 4: **for**  $j = 1 : N$  **do**
- 5:     Draw  $\mathbf{z}'_{l-1} \sim \pi_q(\cdot|\mathbf{z}_{l-1}^{(j)})$ .
- 6:     Draw  $\mathbf{z}_l^{*'} \sim \pi_q(\cdot|\mathbf{z}_l^{*(j)})$ , then  $\mathbf{z}_l' = (\mathbf{z}'_{l-1}, \mathbf{z}_l^{*'})$
- 7:     Compute the spatially-varying parameter  $\mathbf{x}'_l = \mu_{\theta_l}(\mathbf{z}_l')$  by the decoder model
- 8:     Compute the likelihood function by solving the forward model  $\mathcal{F}_l(\mathbf{x}'_l)$
- 9:     Compute the acceptance ratio

$$\alpha = \min \left( 1, \frac{\pi(\mathbf{z}'_l|\mathcal{D}_{obs})\pi_q(\mathbf{z}_{l-1}^{(j)}|\mathbf{z}'_{l-1})\pi_q(\mathbf{z}_l^{*(j)}|\mathbf{z}_l^{*'})}{\pi(\mathbf{z}_l^{(j)}|\mathcal{D}_{obs})\pi_q(\mathbf{z}'_{l-1}|\mathbf{z}_{l-1}^{(j)})\pi_q(\mathbf{z}_l^{*'}|\mathbf{z}_l^{*(j)})} \right)$$

- 10:     Draw  $\rho$  from uniform distribution  $\mathcal{U}[0, 1]$
- 11:     **if**  $\rho < \alpha$  **then**
- 12:         Let  $\mathbf{z}_{l-1}^{(j+1)} = \mathbf{z}'_{l-1}$  and  $\mathbf{z}_l^{*(j+1)} = \mathbf{z}_l^{*'}$ ,  $\mathbf{x}_l^{(j+1)} = \mathbf{x}'_l$
- 13:     **else**
- 14:         Let  $\mathbf{z}_{l-1}^{(j+1)} = \mathbf{z}_{l-1}^{(j)}$  and  $\mathbf{z}_l^{*(j+1)} = \mathbf{z}_l^{*(j)}$ ,  $\mathbf{x}_l^{(j+1)} = \mathbf{x}_l^{(j)}$
- 15:     **end if**
- 16: **end for**

**Output:** posterior samples  $\{\mathbf{x}_l^{(i)}\}_{i=n_l}^{N_l}$

---

where  $\hat{\mathbf{n}}$  is the unit normal vector to the Neumann boundary  $\Gamma_N$ . The Neumann boundary  $\Gamma_N$  consists of the top and bottom boundaries and the Dirichlet boundary  $\Gamma_D$  consists of the right and left walls. We consider a source term  $f(\mathbf{s}) = 3$ . For the forward model  $\mathcal{F}$ , given any permeability field  $\mathbf{K}$ , Eqs. (33) and (34) are solved by a mixed finite element formulation implemented in FEniCS [63] with third-order Raviart-Thomas elements for the velocity  $\mathbf{v}$ , and fourth-order discontinuous elements for the pressure  $p$ . The computational cost for solving this forward model with different discretizations is reported in Table 1.

Table 1: Computational cost of solving the forward model in Eqs. (33) and (34) with different discretizations.

Discretization	Output dimension	Seconds	Normalized time
$64 \times 64$	$3 \times 4096$	3.10	23.85
$32 \times 32$	$3 \times 1024$	0.67	5.15
$16 \times 16$	$3 \times 256$	0.13	1.0

To enforce the non-negative permeability constraint, we consider the log-permeability as the main parameter of interest in the inverse problem with  $\mathbf{x} = \log(\mathbf{K})$ . The inverse problem for the above model is to infer the unknown log-permeability field given noisy pressure measurements at some sensor locations. In this paper, the exact log-permeability field  $\mathbf{x}_{exact}$  is defined in a  $64 \times 64$  uniform grid. The generated data by the generative model are usually smooth and blurry compared to the original data due to information compression. We consider 64 pressure observations that are uniformly located in  $[0.0625 + 0.125i, 0.0625 + 0.125i], i = 0, 1, 2, \dots, 7$ . A 5% independent additive Gaussian random noise is considered on these 64 pressure observations which are obtained by the above forward model given the exact log-permeability field.

#### 4.1. Test problem 1: Gaussian Random Field (GRF)

For GRF based log-permeability data, one can adopt the KLE method for a reduced-order representation. Bayesian inference often works well on such low-dimensional inversion tasks. However, the KLE expansion requires a-priori knowledge of the length scale in the GRF covariance function. This information is of course not available in most practical

applications. Data-driven methods like DGM do not require such restrictive assumptions. The only prior knowledge available is based on the given training dataset.

In this section, we apply the proposed method on a two-scales scenario and provide a comparison with the reference case of the single-scale method. In the two-scales test case, we consider the  $16 \times 16$  uniform grid as the coarsest-scale and the  $64 \times 64$  uniform grid as the finest-scale. The finest-scale is the same with the scale of the exact log-permeability field. Using the proposed method, the generative models are trained on  $16 \times 16$  and  $64 \times 64$  ( $16 - 64$ ) resolutions. We use the pre-trained MDGM with MCMC for Bayesian inference from coarse to fine, and compare the results with the single scale ( $64 \times 64$  grid) method with focus of our investigation on computational efficiency, accuracy and convergence.

#### 4.1.1. Multiscale dataset

As discussed in Section 3.4, we adopt an unsupervised learning method for parameter representation before addressing the solution of the inverse problem. In particular, one can train a generative model for the sampling of log-permeability fields. For such an unsupervised learning problem, we only need to obtain the dataset  $\{\mathbf{x}^{(i)}\}_{i=1}^N$  based on historical sample information or the underlying prior distribution  $\pi(\mathbf{x})$ . In this example, we assume that the log-permeability field is a Gaussian random field, i.e.  $\mathbf{x}(\mathbf{s}) \sim \mathcal{GP}(m(\mathbf{s}), k(\mathbf{s}_1, \mathbf{s}_2))$ , where  $m(\mathbf{s})$  and  $k(\mathbf{s}_1, \mathbf{s}_2)$  are the mean and covariance functions, respectively.  $\mathbf{s}_1 = (x_1, y_1)$  and  $\mathbf{s}_2 = (x_2, y_2)$  denote two arbitrary spatial locations. The covariance function  $k(\mathbf{s}_1, \mathbf{s}_2)$  in this paper is taken as:

$$k(\mathbf{s}_1, \mathbf{s}_2) = \sigma_{\log(K)}^2 \exp\left(-\sqrt{\left(\frac{x_1 - x_2}{l_1}\right)^2 + \left(\frac{y_1 - y_2}{l_2}\right)^2}\right), \quad (37)$$

where  $\sigma_{\log(K)}^2$  is the variance, and  $l_1$  and  $l_2$  are the length scales along the  $x$  and  $y$  axes, respectively. In this example, we set  $m(\mathbf{s}) = 1$  and  $\sigma_{\log(K)}^2 = 0.5$ . As discussed earlier, it is hard to tackle the multiple length scales setup using the KLE method, while the deep generative model has a distinct superiority on the prior assumption since the dataset embodies these assumptions or information in a natural way. We assume that the length scales are not fixed in the prior distribution. Let the length scales be  $l_1 = l_2 = 0.2 + 0.01i, i = 0, 1, 2, \dots, 9$ . The finest-scale parameter  $\mathbf{x}_2$  is uniformly discretized into an  $H \times W = 64 \times 64$  grid. For

each length scale, we generate 2500 samples for the training dataset  $\{\mathbf{x}_2^{(i)}\}_{i=1}^N$ , where  $N$  is 25000. The exact log-permeability field  $\mathbf{x}_{exact}$  for the Bayesian inversion has never been seen in the training procedure. We set the  $16 \times 16$  grid as the coarsest-scale in the parameter estimation. We adopt the upscaling operator in Eq. (9) with  $n_e = 16$  for each realization in  $\{\mathbf{x}_2^{(i)}\}_{i=1}^N$ , and then obtain the  $16 \times 16$  grid dataset  $\{\mathbf{x}_1^{(i)}\}_{i=1}^N$ . This constitutes the training dataset for the generative model in the coarse-scale.

#### 4.1.2. Training and results of the MDGM

With the training datasets available on each scale, we can train the generative model from coarse- to fine-scale. The coarsest-scale generative model only involves the single-scale parameters  $\mathbf{x}_1$ . The training procedure of the single-scale generative model is illustrated in Algorithm 1. The schematic network architecture is shown in Fig. 4. The detailed encoder and decoder networks can be seen in Appendix A. All of the encoder and decoder neural networks in this paper are trained on a NVIDIA GeForce GTX 1080 Ti GPU card. The loss function is defined in Eq. (15) for training the single-scale generative model. For all training procedures in this paper, we set the batch size  $n$  in the loss function to 64, and the sampling size  $m$  to 1. For the optimization of all neural networks, the Adam optimizer [64] was adopted with a learning rate of  $2 \times 10^{-4}$ . The above setup is kept consistent for all training models in this paper. The only differences are the training epochs and hyperparameters  $\tilde{\beta}$  in the loss functions, which will be specified in the remaining cases. All the models in the GRF case are trained with 30 epoches, and the hyperparameter  $\tilde{\beta}$  is 0.5. It takes about 24 minutes and 73 minutes for the training of a single-scale generative model on  $16 \times 16$  and  $64 \times 64$  resolutions, respectively.

For the  $64 \times 64$  single-scale generative model, we use the pre-trained model  $p_{\theta}(\mathbf{x}|\mathbf{z})$  for MCMC exploration to estimate directly the parameter in the  $64 \times 64$  resolution with  $\mathbf{z} \in \mathbb{R}^{256}$ . For the  $16 \times 16$  single-scale generative model, the latent variables  $\mathbf{z}_1 \in \mathbb{R}^{16}$  serve as the bridge to the fine-scale estimation. We use the pre-trained model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  for MCMC exploration of the posterior in the coarse-scale. For the training of the MDGM, the pre-trained model  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$  is part of the encoder network as illustrated in Fig. 5, and an input to Algorithm 2. The loss function in Eq. (21) is used for training the models  $q_{\phi_2^*}(\mathbf{z}_2^*|\mathbf{x}_2)$  and  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*)$ ,

where  $\mathbf{z}_2^* \in \mathbb{R}^{256}$ . The training time for the models  $q_{\phi_2^*}(\mathbf{z}_2^*|\mathbf{x}_2)$  and  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*)$  is about 78 minutes.

Once the generative models are obtained in the different scales, the parameters can be generated in each scale by first sampling the corresponding latent variable, and then by decoding it using the corresponding generative model. Here, we illustrate the MDGM results in Fig. 6. For the pre-trained generative model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$ , we can sample the latent variables  $\mathbf{z}_1$  from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ , and use these latent variables as input to the model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$ . The first row of Fig. 6 shows 4 realizations with  $16 \times 16$  resolution that are output of this decoder network. It is apparent that all of these images have distinct features in the spatial distribution of high- and low-value regions. However, they are very smooth so that one cannot capture any local information. Fortunately, this is a good choice to highlight the obvious global features, which are of great importance in Bayesian inversion. It is a fundamental and necessary requirement for spatially-varying parameter estimates to be consistent with the exact parameter in capturing global features.

The performance of the  $64 \times 64$  resolution generative model  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*)$  in this two-scales MDGM is shown in the second row of Fig. 6. The latent variable  $\mathbf{z}_1$  is the mean of the distribution  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$ , i.e.  $\mathbf{z}_1 = \arg \max q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$ , where  $\mathbf{x}_1 \in \mathbb{R}^{16 \times 16}$  is the first image in this row. The latent variable  $\mathbf{z}_2^*$  is sampled from the Gaussian distribution  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ . We observe that that the three samples generated by the model  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*)$  have some particularities. Unlike samples in the first row, they keep similar spatial distribution of high- and low-value regions that inherit from the first  $16 \times 16$  resolution image  $\mathbf{x}_1$  while they are refined locally in a diverse manner. The model  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$  acts as a messenger in the MDGM. The information of low-resolution  $\mathbf{x}_1$  that captures global features is encoded by  $\mathbf{z}_1$ , which together with the random variables  $\mathbf{z}_2^*$  that is used to supplement local details are decoded by the generative model  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*)$ . This demonstrates that these two latent variables have different missions in the generative model, in particular  $\mathbf{z}_1$  plays an important role in the global feature generation, while  $\mathbf{z}_2^*$  contributes to local details. This disentangled representation [65, 53, 66, 67] for local and global features assists in an accurate and efficient Bayesian multiscale estimation.

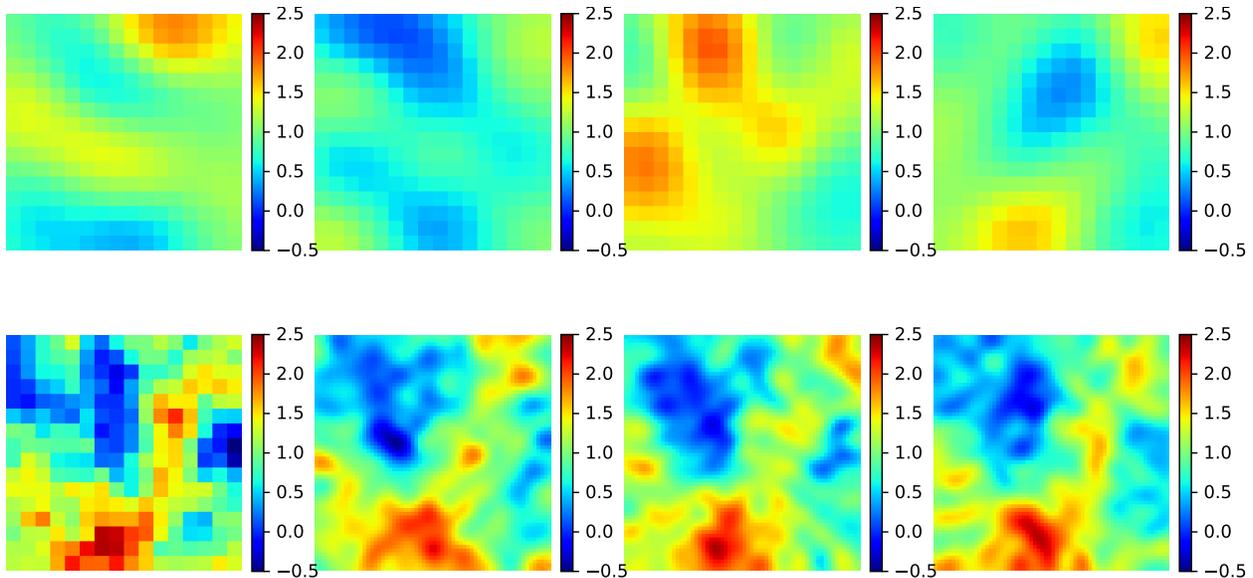


Figure 6: First row:  $16 \times 16$  resolution random samples using the generative model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$ , where  $\mathbf{z}_1 \in \mathbb{R}^{16}$  is sampled from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ . Second row:  $64 \times 64$  resolution random samples using the generative model  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_2)$ , where  $\mathbf{z}_2 = (\mathbf{z}_1, \mathbf{z}_2^*)$ . We let  $\mathbf{z}_1 = \arg \max q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$ , where  $\mathbf{x}_1$  is the first image in this row and  $\mathbf{z}_2^* \in \mathbb{R}^{256}$  is sampled from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ .

#### 4.1.3. The inversion results and discussion

The reference experiment considered is the single-scale method. We use Algorithm 3 with the pre-trained model  $\mathbf{x} = \mu_{\theta}(\mathbf{z})$  to estimate directly  $\pi(\mathbf{z}|\mathcal{D}_{obs})$ , where  $\mathbf{x} \in \mathbb{R}^{64 \times 64}$  and  $\mathbf{z} \in \mathbb{R}^{256}$ . All the implementations using MCMC treat the latent variables as random variables, and use the pre-trained model to generate  $\mathbf{x}$  for evaluating the likelihood function. The proposal distribution for the random variables applied in this paper is preconditioned Crank-Nicolson (pCN) that is defined below:

$$\mathbf{z}' = \sqrt{1 - \gamma^2} \mathbf{z} + \gamma \zeta, \quad (38)$$

where  $\mathbf{z}$  and  $\mathbf{z}'$  are current and proposed next states, respectively, and  $\zeta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . The step size of the random movement from the current state to a new position is controlled by the free parameter  $\gamma$ . We set  $\gamma$  be 0.08 for the first 50% and 0.04 for the last 50% states in the Markov chain for all implementations using Algorithm 3. We run 10000 iterations in MCMC to ensure its convergence. For all implementations of the MCMC algorithm, we collected the last 2000 states as the posterior samples. Fig. 7 shows the estimation results

using the reference method. It can be seen that the integration of the deep generative model with MCMC leads to a reasonable estimation of the spatially varying parameter.

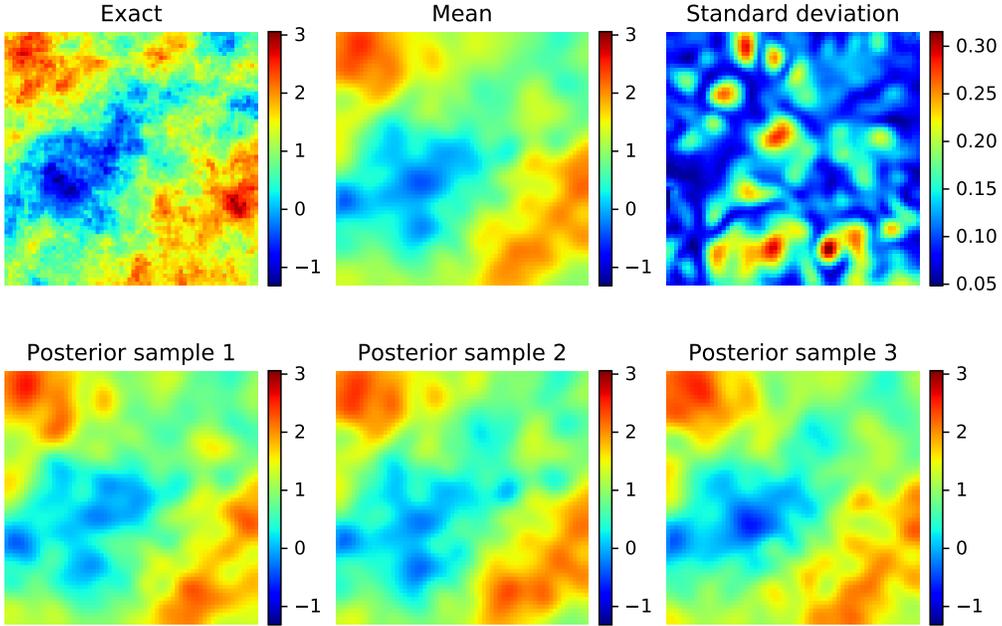


Figure 7: The reference single-scale estimation result with the desired  $64 \times 64$  grid. The mean and standard deviation of the estimated posterior distribution that are computed using the posterior samples are shown in the first row. Realizations from the posterior are shown in the second row. Inference with respect to  $\mathbf{z} \in \mathbb{R}^{256}$  is performed using Algorithm 3.

The multiscale parameter estimation is implemented next from coarse- to fine-scale. We use Algorithm 3 with pre-trained model  $\mathbf{x}_1 = \mu_{\theta_1}(\mathbf{z}_1)$  to estimate  $\pi(\mathbf{z}_1 | \mathcal{D}_{obs})$ , where  $\mathbf{x}_1 \in \mathbb{R}^{16 \times 16}$  and  $\mathbf{z}_1 \in \mathbb{R}^{16}$ . A Markov chain with length 7000 is constructed for  $\mathbf{z}_1$  to estimate the coarse-scale Gaussian log-permeability field. This result explores various global patterns of the log-permeability. Fig. 8 shows that the estimation result is as expected. It is clear that the global spatial distribution of low- or high-values is located at three different regions, consistent with the exact log-permeability field. The estimated log-permeability with  $16 \times 16$  grid captures the important features efficiently since it uses the forward model with only a  $16 \times 16$  grid. However, this occurs at the sacrifice of local information. In addition, the coarse-scale forward model introduces computational error. To resolve these issues, one needs to infer the fine-scale parameter using the high-resolution generative model and the corresponding precise forward solver.

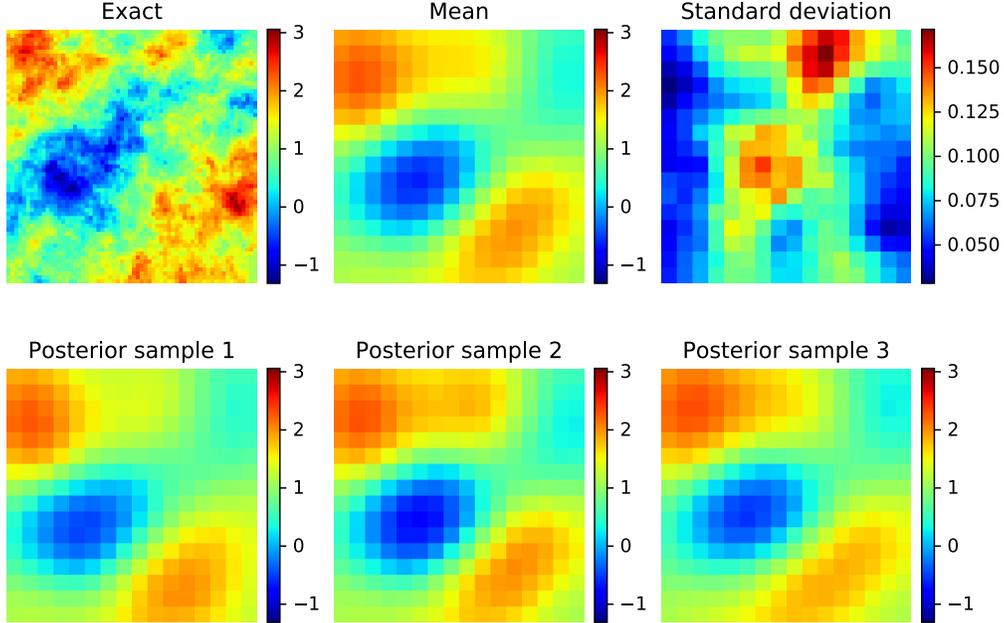


Figure 8: The two-scale estimation result with  $16 \times 16$  coarse-grid and  $64 \times 64$  fine-grid. The coarse-scale results are shown here. Inference with respect to  $\mathbf{z}_1 \in \mathbb{R}^{16}$  is performed using Algorithm 3.

To correct and refine the parameter estimation, we use Algorithm 4 with the pre-trained model  $\mathbf{x}_2 = \mu_{\theta_2}(\mathbf{z}_2)$  to estimate  $\pi(\mathbf{z}_2 | \mathcal{D}_{obs})$ , where  $\mathbf{x}_2 \in \mathbb{R}^{64 \times 64}$ ,  $\mathbf{z}_2 = (\mathbf{z}_1, \mathbf{z}_2^*)$ , and  $\mathbf{z}_1 \in \mathbb{R}^{16}$ , and  $\mathbf{z}_2^* \in \mathbb{R}^{256}$ . Since we have obtained the posterior samples in the coarse-scale, the estimation in the fine-scale takes advantage of the coarse estimation. We need to assign two proposal distributions for  $\mathbf{z}_1$  and  $\mathbf{z}_2^*$ . The two proposal distributions we used in this paper for Algorithm 4 are the pCN in Eq. (38) with different step sizes. The fixed step size  $\gamma$  for the low-dimensional latent variable  $\mathbf{z}_{l-1}$  is 0.01, while the adaptive step size  $\gamma$  for the high-dimensional latent variable  $\mathbf{z}_l^*$  is 0.08 for the first 50% and 0.04 for the last 50% of the states in the Markov chain. The results in Fig. 9 indicate that the estimated parameter with  $64 \times 64$  grid using the proposed method has even better performance than the reference single-scale results in Fig. 7. The details of the low-value region are closer to the exact log-permeability field.

To compare the estimation result with the exact field, we provide an illustration in Fig. 10 that shows the values of the log-parameter field from the left top corner to the right bottom corner. We notice that the true values curve (black line) is very sharp while the posterior mean is much smoother. Compared to other deep generative models [17, 42],

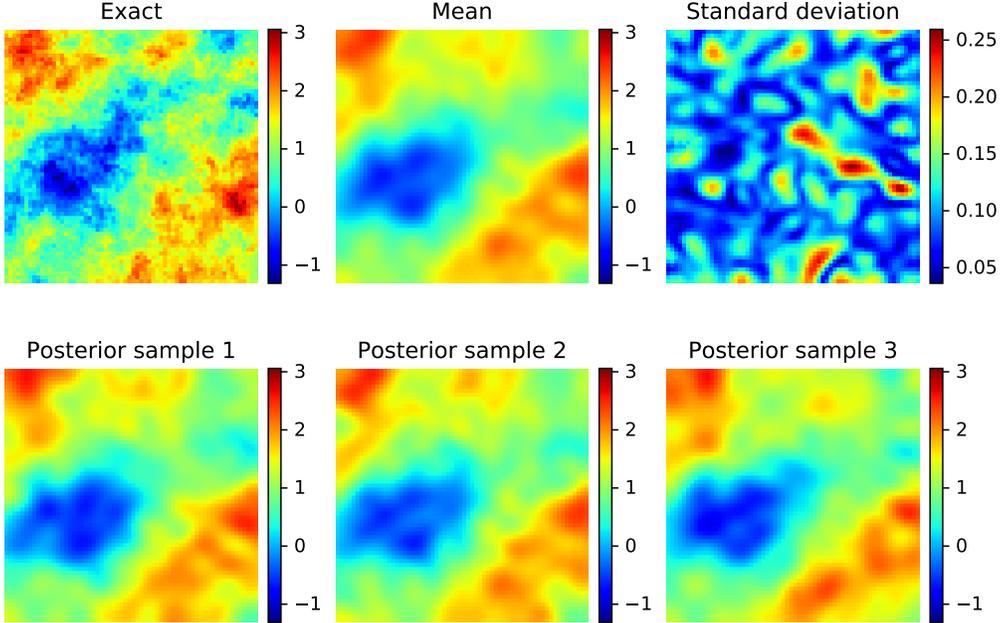


Figure 9: The two-scale estimation result with  $16 \times 16$  coarse-grid and  $64 \times 64$  fine-grid. The fine-scale results are shown here. Inference with respect to two latent variables  $\mathbf{z}_1 \in \mathbb{R}^{16}$  and  $\mathbf{z}_2^* \in \mathbb{R}^{256}$  is performed using Algorithm 4.

the shortcoming of VAE [18] is that it generates blurry samples since the bottleneck layer captures a compressed latent encoding. However, we note that the result obtained from the proposed method is better than the reference method as its mean is much closer to the exact solution.

The main cost of the Bayesian inference using MCMC comes from the forward model evaluation. The forward model’s computational cost in Table 2 suggests that the single-scale method takes about 1.9 times the computational cost of the proposed two-scale method. The acceptance rate of MCMC for all implementations is shown in Table 3. We can note that the proposed method has a much higher acceptance rate than the single-scale method in the desired scale with  $64 \times 64$  grid. The reason is that the inference for the fine-scale parameter is facilitated from the designed latent space of the MDGM that has two latent variables with different influences in the fine-scale parameter generation. The low-dimensional latent variable impacts the global features observed in the fine-scale. We assigned a small step size for its pCN proposal distribution since the global features that are inherited from the coarse-scale estimation only need slight adaption, while the high-dimensional latent variable

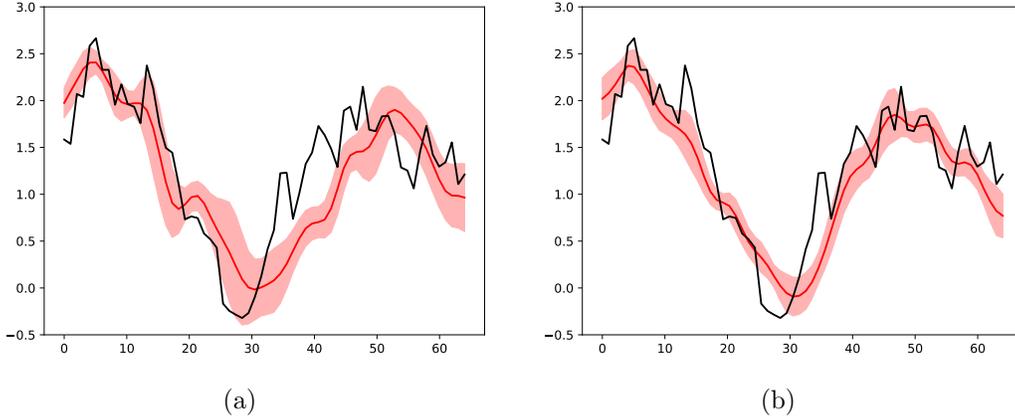


Figure 10: The log-permeability from the left top corner to the right bottom corner on a  $64 \times 64$  grid. The results from left to right are obtained by (a) reference (64) and (b) two scales (16 – 64) estimation, respectively. The black and red lines show the true and the posterior mean log-permeability, respectively. The shaded region shows values within two standard deviations of the mean.

estimated for the local refinement is assigned with a big step size. The reference method often rejects the proposed samples and gets trapped in local modes. However, in the fine-scale estimation in the proposed method, the main changes in the local features will lead to small changes in the likelihood function so that most of the proposed samples are accepted.

Table 2: The iterations (its) and approximated cpu time in seconds(s) for solving the forward model in different experiments for the GRF test problem.

Experiment	$16 \times 16$	$64 \times 64$	Total
one scale (64)*	-	10000 its	10000 its
	-	31000 s	31000 s
two scales (16 – 64)	7000 its	5000 its	12000 its
	910	15500 s	16410 s

The convergence of the Markov chain used to estimate parameters with a  $64 \times 64$  grid is of concern since the forward evaluation on such a scale is very expensive. To assess the convergence in the desired scale with a  $64 \times 64$  grid, we employ three metrics using variables available in the iterative process. Since the observation data (noisy pressure measurements) is the only basis for parameter estimation, we compute the misfit between the observation

Table 3: The acceptance ratio of the MH algorithm in different experiments for the GRF test problem.

Experiment	$16 \times 16$	$64 \times 64$
one scale (64)	-	30.6%
two scales (16 – 64)	37.5%	67.8%

data and its corresponding prediction using the sum of squared residuals of observations ( $SSR_{obs}$ ) as the iteration proceeds:

$$SSR_{obs} = \sum_{i=1}^{N_{obs}} \left( \mathcal{F}(\mu(\mathbf{z}))^{(i)} - \mathcal{D}_{obs}^{(i)} \right)^2, \quad (39)$$

where  $N_{obs} = 64$  in this paper, and  $\mathcal{F}(\mu(\mathbf{z}))^{(i)}$  and  $\mathcal{D}_{obs}^{(i)}$  are the  $i$ -th predicted pressure value and its corresponding true observation, respectively. Ideally, we expect that the value of  $SSR_{obs}$  is close to 0, which suggests there is no discrepancy between the predictions and observations. But this cannot be realized even using the exact input for  $\mathcal{F}$  due to measurements noise. To remove the impact of noise, we use an enhanced metric i.e. the normalized sum of squared weighted residual ( $NSSWR$ ) [16, 68]:

$$NSSWR = \frac{1}{SSWR_{ref}} \sum_{i=1}^{N_{obs}} \left( \frac{\mathcal{F}(\mu(\mathbf{z}))^{(i)} - \mathcal{D}_{obs}^{(i)}}{\sigma_n^{(i)}} \right)^2, \quad (40)$$

where  $SSWR_{ref} = \sum_{i=1}^{N_{obs}} \left( \frac{\mathcal{F}(\mathbf{x}_{exact})^{(i)} - \mathcal{D}_{obs}^{(i)}}{\sigma_n^{(i)}} \right)^2$ ,  $\mathbf{x}_{exact}$  is the exact log-permeability field, and  $\sigma_n^{(i)}$  is the standard deviation of Gaussian random noise imposed in the  $i$ -th observation. The  $SSWR$  metric is normalized by the  $SSWR_{ref}$ . Thus, the inversion process has converged when the  $NSSWR$  value is close to 1. Since the target is to estimate the log-permeability field based on the given observations, we also consider the evaluation of the mismatch between the predicted and the exact log-permeability fields using the sum of squared residuals of parameter ( $SSR_{para}$ ) as shown below:

$$SSR_{para} = \sum_{i=1}^M \left( \mu(\mathbf{z})^{(i)} - \mathbf{x}_{exact}^{(i)} \right)^2, \quad (41)$$

where  $M = 4096$ , since the desired scale is discretized to  $64 \times 64$ , and  $\mu(\mathbf{z})^{(i)}$  and  $\mathbf{x}_{exact}^{(i)}$  are the  $i$ -th value of the predicted and the exact log-permeability fields, respectively.

Fig. 11 shows the convergence results in the GRF case. The comparison of the proposed method with the single-scale method is also given in Fig. 11. It can be seen that better performance in all evaluation metrics is produced by the proposed method. A big distinction takes place in the decrease of  $SSR_{para}$  illustrated in Fig. 11(c). We show the sampled log-permeability field with  $64 \times 64$  grid at different iterations in the Markov chain in Fig. 12. For the single-scale method, each dimension of the estimated latent variable  $\mathbf{z} \in \mathbb{R}^{256}$  has equivalent importance for the generation of the log-permeability. The random walk in such a high-dimensional space has difficulty in efficiently exploring the space and identifying a good estimate and its uncertainty. Together with a random initial state sampled from the prior distribution, the exploration takes a long time to reach the stationary distribution. The second row in Fig. 12 presents the state evolution for the two-scale method. The first-state that inherited the coarse-scale estimation captures well most of the non-local features of the exact log-permeability field. Thus with the iterations shown, we only need to correct the local features to explore the posterior distribution.

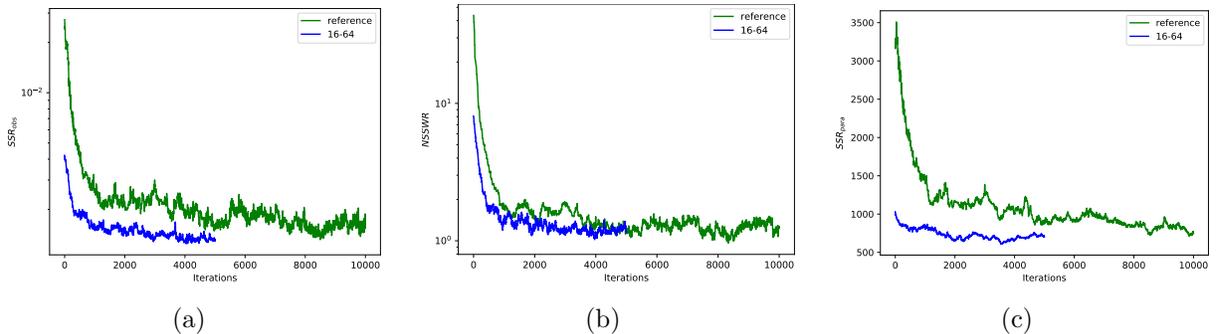


Figure 11: The convergence of the Markov chain used for the Gaussian log-permeability field estimation with a  $64 \times 64$  grid. The evaluation metrics from left to right are (a) the SSR of observable pressure values, (b) NSSWR values, and (c) the SSR of parameter field.

#### 4.2. Test problem 2: Non-Gaussian Random Field

In the second test problem, we consider a channelized log-permeability field as the exact parameter in the inversion experiment. For such a non-Gaussian permeability field is often difficult to obtain a good parameterization using conventional methods such as sparse-grid approximations [13], wavelets [14], or principal component analysis (PCA) [69, 70, 71].

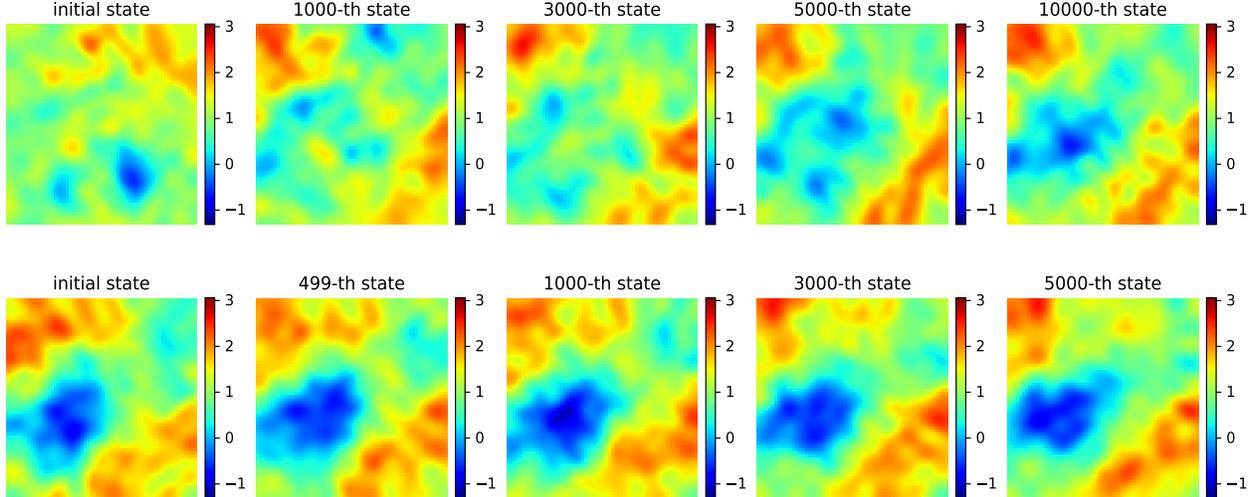


Figure 12: The states of the Gaussian log-permeability field with  $64 \times 64$  grid during iterations in the Markov chain. The results from top to bottom row are obtained from (a) reference (64) and (b) two scales (16 – 64) experiment, respectively.

Furthermore, the Bayesian inference using random walk MCMC based on recent reported learning-based parameterization methods [16, 15, 46, 22] for high-dimensional non-Gaussian parameters is not a good choice. We show the benefits of the multiscale method in the parameterization of non-Gaussian random fields and Bayesian inference. In this test example, we demonstrate the proposed method with two- and three-scales scenarios and provide comparisons with the reference case of a single-scale method.

#### 4.2.1. Multiscale dataset

Suppose that the prior information for the channel location before any measurement is from the image [22] with size of  $2500 \times 2500$  shown in Fig. 13(a). One can crop the large image with a fixed stride. With a 16 stride in the horizontal and vertical directions, we obtained 23104 training samples of size of  $64 \times 64$ . To provide sufficient data for the training of the generative model, we flip the entries in each row of the image in the left/right direction by the *fliplr* operation <sup>2</sup> implemented in Numpy package to obtain a new image, and cropped this image to obtain additional 23104 samples. A sample cropped by this procedure is illustrated in Fig. 13(b). The binary image depicts channels with white regions. We assume that the

<sup>2</sup><https://numpy.org/doc/1.18/reference/generated/numpy.fliplr.html>

channelized regions have different log-permeabilities resulting in high- and low-permeability values in white and black regions, respectively. We set the log-permeability values for each region by independently sampling from two Gaussian Random Fields (GRFs). The means of the GRFs for the high-permeability channelized regions and low-permeability regions are 4 and 0, respectively. The covariance function in Eq. (37) with length scales  $l_1$  and  $l_2$  equal to 0.3 is applied. The variance is 0.5 for both GRFs. By imposing two GRFs for the binary image samples, we can generate the training data as shown in Fig. 13(c). We assume the generated samples are i.i.d. sampled from the underlying prior distribution  $\pi(\mathbf{x})$ . We take 40000 samples from generated 46208 realizations as training data. The exact log-permeability field  $\mathbf{x}_{exact}$  for Bayesian inversion is sampled from the remaining samples, which has never been seen in the training procedure. The  $\mathbf{x}_{exact}$  we used in this test problem is the image shown in Fig. 13(c).

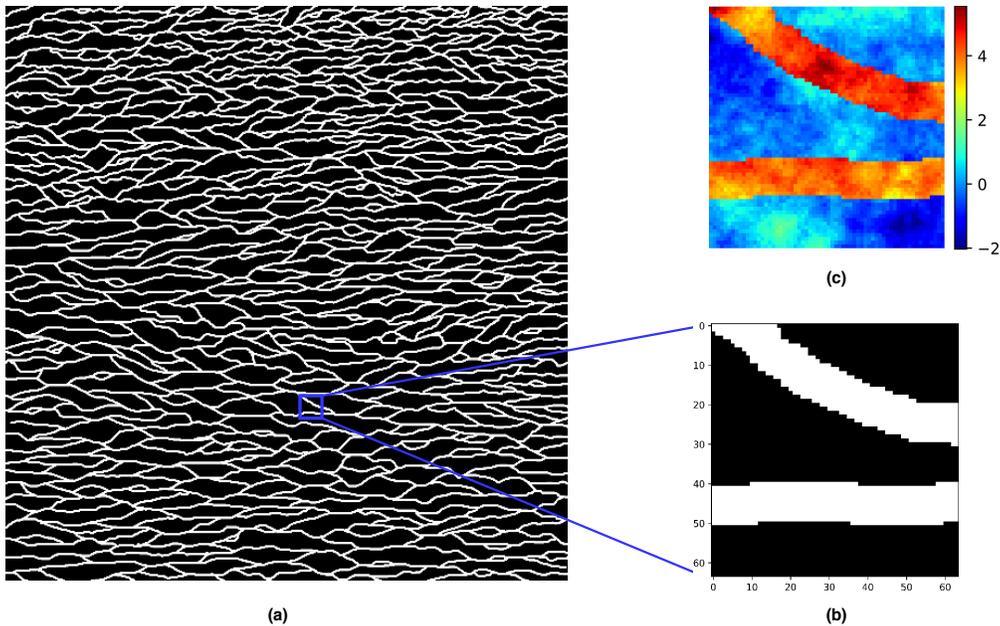


Figure 13: (a) The large image contains the prior information of the channel location (b) cropped binary image samples from the large image (c) A sample from the underlying prior distribution  $\pi(\mathbf{x})$  by assigning two different GRFs to the binary image.

The training of MDGM needs data with different discretizations. In this test example, we consider two types of MDGM, which are a two-scale model with  $16 \times 16$  grid and  $64 \times 64$  grid ( $16 - 64$ ) and a three-scale model with  $16 \times 16$  grid,  $32 \times 32$  grid and  $64 \times 64$  grid

(16 – 32 – 64). To generate these datasets, one can adopt the upscaling operator in Eq. (9) with  $n_e = 4$  over the original  $64 \times 64$  grid data, and apply it again with  $n_e = 4$  in the generated  $32 \times 32$  grid data to obtain the  $16 \times 16$  grid data. An example with different discretizations has been illustrated in Fig. 2.

#### 4.2.2. Training and results of the MDGM

Using the generated training datasets, we applied the same neural network used in the GRF case to train the MDGM. We will discuss the training procedure and performance below for the two- and three-scales cases. The reference single-scale generative model  $p_{\theta}(\mathbf{x}|\mathbf{z})$  requires 3.15 hours to train 50 epochs using the loss function in Eq. (15) and Algorithm 1, where  $\mathbf{x} \in \mathbb{R}^{64 \times 64}$ ,  $\mathbf{z} \in \mathbb{R}^{256}$ , and  $\tilde{\beta} = 0.5$ .

*Two scales (16–64).* In this example, the  $\mathbf{x}_1, \mathbf{x}_2$  denote the parameters with  $16 \times 16$  grid and  $64 \times 64$  grid, respectively. For the  $\mathbf{x}_1$ , the generative model is trained using Algorithm 1 with  $\tilde{\beta} = 1$  in Eq. (15). It takes about 37 minutes for 30 training epochs. We obtained  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  and  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$ , where  $\mathbf{x}_1 \in \mathbb{R}^{16 \times 16}$  and  $\mathbf{z}_1 \in \mathbb{R}^{16}$ . The model  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$  is the input to Algorithm 2 used for training the finer-scale generative model. We use the pre-trained model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  to reconstruct the parameters to compute the likelihood function when we use Algorithm 3 to estimate  $\pi(\mathbf{z}_1|\mathcal{D}_{obs})$ , where  $\mathbf{z}_1 \in \mathbb{R}^{16}$ . The generated samples using the model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  are shown in the first row of Fig. 14(a). As expected, these samples present the most important features i.e. the location of the channels without much local information. The finer-scale generative model  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*)$  is trained using Algorithm 2 with the loss function in Eq. (21) for the estimation refinement, where  $\tilde{\beta} = 2.5$ ,  $\mathbf{x}_2 \in \mathbb{R}^{64 \times 64}$ , and  $\mathbf{z}_2^* \in \mathbb{R}^{256}$ . Training 50 epochs takes about 3.3 hours. The performance of this is shown in the second row of Fig. 14(a), where the first image in this row is a field with a  $16 \times 16$  grid. Encoding this image into a certain  $\mathbf{z}_1$  and together with three randomly sampled latent variables  $\mathbf{z}_2^* \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  generated the last three samples in this row. We can see that the generated fine-scale samples keep similar channels with the given coarse-scale image, while their local refinement shows sufficient diversity. This indicates that the coarse-scale information can be captured by the latent variable  $\mathbf{z}_1$  using the encoder model  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$ . As before, in the finer-scale generative model, the low-dimensional latent variable  $\mathbf{z}_1$  dominates

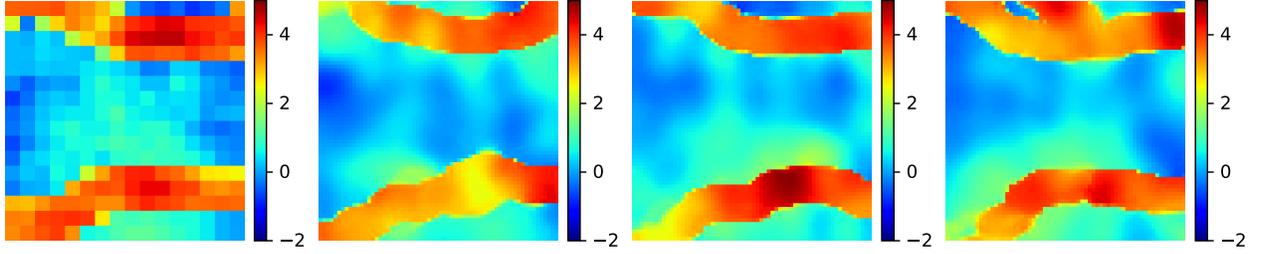
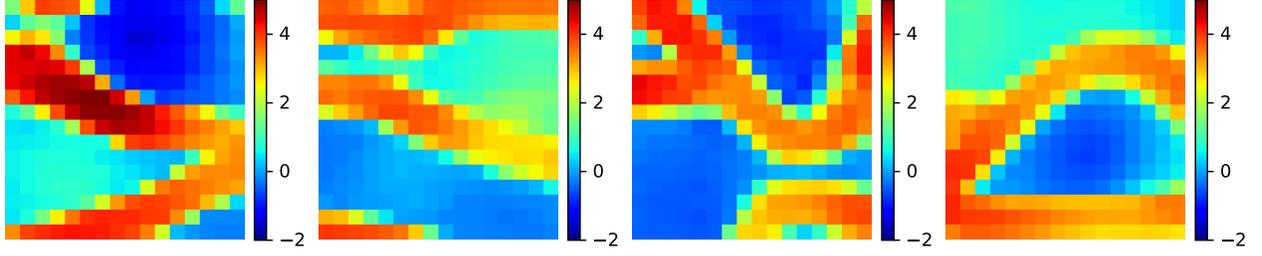
the global features (channels in this example), while the high-dimensional latent variables  $\mathbf{z}_2^*$  are used to capture local features.

*Three scales* (16 – 32 – 64). In the second example with three scales, the parameters with  $16 \times 16$ ,  $32 \times 32$  and  $64 \times 64$  grids are denoted by  $\mathbf{x}_1$ ,  $\mathbf{x}_2$ ,  $\mathbf{x}_3$ , respectively. The model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  is the same as the above two-scales example as shown in the first row of Fig. 14(a). The first row of Fig. 14(b) gives the results of the model  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*)$ , where  $\mathbf{x}_2 \in \mathbb{R}^{32 \times 32}$ ,  $\mathbf{z}_1 \in \mathbb{R}^{16}$  is encoded from the first image in this row using the model  $q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$ , and  $\mathbf{z}_2^* \in \mathbb{R}^{64}$  is sampled from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ . It takes about 1.3 hours to train 50 epochs with  $\tilde{\beta} = 0.7$  in Eq. (21). Compared with the two-scales case, one can notice that the generated samples have highly consistent channels that inherit from the first image but maintain local diversity. In order to sample the desired parameters with  $64 \times 64$  grid, we also adopt Algorithm 2 with  $\tilde{\beta} = 0.7$  in Eq. (21) to train the model  $p_{\theta_3}(\mathbf{x}_3|\mathbf{z}_2, \mathbf{z}_3^*)$ , where  $\mathbf{z}_2 \in \mathbb{R}^{80}$ ,  $\mathbf{z}_3^* \in \mathbb{R}^{256}$ , and the training time is 3.75 hours. The pre-trained model  $q_{\phi_2}(\mathbf{z}_2|\mathbf{x}_2)$  is the input to encode the given coarse-scale training data, where  $\mathbf{z}_2 = (\mathbf{z}_1, \mathbf{z}_2^*)$ . The samples using the pre-trained model  $p_{\theta_3}(\mathbf{x}_3|\mathbf{z}_2, \mathbf{z}_3^*)$  are the last three samples shown in the second row of Fig. 14(b), where  $\mathbf{z}_2$  is encoded from the first image in this row. As with the previous examples, the channels basically coincide with those in the first image. However, the difference is that the generated samples not only retain the global features of the encoded coarse-scale image but also discover local details.

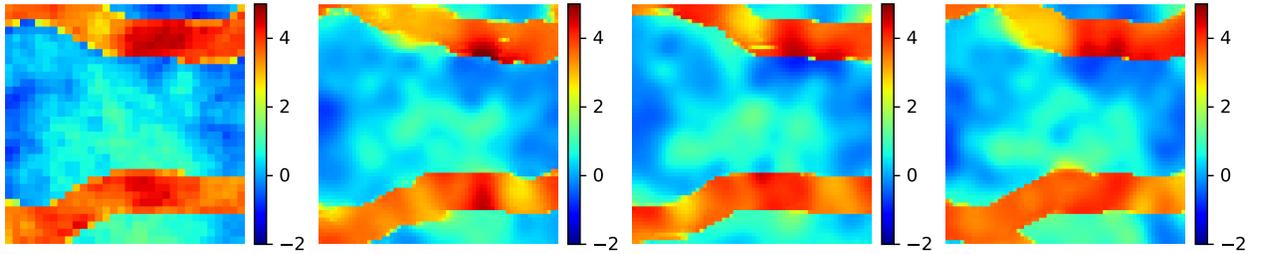
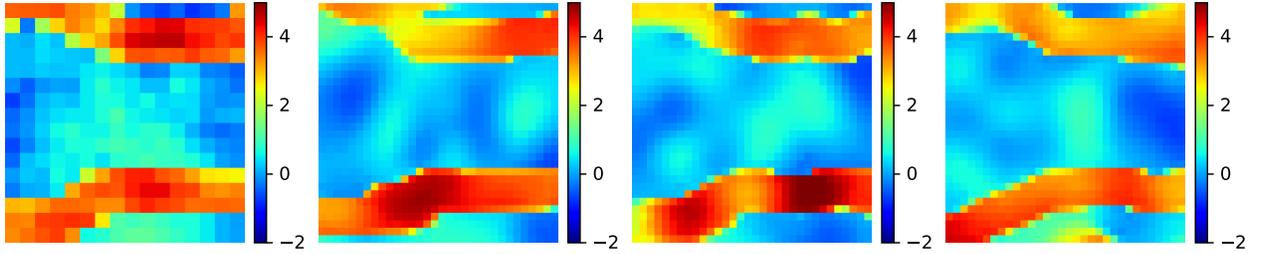
In summary, both the two- and three-scales models can generate samples with the correct spatial distribution of channels in the coarsest-scale with the  $16 \times 16$  grid. For the fine-scale parameter generation, the low-dimensional latent variables encode the information of the coarse-scale parameter and define the global features of the fine-scale parameter. The high-dimensional latent variables capture fine-details and provide diversity in the generated samples.

#### 4.2.3. The inversion results and discussion

Once all generative models are trained, we can use them in the Bayesian inversion process. To assess the efficiency and accuracy of the proposed multiscale method, we first evaluate the posterior distribution using the reference single-scale method. Algorithm 3 using the pre-trained model  $\mathbf{x} = \mu_{\theta}(\mathbf{z})$  as the input can perform the random walk in the latent space,



(a)



(b)

Figure 14: (a) The 16 – 64 MDGM. First row:  $16 \times 16$  resolution random samples using the generative model  $p_{\theta_1}(\mathbf{x}_1|\mathbf{z}_1)$  by randomly sampling  $\mathbf{z}_1 \in \mathbb{R}^{16}$  from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$  as input. Second row:  $64 \times 64$  resolution random samples using the model  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*)$ . We let  $\mathbf{z}_1 = \arg \max q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$ , and  $\mathbf{x}_1$  is the first image in this row, and  $\mathbf{z}_2 \in \mathbb{R}^{256}$  is randomly sampled from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ . (b) The 16 – 32 – 64 MDGM. The  $16 \times 16$  resolution generative model is the same with the 16 – 64 MDGM, shown in the first row of (a). First row:  $32 \times 32$  resolution random samples using the model  $p_{\theta_2}(\mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2^*)$ . We let  $\mathbf{z}_1 = \arg \max q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$ , and  $\mathbf{x}_1$  is the first image in this row, and  $\mathbf{z}_2 \in \mathbb{R}^{64}$  is randomly sampled from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ . Second row:  $64 \times 64$  resolution random samples using the model  $p_{\theta_3}(\mathbf{x}_3|\mathbf{z}_2, \mathbf{z}_3^*)$ . We let  $\mathbf{z}_2 = \arg \max q_{\phi_2}(\mathbf{z}_2|\mathbf{x}_2)$ , and  $\mathbf{x}_2$  is the first image in this row, and  $\mathbf{z}_3 \in \mathbb{R}^{256}$  is randomly sampled from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ .

where  $\mathbf{x} \in \mathbb{R}^{64 \times 64}$  and  $\mathbf{z} \in \mathbb{R}^{256}$ . Although the dimension of the latent variable  $\mathbf{z}$  is still high, the convergence can be realized by constructing a Markov chain with the length of 30000. The inferred results are depicted in Fig. 15. The estimated mean and samples mostly match the exact channels and some important local details.

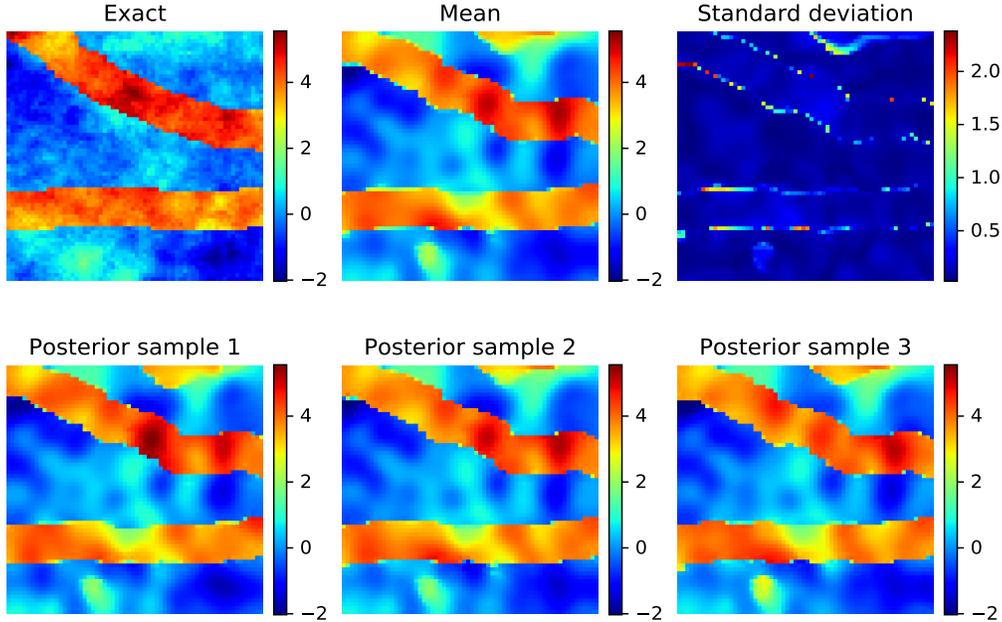


Figure 15: The referenced single-scale estimation result in the desired  $64 \times 64$  grid. Inference with respect to  $\mathbf{z} \in \mathbb{R}^{256}$  is performed using Algorithm 3.

*Two scales (16–64).* For the two-scales inversion example, one needs to estimate successively the parameter with the pre-trained generative model in the coarse-scale with a  $16 \times 16$  grid and fine-scale with a  $64 \times 64$  grid. The coarse-scale estimation uses Algorithm 3 and the pre-trained model  $\mathbf{x}_1 = \mu_{\theta_1}(\mathbf{z}_1)$ , where  $\mathbf{x}_1 \in \mathbb{R}^{16 \times 16}$  and  $\mathbf{z}_1 \in \mathbb{R}^{16}$ . The length of the Markov chain for  $\mathbf{z}_1$  is 7000. The posterior log-permeability fields are illustrated in Fig. 16. Obviously, the method identified all channel locations, which is the most important information in channelized parameter estimation. Similar to the Gaussian case shown in Fig. 8, the estimated variance in the coarse-scale is very low since slightly varying a global feature of the log-permeability will greatly impact the value of the pressure field. Based on the coarse-scale estimation, the refinement results with  $64 \times 64$  grid are shown in Fig. 17. This only needs to run 7000 iterations using Algorithm 4 with the pre-trained model  $\mathbf{x}_2 = \mu_{\theta_2}(\mathbf{z}_2^*, \mathbf{z}_1)$ , where  $\mathbf{x}_2 \in \mathbb{R}^{64 \times 64}$ ,  $\mathbf{z}_2^* \in \mathbb{R}^{256}$ , and  $\mathbf{z}_1 \in \mathbb{R}^{16}$ . The fine-scale local details are

similar to those in the exact log-permeability. However, benefited from the coarse-scale estimation, the calculation saved a lots of computational cost requiring a reduced number of forward model evaluations the  $64 \times 64$  grid.

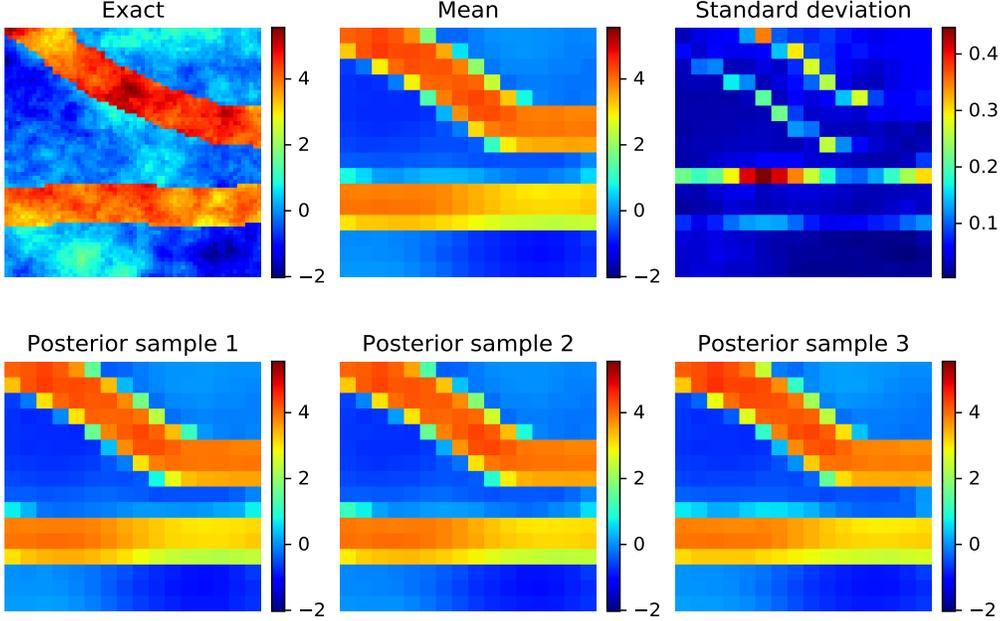


Figure 16: The estimation result with  $16 \times 16$  grid. The results only involve inference in the coarsest scale with  $16 \times 16$  grid, which needs to be corrected and refined in the finer-scale. Inference is performed with respect to  $\mathbf{z}_1 \in \mathbb{R}^{16}$  using Algorithm 3.

*Three scales (16 – 32 – 64).* To discuss the impact of the number of scales, we also studied the three-scale estimation with pre-trained three-scale generative models. The coarsest scale result was shown in Fig. 16. To refine the estimated results in the scale with  $32 \times 32$  grid, we run 7000 iterations in Algorithm 4 with the pre-trained model  $\mathbf{x}_2 = \mu_{\theta_2}(\mathbf{z}_2^*, \mathbf{z}_1)$ , where  $\mathbf{x}_2 \in \mathbb{R}^{32 \times 32}$ ,  $\mathbf{z}_2^* \in \mathbb{R}^{64}$ , and  $\mathbf{z}_1 \in \mathbb{R}^{16}$ . Unlike the estimation in the coarsest-scale that can only identify the location of the channels, the refined results with the  $32 \times 32$  grid can also provide a good estimation regarding the spatial distribution of high- and low-values as shown in Fig. 18. Since most salient features were captured in previous scales, the desired scale estimation becomes much easier. We only need to perform 5000 iterations in Algorithm 4 with the pre-trained model  $\mathbf{x}_3 = \mu_{\theta_3}(\mathbf{z}_3^*, \mathbf{z}_2)$  to guarantee its convergence, where  $\mathbf{x}_3 \in \mathbb{R}^{64 \times 64}$ ,  $\mathbf{z}_3^* \in \mathbb{R}^{256}$ , and  $\mathbf{z}_2 \in \mathbb{R}^{80}$ . Fig. 19 summarizes the final three-scales inference results where we can notice that the estimation is more accurate than the previous two experiments in

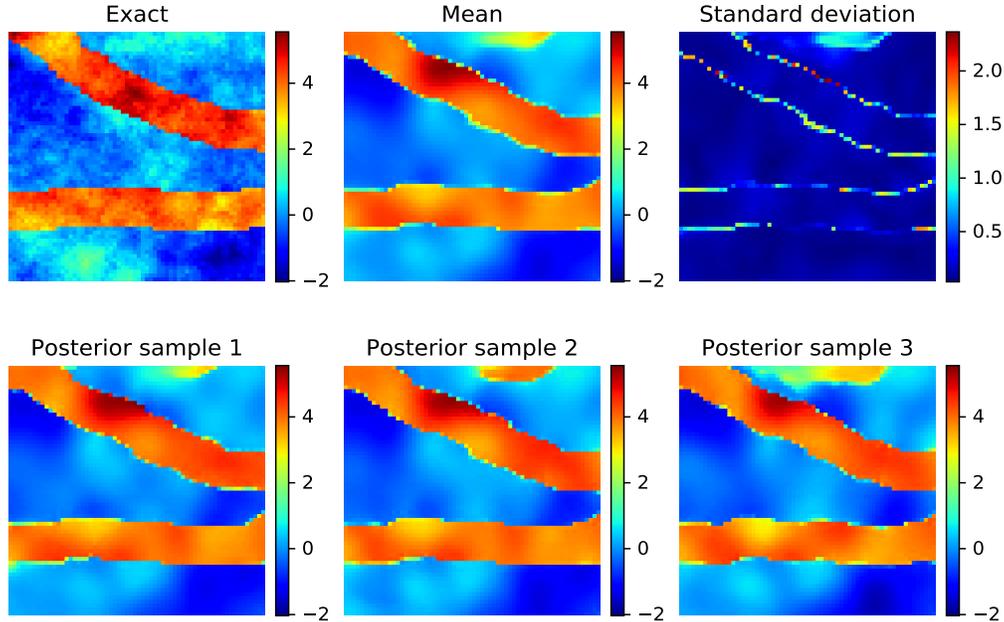


Figure 17: The two-scale estimation result with the desired  $64 \times 64$  grid. The results involve inference across scales. Inference is performed with respect to  $\mathbf{z}_1 \in \mathbb{R}^{16}$  and  $\mathbf{z}_2^* \in \mathbb{R}^{256}$  using Algorithm 4.

both capturing the channels and the local permeability details.

As with the previous examples, we are interested in the posterior estimation in the scale of the  $64 \times 64$  grid. The estimated results with uncertainty from the left bottom corner to the right top corner of the log-permeability field are given in Fig. 20. It can be seen that the posterior means of all experiments are close to the exact value, while the local features estimation by the reference method is worse than the results by the multiscale method. For a channelized log-permeability, the location of the channels will greatly impact the predicted pressure values in the observation/sensor locations. Unlike the parameter only decoded from one latent variable in the single scale method, the latent variables played different roles in MDGM to exploit multiscale characteristics. The channels were identified from the coarse-scale inference, whereas the fine-scale inference corrects and refines the coarse-scale estimation while exploring and learning local features.

The computational cost of the forward model in each experiment is given in Table 4. For non-Gaussian random fields, the superiority of the proposed method is more prominent than the Gaussian random field case. The reference method takes about 2.9 and 4.4 more CPU time than the two-scales and three-scales methods, respectively. The main difference

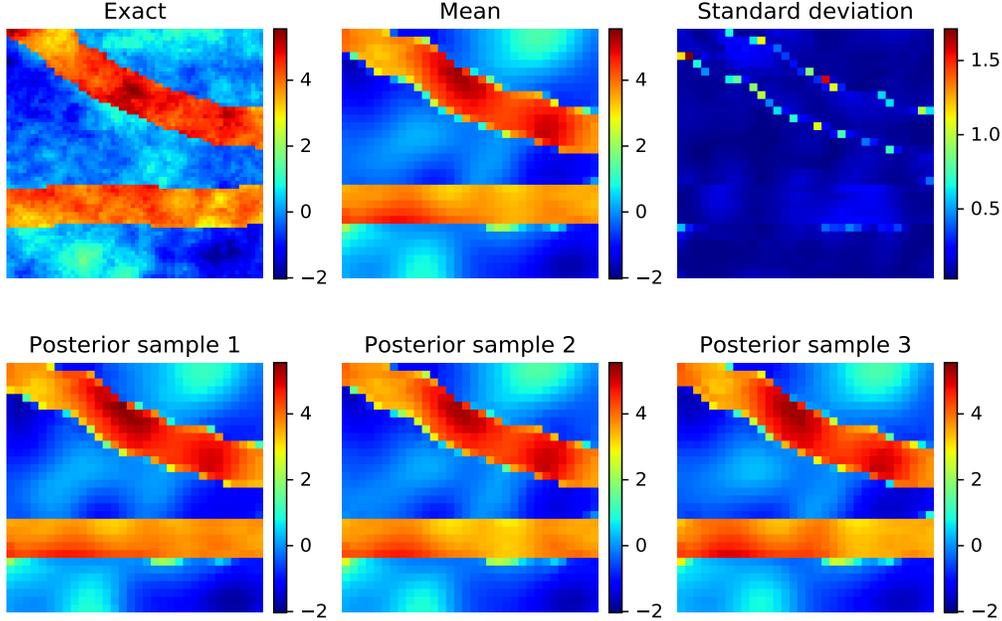


Figure 18: The three-scale estimation result with  $32 \times 32$  grid, which needs to be corrected and refined in the finer-scale. The results involve inference across scales. Inference is performed with respect to  $\mathbf{z}_1 \in \mathbb{R}^{16}$  and  $\mathbf{z}_2^* \in \mathbb{R}^{64}$  using Algorithm 4.

among them is the computational cost in the desired scale. We designed the multiscale scheme to reduce the computational cost in the fine-scale with some additional coarse-scale forward evaluations. As a consequence, this leads to significant computational savings during Bayesian inference. One can apply the proposed method to a computationally more intensive model. Correspondingly, more reduced computational time can be expected. The acceptance rate of each MCMC implementation is shown in Table 5. For the reference single-scale experiment, each dimension of the latent variable has equivalent importance in the generation of the log-permeability. At the same time, channels are the most salient features in this Bayesian inversion example. To infer a high-dimensional latent variable one needs to explore a high-dimensional state space, which will lead to a very low acceptance rate. For the multiscale method, the acceptance rate in the coarsest-scale is still very low. Based on the coarse-scale estimation, applying Algorithm 4 to refine and correct the estimation will become much easier since two proposal distributions with different step sizes provide an informed and efficient exploration. Using a small step size for the low-dimensional latent variable to correct the global features and using a big step size for the high-dimensional latent

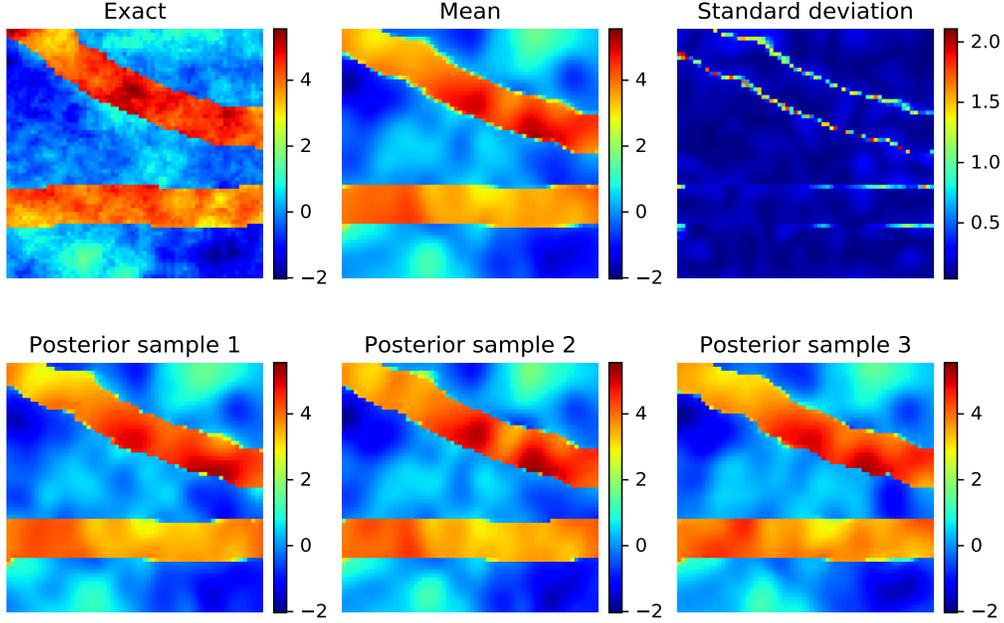


Figure 19: The three-scale estimation result with the desired  $64 \times 64$  grid. The results involve inference across scales. Inference was performed with respect to  $\mathbf{z}_2 \in \mathbb{R}^{80}$  and  $\mathbf{z}_3^* \in \mathbb{R}^{256}$  is performed using Algorithm 4.

variable to explore local features is goal-oriented, which not only improves the acceptance rate but also leads to a better estimation.

Table 4: The iterations (its) and approximated cpu time in seconds(s) for solving the forward model in different experiments for the non-Gaussian random field test problem.

Experiment	$16 \times 16$	$32 \times 32$	$64 \times 64$	Total
one scale (64)*	-	-	30000 its	30000 its
	-	-	93000 s	93000 s
two scales (16 – 64)	7000 its	-	10000 its	17000 its
	910 s	-	31000 s	31910 s
three scales (16 – 32 – 64)	7000 its	7000 its	5000 its	19000 its
	910 s	4690 s	15500 s	21100 s

As with the Gaussian case, we use three evaluation metrics to assess the convergence in the desired scale with the  $64 \times 64$  grid. Fig. 21 indicates a better performance in the channelized log-permeability estimation. It is obvious that the reference single-scale method takes a long exploration to converge to the stationary distribution. As discussed earlier,

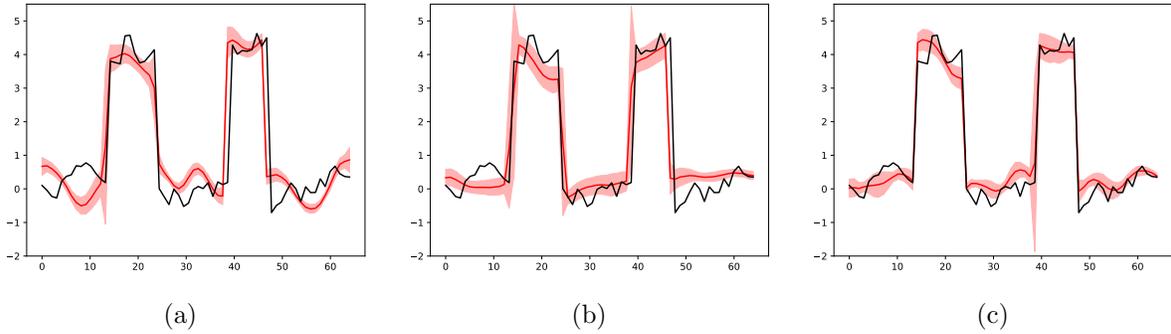


Figure 20: The estimated log-permeability values in the line from the left bottom corner to the right top corner on a  $64 \times 64$  grid. The results from left to right are obtained by (a) reference (64), (b) two-scales (16 – 64), (c) three-scales (16 – 32 – 64) experiment, respectively. The black and red lines show the true and the posterior mean log-permeability, respectively. The shaded region shows values within two standard deviations of the mean.

Table 5: The acceptance ratio of the MH algorithm in different experiments for the non-Gaussian random field test problem.

Experiment	$16 \times 16$	$32 \times 32$	$64 \times 64$
one scale (64)	-	-	5.47%
two scales (16 – 64)	2.13%	-	12.07%
three scales (16 – 32 – 64)	2.13%	9.90%	40.28%

the inferred latent variable of the single-scale inference is still high-dimensional, and each dimension keeps equivalent importance in the log-permeability generation. In such scenarios, it is easy to get trapped in local modes. Fig. 22 provides the state evolution of the Markov chain to infer the log-permeability on the  $64 \times 64$  grid. The single-scale method captures well the true solution at about the 15000–th iteration, while the multiscale method only needs to refine the coarse-scale estimation for fine-scale inference. By greatly reducing the fine-scale forward model evaluations, the computational burden in Bayesian inverse problems is reduced. For a parameter like the channelized log-permeability with obvious multiscale characteristics, it can readily be seen that the three-scales experiment performs better than the two-scales experiment with respect to stability, efficiency and accuracy.

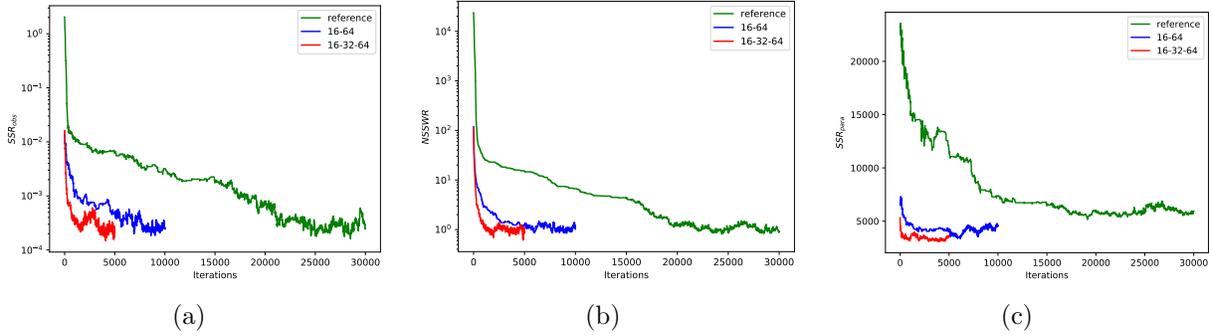


Figure 21: The convergence of the Markov chain used for the non-Gaussian log-permeability field estimation with a  $64 \times 64$  grid. The evaluation metrics from left to right are (a) the SSR of observable pressure values (b) NSSWR values (c) the SSR of parameter field, respectively.

## 5. Conclusions

In this work, we introduced a novel multiscale parameter estimation framework for Bayesian inverse problems based on a multiscale deep generative model. The deep generative model has been proven to be promising for the characterization of complex spatially varying parameters. To exploit the multiscale characteristics, we extended the existing VAE-based deep generative model into a multiscale framework with multiple latent variables. Endowing the latent variables with different missions using training data at various scales, the low-dimensional latent variables can generate coarse-scales parameters and dominate the global features in finer-scale parameter generation, while the high-dimensional latent variables can enrich local details. We demonstrated the model with Gaussian and non-Gaussian parameter estimation. Combining pre-trained multiscale deep generative models with a multiscale inference strategy, we hierarchically performed inference from coarse- to fine-scale.

Benefited from the construction of the latent space in the multiscale generative model, the coarse-scale estimation explores in the low-dimensional latent space and searches for all possible global patterns by invoking the extremely cheap forward model. Using previous estimation results, the fine-scale estimation refines the parameters by correcting the global features and enriching the local features using the expensive fine-scale forward model. It was demonstrated that coarse-scale estimation information could pass across scales via the designed latent space, which plays an important role in accelerated convergence. In the two

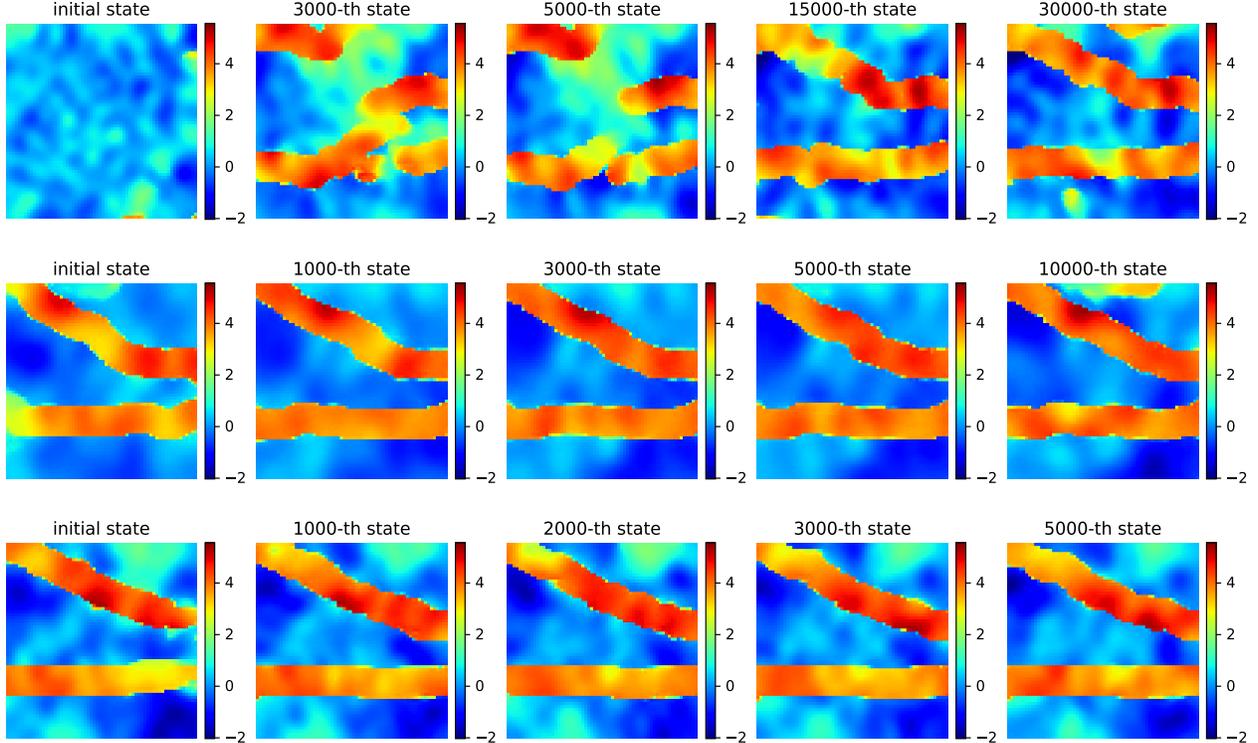


Figure 22: The states of the non-Gaussian log-permeability field with  $64 \times 64$  grid in the Markov chain. The results from top to bottom row are obtained from (a) reference (64), (b) two scales (16 – 64), (c) three scales (16 – 32 – 64) experiment, respectively.

test cases, the proposed method shows superior performance over the reference single-scale method in computational cost and accuracy. We also discussed in the non-Gaussian case, the importance of the number of scales considered in the generative model and parameter estimation.

Some challenges and extensions are worthy to explore in the future. The fundamental requirement for the proposed method is to train a stable and desired multiscale deep generative model, which involves different setups for various types of parameters, like the number of training data, the number of scales, hyperparameter selection, and so on. Further study of the multiscale generative model has promising applications on super resolution, multiscale uncertainty quantification, and so on. In addition, note that we use the simple Metropolis–Hastings algorithm with pCN proposal distribution as the Bayesian inference method. Enhanced sampling techniques like sequential MC (SMC) that can realize parallel computation will result in accelerated exploration and high-efficiency.

## Acknowledgements

N.Z. acknowledges support from the Defense Advanced Research Projects Agency (DARPA) under the Physics of Artificial Intelligence (PAI) program (contract HR00111890034). Computing resources were provided by the AFOSR Office of Scientific Research through the DURIP program and by the University of Notre Dame’s Center for Research Computing (CRC).

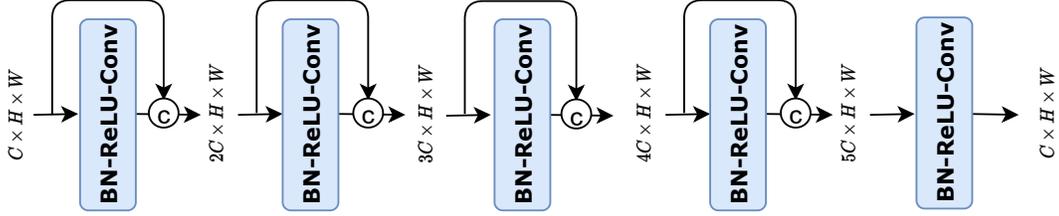
## Appendix A. Neural network architectures for the encoder and decoder networks

In this work, we use convolutional neural networks (CNNs) [72] for the encoder and decoder models. CNNs are more effective in capturing multiscale features than fully-connected neural networks and allow modeling of the hierarchical nature of the features [73]. The implemented encoder and decoder neural networks [74, 75, 76, 16] are illustrated in Fig. A.23. The batch size is 64 for all implementations.

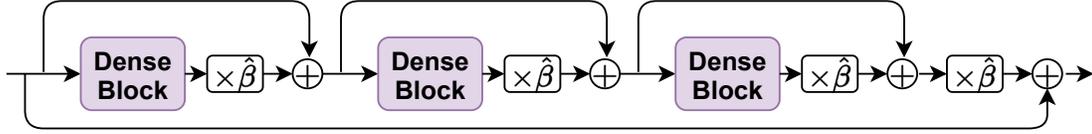
## Appendix B. Concatenation of latent variables

In the MDGM, the  $l$ -th scale encoder network includes two parts i.e. the augmented encoder network and the  $(l - 1)$ -th scale encoder network (see Fig. 5). Since the training is recursive, the  $(l - 1)$ -th encoder network also includes the augmented encoder network and the  $(l - 2)$ -th scale encoder networks and so on. The latent variable in the  $l$ -th scale is  $\mathbf{z}_l = (\mathbf{z}_1, \mathbf{z}_2^*, \dots, \mathbf{z}_l^*)$ . In this paper, all the augmented encoder and decoder networks employ the same architectures described in Appendix A, so the elements  $(\mathbf{z}_1, \mathbf{z}_2^*, \dots, \mathbf{z}_l^*)$  in  $\mathbf{z}_l$  have proportional sizes depending on their input size. For example, we can obtain  $\mathbf{z}_1 = q_{\phi_1}(\mathbf{z}_1|\mathbf{x}_1)$  and  $\mathbf{z}_2^* = q_{\phi_2^*}(\mathbf{z}_2^*|\mathbf{x}_2)$  using the encoder model, where  $\mathbf{z}_1 \in \mathbb{R}^{4 \times 4}$ ,  $\mathbf{z}_2^* \in \mathbb{R}^{8 \times 8}$ ,  $\mathbf{x}_1 \in \mathbb{R}^{16 \times 16}$ , and  $\mathbf{x}_2 \in \mathbb{R}^{32 \times 32}$ . The input size of the  $l$ -th scale decoder network should be  $C \times H \times W$ . For example, in the previous example, the size of  $\mathbf{z}_2$  is  $2 \times 8 \times 8$ , where the number of channels is equal to  $C = l$  since  $\mathbf{z}_l$  is stacked by the outputs of  $l$  encoders. Also,  $H \times W$  is the size of  $\mathbf{z}_l^*$ , and  $\mathbf{z}_l$  must be reshaped as a tensor in such size.

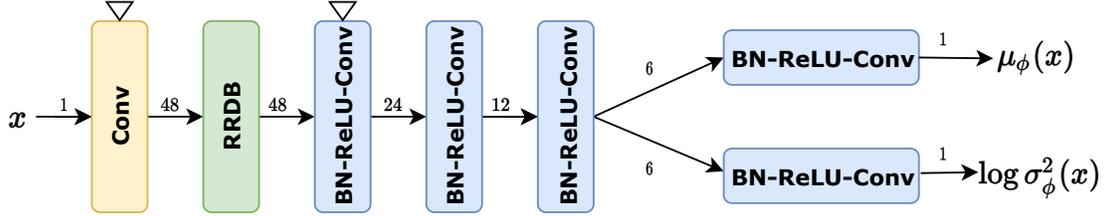
(a) Dense Block



(b) RRDB



(c) Encoder



(d) Decoder

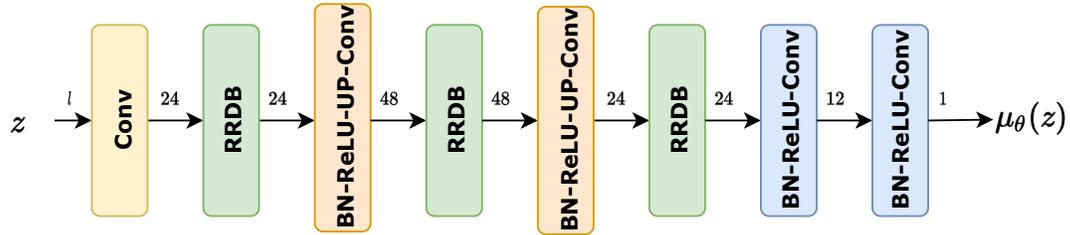


Figure A.23: (a) Dense block with five layers. Its input consists of  $C$  feature maps/channels with size  $H \times W$ . In each layer, its output is computed successively by three operators, i.e. Batch Normalization (BN), Rectified Linear Units (ReLU), and Convolution (Conv), where  $C$  is specified above the arrows in the encoder and decoder architectures in sub-figures (c) and (d). The output feature maps are concatenated with the input feature maps. The concatenated feature maps are the input to the next layer. (b) A residual-in-residual dense block (RRDB) using 3 residual dense blocks. In each block, the output is multiplied by a constant  $\hat{\beta}$  and then is added to the input with the result serving as the input for the next dense block. We let  $\hat{\beta}$  be 0.2 in this paper. (c) Encoder neural network architecture. The feature map size  $H \times W$  is halved by Conv operator in  $\nabla$  with a stride 2. (d) Decoder neural network architecture. The number of feature maps of the input is equal to the scale number  $l$ . The feature map size  $H \times W$  is doubled by applying the nearest upsampling (UP) operator.

To make the other  $(l - 1)$  elements (i.e.  $z_1, z_2^*, \dots, z_{l-1}^*$ ) in  $z_l$  to be of consistent size with  $z_l^*$ , we use the *Upsample* operator<sup>3</sup> in the Pytorch library [77] over these elements and then concatenate<sup>4</sup> all of them as input  $z_l \in \mathbb{R}^{C \times H \times W}$  for the  $l$ -th scale decoder networks. The scale factor in the *Upsample* operator depends on the output and input sizes. Their sizes satisfy the following relationship:

$$\begin{aligned} H_{out} &= H_{in} \times \text{scale factor}, \\ W_{out} &= W_{in} \times \text{scale factor}, \end{aligned} \tag{B.1}$$

where  $[H_{in} \times W_{in}]$  and  $[H_{out} \times W_{out}]$  are the input and output sizes, respectively. We used the nearest mode in *Upsample* operator. A simple example is given in Fig. B.24 to illustrate this process (the scale factor here is 2).

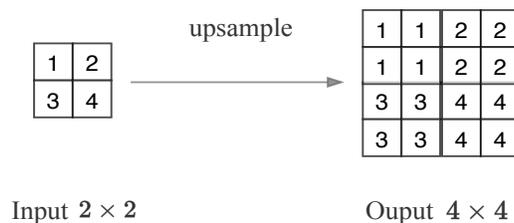


Figure B.24: The *Upsample* operator example with nearest mode, where the input image size is  $2 \times 2$  and the output size is  $4 \times 4$ .

## References

- [1] A. Tarantola, Inverse Problem Theory and Methods for Model Parameter Estimation, Society for Industrial and Applied Mathematics, 2005. arXiv:<https://epubs.siam.org/doi/pdf/10.1137/1.9780898717921>, doi:10.1137/1.9780898717921. URL <https://epubs.siam.org/doi/abs/10.1137/1.9780898717921>
- [2] H. W. Engl, M. Hanke, A. Neubauer, Regularization of inverse problems, Vol. 375, Springer Science & Business Media, 1996. doi:<https://doi.org/10.1007/>

<sup>3</sup><https://pytorch.org/docs/master/generated/torch.nn.Upsample.html>

<sup>4</sup><https://pytorch.org/docs/master/generated/torch.cat.html>

978-3-540-70529-1\_52.

URL <https://www.springer.com/gp/book/9780792341574>

- [3] M. Zhdanov, Geophysical Inverse Theory and Regularization Problems, Methods in Geochemistry and Geophysics, Elsevier Science, 2002.

URL <https://books.google.com/books?id=tHtDETv7VCoC>

- [4] P. C. Hansen, Discrete Inverse Problems, Society for Industrial and Applied Mathematics, 2010. arXiv:<https://epubs.siam.org/doi/pdf/10.1137/1.9780898718836>, doi:10.1137/1.9780898718836.

URL <https://epubs.siam.org/doi/abs/10.1137/1.9780898718836>

- [5] A. M. Stuart, Inverse problems: A Bayesian perspective, Acta Numerica 19 (2010) 451–559. doi:10.1017/S0962492910000061.

- [6] D. M. Blei, A. Kucukelbir, J. D. McAuliffe, Variational inference: A review for statisticians, Journal of the American statistical Association 112 (518) (2017) 859–877. arXiv: <https://doi.org/10.1080/01621459.2017.1285773>, doi:10.1080/01621459.2017.1285773.

URL <https://doi.org/10.1080/01621459.2017.1285773>

- [7] L. Bruder, P.-S. Koutsourelakis, Beyond black-boxes in Bayesian inverse problems and model validation: Applications in solid mechanics of elastography, International Journal for Uncertainty Quantification 8 (5) (2018). doi:<https://doi.org/10.1615/Int.J.UncertaintyQuantification.2018025837>.

- [8] B. Jin, A variational Bayesian method to inverse problems with impulsive noise, Journal of Computational Physics 231 (2) (2012) 423 – 435. doi:<https://doi.org/10.1016/j.jcp.2011.09.009>.

URL <http://www.sciencedirect.com/science/article/pii/S0021999111005298>

- [9] S. Atkinson, N. Zabaras, Structured Bayesian Gaussian process latent variable model: Applications to data-driven dimensionality reduction and high-dimensional inversion, Journal of Computational Physics 383 (2019) 166 – 195. doi:<https://doi.org/10.1016/j.jcp.2019.04.011>.

1016/j.jcp.2018.12.037.

URL <http://www.sciencedirect.com/science/article/pii/S0021999119300397>

- [10] Q. Liao, J. Li, An adaptive reduced basis ANOVA method for high-dimensional Bayesian inverse problems, *Journal of Computational Physics* 396 (2019) 364 – 380. doi:<https://doi.org/10.1016/j.jcp.2019.06.059>.

URL <http://www.sciencedirect.com/science/article/pii/S002199911930467X>

- [11] S. Mo, Y. Zhu, N. Zabaras, X. Shi, J. Wu, Deep Convolutional Encoder-Decoder Networks for Uncertainty Quantification of Dynamic Multiphase Flow in Heterogeneous Media, *Water Resources Research* 55 (1) (2019) 703–728. arXiv: <https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018WR023528>, doi:<https://doi.org/10.1029/2018WR023528>.

URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018WR023528>

- [12] X. Ma, N. Zabaras, An efficient Bayesian inference approach to inverse problems based on an adaptive sparse grid collocation method, *Inverse Problems* 25 (3) (2009) 035013. doi:[10.1088/0266-5611/25/3/035013](https://doi.org/10.1088/0266-5611/25/3/035013).

URL <https://doi.org/10.1088/0266-5611/25/3/035013>

- [13] J. Wan, N. Zabaras, A Bayesian approach to multiscale inverse problems using the sequential Monte Carlo method, *Inverse Problems* 27 (10) (2011) 105004. doi:[10.1088/0266-5611/27/10/105004](https://doi.org/10.1088/0266-5611/27/10/105004).

URL <https://doi.org/10.1088/0266-5611/27/10/105004>

- [14] L. Ellam, N. Zabaras, M. Girolami, A Bayesian approach to multiscale inverse problems with on-the-fly scale determination, *Journal of Computational Physics* 326 (2016) 115 – 140. doi:<https://doi.org/10.1016/j.jcp.2016.08.031>.

URL <http://www.sciencedirect.com/science/article/pii/S0021999116303886>

- [15] E. Laloy, R. Héroult, J. Lee, D. Jacques, N. Linde, Inversion using a new low-dimensional representation of complex binary geological media based on a deep neural network,

- Advances in Water Resources 110 (2017) 387 – 405. doi:<https://doi.org/10.1016/j.advwatres.2017.09.029>.  
URL <http://www.sciencedirect.com/science/article/pii/S0309170817306243>
- [16] S. Mo, N. Zabarar, X. Shi, J. Wu, Integration of adversarial autoencoders with residual dense convolutional networks for estimation of non-Gaussian hydraulic conductivities, *Water Resources Research* 56 (2) (2020) e2019WR026082, e2019WR026082 10.1029/2019WR026082. arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2019WR026082>, doi:10.1029/2019WR026082.  
URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019WR026082>
- [17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems*, Vol. 27, Curran Associates, Inc., 2014, pp. 2672–2680.  
URL <https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf>
- [18] D. P. Kingma, M. Welling, Auto-encoding variational Bayes, arXiv preprint arXiv:1312.6114 (2013).
- [19] D. J. Rezende, S. Mohamed, Variational inference with normalizing flows, arXiv preprint arXiv:1505.05770 (2015).
- [20] G. A. Padmanabha, N. Zabarar, Solving inverse problems using conditional invertible neural networks, arXiv preprint arXiv:2007.15849 (2020).
- [21] S. W. Canchumuni, A. A. Emerick, M. A. C. Pacheco, Towards a robust parameterization for conditioning facies models using deep variational autoencoders and ensemble smoother, *Computers & Geosciences* 128 (2019) 87 – 102. doi:<https://doi.org/10.1016/j.cageo.2019.04.006>.  
URL <http://www.sciencedirect.com/science/article/pii/S0098300419300378>

- [22] E. Laloy, R. Hérault, D. Jacques, N. Linde, Training-image based geostatistical inversion using a spatial generative adversarial neural network, *Water Resources Research* 54 (1) (2018) 381–406. arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2017WR022148>, doi:<https://doi.org/10.1002/2017WR022148>.  
URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017WR022148>
- [23] L. Mosser, O. Dubrule, M. J. Blunt, Stochastic seismic waveform inversion using generative adversarial networks as a geological prior, *Mathematical Geosciences* 52 (1) (2020) 53–79. doi:<https://doi.org/10.1007/s11004-019-09832-6>.
- [24] J. Li, Y. M. Marzouk, Adaptive Construction of Surrogates for the Bayesian Solution of Inverse Problems, *SIAM Journal on Scientific Computing* 36 (3) (2014) A1163–A1186. arXiv:<https://doi.org/10.1137/130938189>, doi:10.1137/130938189.  
URL <https://doi.org/10.1137/130938189>
- [25] Y. M. Marzouk, H. N. Najm, L. A. Rahn, Stochastic spectral methods for efficient Bayesian solution of inverse problems, *Journal of Computational Physics* 224 (2) (2007) 560 – 586. doi:<https://doi.org/10.1016/j.jcp.2006.10.010>.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999106004839>
- [26] C. Chen, Q. Liao, ANOVA Gaussian process modeling for high-dimensional stochastic computational models, *Journal of Computational Physics* 416 (2020) 109519. doi:<https://doi.org/10.1016/j.jcp.2020.109519>.  
URL <http://www.sciencedirect.com/science/article/pii/S002199912030293X>
- [27] I. Bilonis, N. Zabaras, B. A. Konomi, G. Lin, Multi-output separable Gaussian process: Towards an efficient, fully Bayesian paradigm for uncertainty quantification, *Journal of Computational Physics* 241 (2013) 212 – 239. doi:<https://doi.org/10.1016/j.jcp.2013.01.011>.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999113000417>
- [28] K. Li, K. Tang, J. Li, T. Wu, Q. Liao, A hierarchical neural hybrid method for failure

- probability estimation, *IEEE Access* 7 (2019) 112087–112096. doi:10.1109/ACCESS.2019.2934980.
- [29] Y. Zhu, N. Zabararas, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, *Journal of Computational Physics* 366 (2018) 415 – 447. doi:<https://doi.org/10.1016/j.jcp.2018.04.018>.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999118302341>
- [30] S. Mo, N. Zabararas, X. Shi, J. Wu, Deep Autoregressive Neural Networks for High-Dimensional Inverse Problems in Groundwater Contaminant Source Identification, *Water Resources Research* 55 (5) (2019) 3856–3881. arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018WR024638>, doi:<https://doi.org/10.1029/2018WR024638>.  
URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018WR024638>
- [31] I. Bilonis, N. Zabararas, Solution of inverse problems with limited forward solver evaluations: A Bayesian perspective, *Inverse Problems* 30 (1) (2013) 015004. doi:10.1088/0266-5611/30/1/015004.  
URL <https://doi.org/10.1088/0266-5611/30/1/015004>
- [32] M. A. Ferreira, A. Marco, H. K. Lee, *Multiscale Modeling: A Bayesian Perspective*, Springer Science & Business Media, 2007.
- [33] P. Koutsourelakis, A multi-resolution, non-parametric, Bayesian framework for identification of spatially-varying model parameters, *Journal of Computational Physics* 228 (17) (2009) 6184 – 6211. doi:<https://doi.org/10.1016/j.jcp.2009.05.016>.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999109002708>
- [34] D. Higdon, H. Lee, Zhuoxin Bi, A Bayesian approach to characterizing uncertainty in inverse problems using coarse and fine-scale information, *IEEE Transactions on Signal Processing* 50 (2) (2002) 389–399. doi:10.1109/78.978393.

- [35] B. Peherstorfer, K. Willcox, M. Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization, *SIAM Review* 60 (3) (2018) 550–591. arXiv:<https://doi.org/10.1137/16M1082469>, doi:10.1137/16M1082469.  
URL <https://doi.org/10.1137/16M1082469>
- [36] L. Yan, T. Zhou, Adaptive multi-fidelity polynomial chaos approach to Bayesian inference in inverse problems, *Journal of Computational Physics* 381 (2019) 110 – 128. doi:<https://doi.org/10.1016/j.jcp.2018.12.025>.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999119300063>
- [37] Y. Efendiev, T. Hou, W. Luo, Preconditioning Markov Chain Monte Carlo simulations using coarse-scale models, *SIAM Journal on Scientific Computing* 28 (2) (2006) 776–803. arXiv:<https://doi.org/10.1137/050628568>, doi:10.1137/050628568.  
URL <https://doi.org/10.1137/050628568>
- [38] T. J. Dodwell, C. Ketelsen, R. Scheichl, A. L. Teckentrup, A hierarchical multilevel Markov Chain Monte Carlo algorithm with applications to uncertainty quantification in subsurface flow, *SIAM/ASA Journal on Uncertainty Quantification* 3 (1) (2015) 1075–1108. arXiv:<https://doi.org/10.1137/130915005>, doi:10.1137/130915005.  
URL <https://doi.org/10.1137/130915005>
- [39] V. H. Hoang, C. Schwab, A. M. Stuart, Complexity analysis of accelerated MCMC methods for Bayesian inversion, *Inverse Problems* 29 (8) (2013) 085010. doi:10.1088/0266-5611/29/8/085010.  
URL <https://doi.org/10.1088/0266-5611/29/8/085010>
- [40] A. Beskos, A. Jasra, K. Law, R. Tempone, Y. Zhou, Multilevel sequential Monte Carlo samplers, *Stochastic Processes and their Applications* 127 (5) (2017) 1417 – 1440. doi:<https://doi.org/10.1016/j.spa.2016.08.004>.  
URL <http://www.sciencedirect.com/science/article/pii/S0304414916301326>
- [41] T. Cui, G. Detommaso, R. Scheichl, Multilevel dimension-independent likelihood-informed MCMC for large-scale inverse problems, arXiv preprint arXiv:1910.12431 (2019).

- [42] L. Dinh, D. Krueger, Y. Bengio, Nice: Non-linear independent components estimation, arXiv preprint arXiv:1410.8516 (2014).
- [43] K. Tang, X. Wan, Q. Liao, Deep density estimation via invertible block-triangular mapping, *Theoretical and Applied Mechanics Letters* 10 (3) (2020) 143 – 148. doi:  
<https://doi.org/10.1016/j.taml.2020.01.023>.  
URL <http://www.sciencedirect.com/science/article/pii/S209503492030026X>
- [44] Z. Xu, Q. Liao, Gaussian Process based expected information gain computation for Bayesian optimal design, *Entropy* 22 (2) (2020). doi:10.3390/e22020258.  
URL <https://www.mdpi.com/1099-4300/22/2/258>
- [45] Y. Liu, W. Sun, L. J. Durlofsky, A deep-learning-based geological parameterization for history matching complex models, *Mathematical Geosciences* 51 (6) (2019) 725–766. doi:  
<https://doi.org/10.1007/s11004-019-09794-9>.
- [46] M. Tang, Y. Liu, L. J. Durlofsky, A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems, *Journal of Computational Physics* 413 (2020) 109456. doi:  
<https://doi.org/10.1016/j.jcp.2020.109456>.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999120302308>
- [47] G. Dorta, S. Vicente, L. Agapito, N. D. F. Campbell, I. Simpson, Structured uncertainty prediction networks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 5477–5485. doi:10.1109/CVPR.2018.00574.
- [48] G. Dorta, S. Vicente, L. Agapito, N. D. F. Campbell, I. Simpson, Training VAEs under structured residuals (2018). arXiv:1804.01050.
- [49] M. Schöberl, N. Zabaras, P.-S. Koutsourelakis, Predictive collective variable discovery with deep Bayesian models, *The Journal of Chemical Physics* 150 (2) (2019) 024109. arXiv:  
<https://doi.org/10.1063/1.5058063>, doi:10.1063/1.5058063.  
URL <https://doi.org/10.1063/1.5058063>
- [50] D. Lu, G. Zhang, C. Webster, C. Barbier, An improved multilevel Monte Carlo method for estimating probability distribution functions in stochastic oil reser-

- voir simulations, *Water Resources Research* 52 (12) (2016) 9642–9660. arXiv:  
<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2016WR019475>,  
doi:<https://doi.org/10.1002/2016WR019475>.
- URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2016WR019475>
- [51] X. Wen, L. Durlafsky, M. Edwards, Upscaling of channel systems in two dimensions using flow-based grids, *Transport in Porous Media* 51 (3) (2003) 343–366. doi:<https://doi.org/10.1023/A:1022318926559>.
- [52] X.-H. Wen, J. J. Gómez-Hernández, Upscaling hydraulic conductivities in heterogeneous media: An overview, *Journal of Hydrology* 183 (1) (1996) ix – xxxii. doi:[https://doi.org/10.1016/S0022-1694\(96\)80030-8](https://doi.org/10.1016/S0022-1694(96)80030-8).
- URL <http://www.sciencedirect.com/science/article/pii/S0022169496800308>
- [53] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, beta-vae: Learning basic visual concepts with a constrained variational framework, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017.
- URL <https://openreview.net/forum?id=Sy2fzU9g1>
- [54] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Vol. 29, Curran Associates, Inc., 2016, pp. 2172–2180.
- URL <https://proceedings.neurips.cc/paper/2016/file/7c9d0b1f96aebd7b5eca8c3edaa19ebb-Paper.pdf>
- [55] I. Higgins, N. Sonnerat, L. Matthey, A. Pal, C. P. Burgess, M. Bosnjak, M. Shanahan, M. Botvinick, D. Hassabis, A. Lerchner, SCAN: Learning hierarchical compositional visual concepts, in: 6th International Conference on Learning Representations, ICLR

2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018.

URL <https://openreview.net/forum?id=rkN2I1-RZ>

- [56] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, Equation of state calculations by fast computing machines, *The Journal of Chemical Physics* 21 (6) (1953) 1087–1092. doi:<https://doi.org/10.1063/1.1699114>.
- [57] W. K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, *Biometrika* 57 (1) (1970) 97–109.  
URL <http://www.jstor.org/stable/2334940>
- [58] C. Andrieu, N. De Freitas, A. Doucet, M. I. Jordan, An introduction to MCMC for machine learning, *Machine Learning* 50 (1-2) (2003) 5–43. doi:<https://doi.org/10.1023/A:1020281327116>.
- [59] A. Gelman, G. O. Roberts, W. R. Gilks, et al., Efficient Metropolis jumping rules, *Bayesian Statistics* 5 (599-608) (1996) 42.
- [60] S. L. Cotter, G. O. Roberts, A. M. Stuart, D. White, MCMC methods for functions: Modifying old algorithms to make them faster, *Statistical Science* 28 (3) (2013) 424–446.  
URL <http://www.jstor.org/stable/43288425>
- [61] M. Hairer, A. M. Stuart, S. J. Vollmer, Spectral gaps for a Metropolis–Hastings algorithm in infinite dimensions, *The Annals of Applied Probability* 24 (6) (2014) 2455–2490.  
URL <http://www.jstor.org/stable/24520134>
- [62] T. J. Dodwell, C. Ketelsen, R. Scheichl, A. L. Teckentrup, Multilevel Markov Chain Monte Carlo, *SIAM Review* 61 (3) (2019) 509–545. arXiv:<https://doi.org/10.1137/19M126966X>, doi:10.1137/19M126966X.  
URL <https://doi.org/10.1137/19M126966X>
- [63] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, G. N. Wells, The FEniCS project version 1.5, *Archive of Numerical Software* 3 (100) (2015).

- [64] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
- [65] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, A. Lerchner, Towards a definition of disentangled representations, arXiv preprint arXiv:1812.02230 (2018).
- [66] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (8) (2013) 1798–1828. doi:10.1109/TPAMI.2013.50.
- [67] R. Suter, D. Miladinovic, B. Schölkopf, S. Bauer, Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness, in: K. Chaudhuri, R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning Research*, PMLR, 2019, pp. 6056–6065.  
URL <http://proceedings.mlr.press/v97/suter19a.html>
- [68] E. Laloy, J. A. Vrugt, High-dimensional posterior exploration of hydrologic models using multiple-try DREAM (ZS) and high-performance computing, *Water Resources Research* 48 (1) (2012). arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2011WR010608>, doi:<https://doi.org/10.1029/2011WR010608>.  
URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2011WR010608>
- [69] A. A. Emerick, Investigation on principal component analysis parameterizations for history matching channelized facies models with ensemble-based data assimilation, *Mathematical Geosciences* 49 (1) (2017) 85–120. doi:<https://doi.org/10.1007/s11004-016-9659-5>.
- [70] H. X. Vo, L. J. Durlofsky, Data assimilation and uncertainty assessment for complex geological models using a new PCA-based parameterization, *Computational Geosciences* 19 (4) (2015) 747–767. doi:<https://doi.org/10.1007/s10596-015-9483-x>.

- [71] P. Sarma, L. J. Durlofsky, K. Aziz, W. H. Chen, et al., A new approach to automatic history matching using kernel PCA, in: SPE Reservoir Simulation Symposium, Society of Petroleum Engineers, 2007. doi:<https://doi.org/10.2118/106176-MS>.
- [72] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105.  
URL <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>
- [73] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 818–833.
- [74] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269. doi:[10.1109/CVPR.2017.243](https://doi.org/10.1109/CVPR.2017.243).
- [75] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, Y. Fu, Residual dense network for image super-resolution, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 2472–2481. doi:[10.1109/CVPR.2018.00262](https://doi.org/10.1109/CVPR.2018.00262).
- [76] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. doi:[10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- [77] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019, pp. 8026–8037.

URL [https://proceedings.neurips.cc/paper/2019/file/  
bdbca288fee7f92f2bfa9f7012727740-Paper.pdf](https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf)