Journal of Machine Learning Research 24 (2023) 1-60 Submitted 1/22; Revised 2/23; Published 5/23

Variational Inverting Network for Statistical Inverse Problems of Partial Differential Equations

Junxiong Jia JJX323@MAIL.XJTU.EDU.CN
School of Mathematics and Statistics,
Xi'an Jiaotong University,
Xi'an, 710049, China

Yanni Wu WUYANNI@STU.XJTU.EDU.CN
School of Mathematics and Statistics,
Xi'an Jiaotong University,
Xi'an, 710049, China

Peijun Li LIPEIJUN@MATH.PURDUE.EDU
Department of Mathematics,
Purdue University,
West Lafayette, Indiana, 47907, USA

Deyu Meng* DYMENG@MAIL.XJTU.EDU.CN
School of Mathematics and Statistics,
Ministry of Education Key Lab of Intelligent Networks and Network Security,
Xi'an Jiaotong University,
Xi'an, Shaanxi 710049, China,
Peng Cheng Laboratory,
Shenzhen, Guangdong 518066, China,
Macau Institute of Systems Engineering,
Macau University of Science and Technology,
Taipa, Macau, 999078, China.

Editor: Stefan Harmeling

Abstract

To quantify uncertainties in inverse problems of partial differential equations (PDEs), we formulate them into statistical inference problems using Bayes' formula. Recently, well-justified infinite-dimensional Bayesian analysis methods have been developed to construct dimension-independent algorithms. However, there are three challenges for these infinite-dimensional Bayesian methods: prior measures usually act as regularizers and are not able to incorporate prior information efficiently; complex noises, such as more practical non-i.i.d. distributed noises, are rarely considered; and time-consuming forward PDE solvers are needed to estimate posterior statistical quantities. To address these issues, an infinite-dimensional inference framework has been proposed based on the infinite-dimensional variational inference method and deep generative models. Specifically, by introducing some measure equivalence assumptions, we derive the evidence lower bound in the infinite-dimensional setting and provide possible parametric strategies that yield a general inference framework called the Variational Inverting Network (VINet). This inference framework

*. Corresponding author

can encode prior and noise information from learning examples. In addition, relying on the power of deep neural networks, the posterior mean and variance can be efficiently and explicitly generated in the inference stage. In numerical experiments, we design specific network structures that yield a computable VINet from the general inference framework. Numerical examples of linear inverse problems of an elliptic equation and the Helmholtz equation are presented to illustrate the effectiveness of the proposed inference framework.

Keywords: infinite-dimensional variational inference, inverse problems, Bayesian analysis for functions, partial differential equations, deep neural networks

1. Introduction

Motivated by their significant applications in seismic exploration, radar imaging, and many other domains, inverse problems of partial differential equations (PDEs) have undergone enormous development over the past few decades (Engl et al., 1996). As computational power keeps increasing, researchers are not satisfied with obtaining just an estimated solution but pursue performing some statistical analysis based on uncertain information, which is essential for some applications, such as artifact detection (Zhou et al., 2020). The Bayesian inverse approach transforms inverse problems into statistical inference problems and has provided a framework for analyzing the uncertainties of the interested parameters in inverse problems of PDEs (Stuart, 2010).

As is known, PDEs are usually defined on some infinite-dimensional spaces of functions (Evans, 2010), which makes it difficult to directly use the finite-dimensional Bayes' formula. To resolve this issue, a straightforward approach is to discretize PDEs in order to approximate the original problem in some finite-dimensional spaces and then solve the approximate problems by applying finite-dimensional Bayes' methods. This strategy makes nearly all of the parametric Bayesian inference methods developed in the statistical literature applicable (Berger, 1980; Kaipio and Somersalo, 2005). Recently, under the finite-dimensional setting, machine learning methods, e.g., Wasserstein GAN (Adler and Öktem, 2018), have been employed to learn the prior and construct fast sampling algorithms. However, given that the original problems are defined in infinite-dimensional spaces, two critical issues are inevitably encountered.

1. Model consistency: The elements in a finite-dimensional model (e.g., prior probability measure) possess different intrinsic characteristics from their intuitive infinite-dimensional counterparts. One typical example is the total variation prior measure, which has been comprehensively analyzed in (Lassas and Siltanen, 2004).

2. Algorithm applicability: To ensure that the algorithms can preserve the structures of the original infinite-dimensional problems, the algorithms designed on finite-dimensional space need to be reconstructed on infinite-dimensional space. Through re-design, the algorithms well-defined on infinite-dimensional space will have consistent behavior for different discretizations; see (Cotter et al., 2013; Beskos et al., 2015; Jia et al., 2022) for examples.

To overcome these obstacles, Bayes' formula defined on some separable Banach space is employed to handle the inverse problems of PDEs (Dashti and Stuart, 2017; Stuart, 2010), which is an active research topic in the field of nonparametric Bayesian inference. We refer

to (Ghosal and Vaart, 2017) for a general account of the nonparametric Bayesian inference approach. Recently, the theory of well-posedness of the infinite-dimensional Bayesian analysis method has been generalized to incorporate some machine learning problems (Latz, 2020). Based on this infinite-dimensional Bayesian analysis theory, graph-based Bayesian semi-supervised learning algorithms have been analyzed from many different aspects, such as large data and zero noise limit (Dunlop et al., 2020), as well as the consistency and scalability of sampling algorithms (Hoffmann et al., 2020; Trillos et al., 2020). As new surrogate models of PDEs, operator learning methods and theories have been studied in (Bhattacharya et al., 2021; Li et al., 2020b; Kovachki et al., 2021), which play essential roles in reducing the computational complexity of Bayesian inverse methods. The infinite-dimensional Bayesian inference method proposed for solving inverse problems of PDEs has been validated to be useful for some machine learning problems. Reciprocally, machine learning techniques can also contribute to the generalization of the infinite-dimensional Bayesian theory. Generally speaking, there are three main research directions for infinite-dimensional Bayesian analysis methods: designing prior probability measures on infinite-dimensional spaces, constructing appropriate noise models, and extracting information from posterior measures efficiently. In the following, we review some typical works along these three research directions and point out their essential difficulties in applications.

Regarding prior measures defined on the infinite-dimensional space, the Gaussian measure is prevalent based on its abundant theoretical studies in the field of stochastic PDEs (Prato and Zabczyk, 2014). Related studies on employing Gaussian prior measures can be found in (Agapiou et al., 2014; Bui-Thanh et al., 2013; Bui-Thanh and Nguyen, 2016; Cotter et al., 2009; Jia et al., 2021b; Wang et al., 2018). To characterize the discontinuity of the function parameters, the Besov type prior measure has been proposed (Lassas et al., 2009; Dashti et al., 2012; Jia et al., 2016). Recently, the well-posedness of the Bayesian inverse method under prior measures with exponential tails has been analyzed in (Hosseini and Nigam, 2017). Besides, starting from the seminal work (Knapik et al., 2011), the posterior consistency and contraction rates have been analyzed in detail in a series of works (Agapiou et al., 2013; Giordano and Nickl, 2020; Jia et al., 2021a; Kekkonen et al., 2016; Szabó et al., 2015; Vollmer, 2013), which convey a general understanding of what types of priors make a Bayesian nonparametric method effective. However, as illustrated in (Adler and Öktem, 2018; Arridge et al., 2019), most of these priors need to be pre-designed by handcraft and much attention has been paid to characterize families of priors to ensure posterior consistency and good convergence rates. This makes these priors more often chosen as a regularizer rather than essentially improving the quality of the final output (Arridge et al., 2019). In addition, although the hand-crafted infinite-dimensional prior measure encodes some intuitive ideas of the prior information, it can only reflect some rough prior information, e.g., the function parameter is smooth, changes slowly along certain directions, and may have sharp edges. Thus, it is desirable to adaptably encode a prior specifically suitable for a particular input in data-driven manners (Adler and Öktem, 2018).

For the noise model, the Gaussian measure is frequently used in the research literature, e.g., the Gaussian noise setting is employed in (Bui-Thanh et al., 2013; Bui-Thanh and Nguyen, 2016; Jia et al., 2018; Wang et al., 2018). By characterizing the model error as a Gaussian random variable, a Bayesian method with approximate error can be constructed (Kaipio and Somersalo, 2005; Kaipio et al., 2019). A natural generalization was proposed

in (Jia et al., 2019), which employed a mixture of Gaussian distributions to characterize model noises, making it a better fit for a broader range of noise types due to the universe approximation capability of the mixture of Gaussian for general distributions. Through a learning process, the model error information was encoded into the parameters of Gaussian mixture distributions, which improved the estimated quality by only using an approximate forward solver. However, the method proposed in (Jia et al., 2019) only provides a maximum a posterior estimate, which can be seen as an incomplete Bayesian approach. Furthermore, they have not considered more general and practical non-independently and non-identically distributed (non-i.i.d.) noises.

Furthermore, extracting information from the posterior measure is one of the essential issues for employing Bayes' method. Sampling algorithms, such as the Markov chain Monte Carlo, are often employed. The preconditioned Crank–Nicolson algorithm defined on some infinite-dimensional space has been proposed (Beskos et al., 2008) to ensure the robustness of the convergence speed under mesh refinement, see also (Cotter et al., 2013). Then, multiple dimension-independent Markov chain Monte Carlo type sampling algorithms have been proposed (Agapiou et al., 2017; Beskos et al., 2015; Cui et al., 2016; Feng and Li, 2018). However, as illustrated in (Arridge et al., 2019), the current sampling algorithms still face the critical issue of computational efficiency, especially for large-scale inverse problems.

Under the finite-dimensional setting, variational inference (VI) methods have been widely studied in machine learning (Zhang et al., 2018) to reduce the computational burden of Markov chain Monte Carlo type sampling algorithms. Compared to finite-dimensional problems, however, infinite-dimensional problems have been much less studied for VI methods. When the approximate measures are restricted to be Gaussian, a novel Robbins-Monro algorithm was developed in (Pinski et al., 2015a,b) from a calculus-of-variations viewpoint. It was shown in (Sun et al., 2019) that the Kullback-Leibler (KL) divergence between the stochastic processes is equal to the supremum of the KL divergence between the measures restricted to finite marginals. Meanwhile, they developed a VI method for functions parameterized by Bayesian neural networks. Under the classical mean-field assumption, a general VI framework defined on separable Hilbert space was proposed recently in (Jia et al., 2021b). A function space particle optimization method, including the Stein variational gradient descent, was developed in (Wang et al., 2019) to solve the particle optimization directly in the space of functions. Infinite-dimensional Stein variational gradient descent with preconditioning operators has further been proposed in (Jia et al., 2022), which provides a detailed mathematical analysis in some separable Hilbert space. These VI methods defined on infinite-dimensional space have been proposed to train deep neural networks (DNNs) or solve inverse problems of PDEs, which indeed ameliorate the computational complexity to a certain extent. However, for solving inverse problems of PDEs, many forward and adjoint PDEs still need to be solved, which is time-consuming. Furthermore, as illustrated in (Arridge et al., 2019), these VI methods also require manual pre-design of priors with explicit forms, which are not always sufficiently flexible and adaptable to faithfully deliver complicated prior knowledge underlying the investigated problem.

Through the above review and discussions, we summarize the main points as follows:

1. To more efficiently and adaptably encode prior information (not just for regularizing), machine learning methods with data-driven manners are expected to be involved in the infinite-dimensional setting.

2. The mixture of Gaussian noise modeling strategies has been introduced to improve the noise fitting capability of the problem by current research, but they are still restrictive on i.i.d. noise types, while more practical non-i.i.d. noise cases are expected to be considered.

3. Although infinite-dimensional VI (IDVI) methods have been proposed recently, time-consuming iterations are still inevitably required. More efficient regimes are thus required to be investigated. Besides, it is highly expected for current IDVI methods to flexibly and faithfully specify prior information directly from data rather than commonly adopted hand-crafted manners.

To address these issues, inspired by recent investigations on image denoising and restorations (Yue et al., 2019, 2020), we aim to propose a general IDVI method that intends to integrate prior information learning, noise information learning, and posterior mean and covariance learning into a unified machine learning framework. Unlike image denoising and restoration problems, it is expensive to generate high-quality training datasets for many inverse problems of PDEs. Therefore, it is essential for us to incorporate model information with the data-driven machine learning framework.

In our opinion, the models contain two aspects: the physical model and the probabilistic model. For the physical model, we mean the PDEs that encode our knowledge of the forward process. For the probabilistic model, we indicate the elements of Bayesian methods, such as the prior and noise measures, which encode the structures we employ to solve the inverse problems of PDEs. In this work, we focus on linear inverse problems and intend to construct a generating model (see (Kingma, 2017; Yue et al., 2019, 2020) for examples) that could incorporate the constraint of the two models mentioned above. Explicitly speaking, we employ the mean-field IDVI theory to derive the general forms of the approximate posterior measure that provides the probabilistic model. For using the mean-field IDVI theory, we assume the noises of data are non-i.i.d. in statistics, and the prior measure is Gaussian with the synthetic background truth as the mean function. It should be pointed out that the mean-field IDVI theory here only provides a probabilistic model but not a practical algorithm since the synthetic background truth is not known in the inference stage. The probabilistic model provides the general structure of our methods (see Subsections 2.2 and 2.3 for details), then we parameterize the parameters (some parameters are functions) of the approximate posterior measure by some well-designed DNNs that incorporate the physical model in some specially designed layers (see Subsections 2.3 and 3.1 for details). Through the training process, the prior and noise information will be encoded into the parameters of neural networks, i.e., the general forms of the posterior measure are determined through IDVI theory but the function- and vector-valued parameters are learned from training datasets [1]. During the inference stage, there will be no explicit Bayesian formulation. We only need to directly feed the new data to the trained deep neural network, and then we will easily obtain the parameters of the approximate posterior measure. The prior and noise

1. Compared with the IDVI theory (Jia et al., 2021b), the current work focuses on constructing a new model- and data-driven method that is well-defined in an infinite-dimensional space. No explicit prior measures are needed, and the prior information is encoded in the parameters of DNNs. The likelihood is derived from non-i.i.d. noises assumption and parameterized by DNNs. At last, the posterior can be explicitly obtained without the need for any iterations at the inference stage.

information are then not required to be encoded in explicit prior and noise measures with specific parameters but encoded in the parameters of the DNNs with probabilistic model constraint. Numerical examples are given to illustrate the flexibility and effectiveness of the proposed approach. In summary, this work mainly contains four contributions:

1. Inspired by the investigations on generative models in the finite-dimensional setting (Kingma, 2017; Yue et al., 2019, 2020), we propose a general Bayesian generative model on infinite-dimensional space named as **V**ariational **I**nverting **Net**work (VINet), which is naturally derived from the analysis on the IDVI method for linear inverse problems with non-i.i.d. noises. Due to the general assumptions on the noises and priors, the proposed VINet integrates noise information learning, prior information learning, and posterior statistical estimation into a unified machine learning framework.

2. Instead of only focusing on the interested function parameters as conventional works like deep learning for elastic source imaging (Yoo et al., 2018) and deep Bayesian inversion (Adler and Öktem, 2018), the proposed VINet provides uncertainty information both on the function parameters and on the random noises, allowing the noise and function parameter estimations to be mutually ameliorated during the inverse computational stage.

3. By imposing equivalence conditions on the prior and the approximate posterior measure, we derive the evidence lower bound under the infinite-dimensional setting. A method of amortized variational inference is developed to train all variables in the generative model. The model can then be explicitly used in the inference stage to readily achieve the posterior information of any noisy data without the time-consuming iteration process, making it more efficient for the inverse problems of PDEs than the conventional paradigms illustrated in (Dashti and Stuart, 2017).

4. We provide a specific parametric strategy by DNNs, which has been proven to satisfy the requirements of the general inference theory. Based on the proposed parametric strategy and the general abstract theory, we design a rational specific structure of VINet. Two numerical examples of typical inverse problems of PDEs are given to substantiate the effectiveness of the proposed method.

The rest of the paper is outlined as follows. In Section 2, the general theory is presented. Specifically, in Subsection 2.1, the general settings of the employed Bayesian inference model is given. In Subsection 2.2, the Bayesian inference model is analyzed under the mean-field based infinite-dimensional VI framework. In Subsection 2.3, we construct the general Bayesian generative model (i.e., VINet), derive the amortized variational inference method, and provide a simple preliminary theoretical analysis. In Subsection 2.4, we introduce a specific parametric strategy that can parameterize the posterior. In Section 3, two numerical examples are presented to illustrate the effectiveness of the proposed method. In particular, in Subsection 3.1, we design a specific network structure according to our general theory, and then apply the proposed VINet to a simple toy smoothing model in Subsection 3.2. We then demonstrate a more realistic example that is an inverse source problem governed

by the Helmholtz equation in Subsection 3.3. In Section 4, we summarize our results and propose some directions for future research.

2. Variational Inverting Network

In this section, we construct the general inference model named VINet, which is well-defined on an infinite-dimensional space. Since the overall logical structure is a bit complex, we plan to give a general illustration before going into the details. In Figure 1, we exhibit the overall structure of the present work. Generally speaking, the method can be divided into two stages: (a) Learning stage, and (b) Inference stage.

In the learning stage, we formulate Bayes' formula (see Subsection 2.1) by assuming the noises are non-i.i.d. random variables and the prior measure of the function parameter is Gaussian. In particular, the mean function of the Gaussian measure of each training sample is set to its specific synthetic truth (these measures are thus different for different training samples), which is available in the training stage but not in the inference stage. Then, relying on the infinite-dimensional VI framework, we derive the approximate posterior measures (see Subsection 2.2), which encode the information of all noisy samples and their own specific synthetic truths. One of the key points of the present work is that the approximate posterior measures of all single training data pairs are then parameterized by a carefully designed deep neural network containing a rough physical model (details can be found in Subsection 2.3). Through training on all of the individual learning data pairs, the parameters of the deep neural network are expected to extract their underlying common prediction principle for the prior information.

Once the learning stage is complete, it will be convenient for us to make inferences when new measurement data is given. As illustrated in Figure 1, the new data (there will be no synthetic truth) will be fed to the trained VINet. Then, the trained VINet will generate the posterior information, such as the mean and variance functions.

2.1 The Bayesian inference method

Let $\mathcal{H}_u$ be a separable Hilbert space, N_d be a positive integer, and $H : \mathcal{H}_u \to \mathbb{R}^{N_d}$ be a bounded linear operator. Consider the model

$$\boldsymbol{d} = Hu + \boldsymbol{\epsilon}, \tag{2.1}$$

where $\boldsymbol{d}$ and $\boldsymbol{\epsilon} \in \mathbb{R}^{N_d}$ represent the measurement data and the random noise, respectively. Denote by $\mathcal{C}_0$ a symmetric, positive definite, and trace class operator defined on $\mathcal{H}_u$. Let (e_k, α_k) be an eigen-system of the operator $\mathcal{C}_0$ such that $\mathcal{C}_0 e_k = \alpha_k e_k$. Without loss of generality, we may assume that the eigenvectors $\{e_k\}_{k=1}^{\infty}$ are orthonormal, the eigenvalues $\{\alpha_k\}_{k=1}^{\infty}$ are sorted in a descending order, and the summation of eigenvalues is finite $\sum_{k=1}^{\infty} \alpha_k < \infty$. It follows from Subsection 2.4 in (Dashti and Stuart, 2017) that we have

$$\mathcal{C}_0 = \sum_{k=1}^{\infty} \alpha_k e_k \otimes e_k. \tag{2.2}$$

For the parameter u and the noise $\boldsymbol{\epsilon}$, we assume

$$u \sim \mu_0^u := \mathcal{N}(\bar{u}_0, \mathcal{C}_0), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \Sigma), \tag{2.3}$$

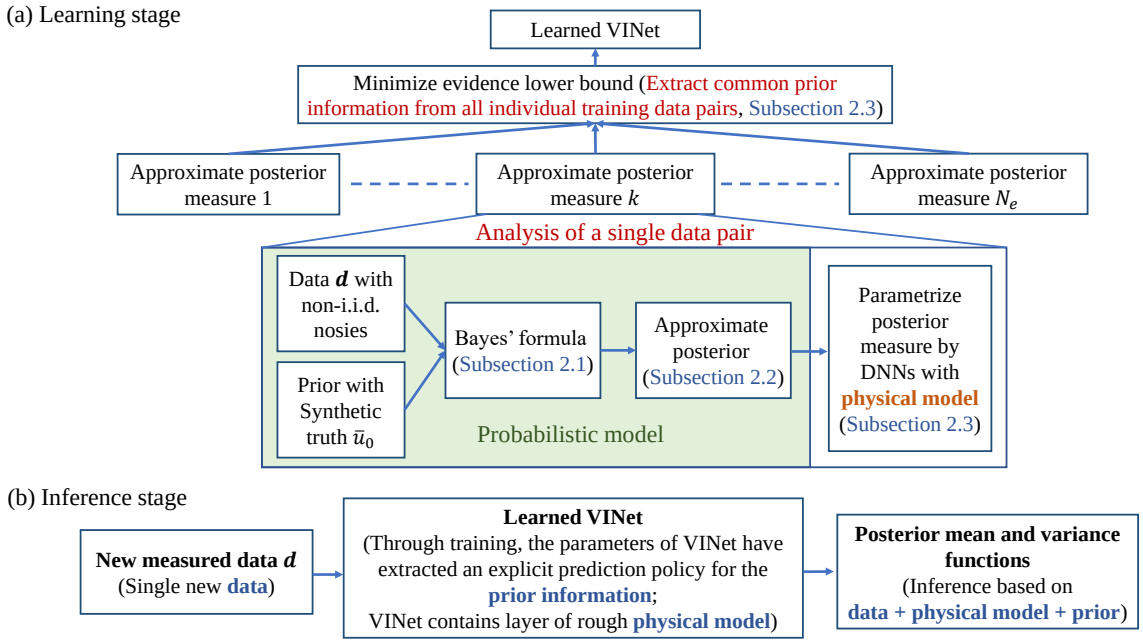

Figure 1: Overall structure of Section 2.

where $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_{N_d})$ with $\{\sigma_i\}_{i=1}^{N_d}$ being a series of positive real numbers. Here, $\mathcal{N}(v, \mathcal{C})$ stands for a Gaussian measure with mean v and covariance operator $\mathcal{C}$. Clearly, $\mathcal{N}(\bar{u}_0, \mathcal{C}_0)$ denotes a Gaussian measure defined on $\mathcal{H}_u$ with mean $\bar{u}_0 \in \mathcal{H}_u$ and covariance operator $\mathcal{C}_0$, and $\mathcal{N}(0, \Sigma)$ is a Gaussian measure defined on $\mathbb{R}^{N_d}$. It is worth mentioning that the mean function $\bar{u}_0$ as indicated in Figure 1 is set to be the synthetic truth. By our understanding, the synthetic truth is similar to, but not exactly the same as, the background truth (see the following Remark 1). The background truth can hardly be obtained (Yue et al., 2019; Davoudi et al., 2019). Since the synthetic truth is set to be the mean function, we usually take the magnitude of the covariance operator $\mathcal{C}_0$ to be small, which reflects the deviation of the synthetic truth from the background truth. For one specific form of the operator $\mathcal{C}_0$, please see Subsection 2.4 and Section 3.

Remark 1 *When solving inverse problems, it is usually difficult to obtain absolutely correct background true parameters as a training dataset. For example, in geophysical applications, the "Marmousi" model, as a typical synthetic model (Haber and Tenorio, 2003), is frequently used for training. Therefore, we may choose the simulated background truth as the prior mean function, and the covariance operator can account for the uncertainty information due to the inaccuracy of the training dataset.*

For the additive noise model (2.1), the data $\boldsymbol{d}$ given u are distributed according to the following translated distribution of $\boldsymbol{\epsilon}$:

$$\boldsymbol{d}|u \sim \mathcal{N}(Hu, \Sigma). \tag{2.4}$$

Usually, we assume that Σ is known a priori when making inferences on the infinite-dimensional parameter u. However, similar to studies such as in (Dunlop et al., 2017;

Jin and Zou, 2010), we assume that the parameters $\{\sigma_i\}_{i=1}^{\infty}$ are unknown random variables, which are called hyper-parameters and also need to be estimated. The key elements of an abstract inverse problem can then be summarized as follows.

- **Noisy data**: The data vector $\boldsymbol{d} \in \mathbb{R}^{N_d}$ is obtained from $\boldsymbol{d} = Hu^\dagger + \boldsymbol{\epsilon}$, where $u^\dagger$ is the background truth and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \Sigma)$ with $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_{N_d})$;
- **Inverse problem**: Based on the noisy data $\boldsymbol{d}$, the goal is to find estimates of u and the hyper-parameters $\boldsymbol{\sigma} := \{\sigma_1, \ldots, \sigma_{N_d}\}$.

We assume that the hyper-parameters in Σ satisfy

$$\boldsymbol{\sigma} \sim \mu_0^{\boldsymbol{\sigma}} := \prod_{i=1}^{N_d} \mathrm{IG}\left(\alpha_i^0, \beta_i^0\right), \tag{2.5}$$

where $\alpha_i^0 > 1$ and $\beta_i^0 > 0$ $(i = 1, \ldots, N_d)$ are hyper-parameters that will be specified in numerical examples, and $\mathrm{IG}(\alpha_i^0, \beta_i^0)$ denotes the inverse Gamma distribution with parameters α_i^0 and β_i^0 $(i = 1, \ldots, N_d)$. Here, the inverse Gamma distribution is employed since it is a conjugate prior measure that yields the same form of approximate posterior measure (see Subsection 2.2). Conjugate prior measures are widely used in the studies of inverse problems and machine learning (Kekkonen et al., 2016; Zhang et al., 2018; Yue et al., 2020), which makes it convenient to construct computational methods.

In order to construct an appropriate Bayes' formula, let us introduce the prior probability measure

$$\mu_0 := \mu_0^u \otimes \mu_0^{\boldsymbol{\sigma}} \tag{2.6}$$

and the potential function

$$\Phi(u, \boldsymbol{\sigma}; \boldsymbol{d}) := \frac{1}{2}\|\boldsymbol{d} - Hu\|_\Sigma^2 + \frac{1}{2}\log(\det \Sigma), \tag{2.7}$$

where $\|\cdot\|_\Sigma := \|\Sigma^{-1/2}\cdot\|$ and $\det\Sigma$ is the determinant of Σ. Since H is a bounded linear operator, it is easy to note that $\Phi(u, \boldsymbol{\sigma}; \boldsymbol{d})$ is continuous with respect to the variables $u, \boldsymbol{\sigma}$, and $\boldsymbol{d}$. Assume $\boldsymbol{d}, \boldsymbol{d}_1$, and $\boldsymbol{d}_2$ are included in a ball of $\mathbb{R}^{N_d}$ with radius r and centered at the origin. Following simple calculations, we have

$$\Phi(u, \boldsymbol{\sigma}; \boldsymbol{d}) = \frac{1}{2}\|\boldsymbol{d} - Hu\|_\Sigma^2 + \frac{1}{2}\log(\det\Sigma) \geq \frac{1}{2}\log\left(\prod_{k=1}^{N_d}\sigma_k\right) = \frac{1}{2}\sum_{k=1}^{N_d}\log(\sigma_k) \tag{2.8}$$

and

$$\begin{aligned} |\Phi(u, \boldsymbol{\sigma}; \boldsymbol{d}_1) - \Phi(u, \boldsymbol{\sigma}; \boldsymbol{d}_2)| &= \frac{1}{2}|\|\boldsymbol{d}_1 - Hu\|_\Sigma^2 - \|\boldsymbol{d}_2 - Hu\|_\Sigma^2| \\ &\leq \frac{1}{2}\|\boldsymbol{d}_1 + \boldsymbol{d}_2 - 2Hu\|_\Sigma \|\boldsymbol{d}_1 - \boldsymbol{d}_2\|_\Sigma \\ &\leq \left(1 + \sum_{k=1}^{N_d}\sigma_k^{-1/2}\right)\sum_{\ell=1}^{N_d}\sigma_\ell^{-1/2}(r + C\|u\|_{\mathcal{H}_u})\|\boldsymbol{d}_1 - \boldsymbol{d}_2\|, \end{aligned} \tag{2.9}$$

where C is a constant larger than the operator norm of H. Noting

$$\exp\left(-\sum_{k=1}^{N_d}\log\sigma_k\right)=\prod_{k=1}^{N_d}\sigma_k^{-1}\in L^1_{\mu_0}(\mathcal{H}_u\times\mathbb{R}^+;\mathbb{R}) \tag{2.10}$$

and

$$\prod_{k=1}^{N_d}\sigma_k^{-1}\left(1+\sum_{k=1}^{N_d}\sigma_k^{-1/2}\right)^2\left(\sum_{\ell=1}^{N_d}\sigma_\ell^{-1/2}\right)^2(r+C\|u\|_{\mathcal{H}_u})^2\in L^1_{\mu_0}(\mathcal{H}_u\times\mathbb{R}^+;\mathbb{R}), \tag{2.11}$$

we know that Assumption 1 and the conditions of Theorems 15–16 in (Dashti and Stuart, 2017) are satisfied, which provide the following Bayes' formula defined on infinite-dimensional space:

$$\frac{d\mu^{\boldsymbol{d}}}{d\mu_0}(u,\boldsymbol{\sigma})=\frac{1}{Z_{\boldsymbol{d}}}\det(\Sigma)^{-1/2}\exp\left(-\frac{1}{2}\|\boldsymbol{d}-Hu\|_\Sigma^2\right), \tag{2.12}$$

where $\mu^{\boldsymbol{d}}$ represents the posterior probability measure and

$$Z_{\boldsymbol{d}}=\int_{(\mathbb{R}^+)^{N_d}}\int_{\mathcal{H}_u}\det(\Sigma)^{-1/2}\exp\left(-\frac{1}{2}\|\boldsymbol{d}-Hu\|_\Sigma^2\right)\mu_0^u(du)\mu_0^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}). \tag{2.13}$$

For the reader's convenience, we list Assumption 1 and Theorems 15–16 of (Dashti and Stuart, 2017) in Subsection 5.1 of the Appendix.

2.2 The infinite-dimensional variational inference method

Before constructing the VINet, we provide a discuss of the mean-field assumption based infinite-dimensional variational inference (IDVI) theory developed in (Jia et al., 2021b). A brief introduction is given in Subsection 5.2 of the Appendix.

In Subsection 2.1, we have not introduced hyper-parameters for the prior measure μ_0^u of the function parameter. Hence, we can choose the prior measure μ_0^u as the reference probability measure required in Assumption 8 in (Jia et al., 2021b). Using the mean-field approximation, i.e., assuming the parameters u and $\boldsymbol{\sigma}$ to be independent random variables, we introduce an approximate probability measure $\nu(du,d\boldsymbol{\sigma})=\nu^u(du)\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma})$ with

$$\frac{d\nu^u}{d\mu_0^u}(u)=\frac{1}{Z_u}\exp\left(-\Phi_u(u)\right) \tag{2.14}$$

and

$$\frac{d\nu^{\boldsymbol{\sigma}}}{d\mu_0^{\boldsymbol{\sigma}}}(\boldsymbol{\sigma})=\frac{1}{Z_{\boldsymbol{\sigma}}}\exp\left(-\Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma})\right), \tag{2.15}$$

where $\Phi_u(\cdot)$ and $\Phi_{\boldsymbol{\sigma}}(\cdot)$ are two potential functions required to be calculated explicitly, and

$$Z_u=\mathbb{E}^{\mu_0^u}\left[\exp(-\Phi_u(u))\right],\quad Z_{\boldsymbol{\sigma}}=\mathbb{E}^{\mu_0^{\boldsymbol{\sigma}}}\left[\exp(-\Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}))\right].$$

Here, $\mathbb{E}^\mu$ represents taking expectation with respect to the probability measure μ.

The VI method essentially needs to solve the optimization problem

$$\arg\min_{\nu^u \in \mathcal{A}_u, \nu^{\boldsymbol{\sigma}} \in \mathcal{A}_{\boldsymbol{\sigma}}} D_{\mathrm{KL}}\left(\nu || \mu^{\boldsymbol{d}}\right), \tag{2.16}$$

where $\mathcal{A}_u$ and $\mathcal{A}_{\boldsymbol{\sigma}}$ are two appropriate spaces of probability measure, and $D_{\mathrm{KL}}\left(\nu||\mu^{\boldsymbol{d}}\right)$ is the Kullback-Leibler (KL) divergence defined as follows:

$$D_{\mathrm{KL}}(\nu||\mu^{\boldsymbol{d}}) = \int_{\mathcal{H}} \log\left(\frac{d\nu}{d\mu^{\boldsymbol{d}}}(x)\right)\frac{d\nu}{d\mu^{\boldsymbol{d}}}(x)\mu^{\boldsymbol{d}}(dx) = \mathbb{E}^{\mu^{\boldsymbol{d}}}\left[\log\left(\frac{d\nu}{d\mu^{\boldsymbol{d}}}(x)\right)\frac{d\nu}{d\mu^{\boldsymbol{d}}}(x)\right].$$

Here $0\log 0 = 0$ is used as a convention. For the finite-dimensional theory (e.g., Chapter 10 in (Bishop, 2006)), no additional assumptions need to be made on $\mathcal{A}_u$ and $\mathcal{A}_{\boldsymbol{\sigma}}$. However, special attention must be paid to $\mathcal{A}_u$ and $\mathcal{A}_{\boldsymbol{\sigma}}$ in the infinite-dimensional theory (Jia et al., 2021b).

Define $T_N^u = \{u \,|\, 1/N \le \|u\|_{\mathcal{Z}_u} \le N\}$ with $\mathcal{Z}_u$ being a Hilbert space that is embedded in $\mathcal{H}_u$ and satisfies $\sup_N \mu_0^u(T_N^u) = 1$. Denote $T_N^{\boldsymbol{\sigma}} = \{\boldsymbol{\sigma} \,|\, 1/N \le \sigma_i \le N \text{ for } i = 1, \ldots, N_d\}$ that obviously satisfies $\sup_N \mu_0^{\boldsymbol{\sigma}}(T_N^{\boldsymbol{\sigma}}) = 1$. Let

$$R_u^1 = \left\{\Phi_u \,\middle|\, \sup_{u \in T_N^u} \Phi_u(u) < \infty \text{ for all } N > 0\right\},$$

$$R_u^2 = \left\{\Phi_u \,\middle|\, \int_{\mathcal{H}_u} \exp\left(-\Phi_u(u)\right)\max(1, \|u\|_{\mathcal{H}_u}^2)\mu_0^u(du) < \infty\right\},$$

$$R_{\boldsymbol{\sigma}}^1 = \left\{\Phi_{\boldsymbol{\sigma}} \,\middle|\, \sup_{\boldsymbol{\sigma} \in T_N^{\boldsymbol{\sigma}}} \Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}) < \infty \text{ for all } N > 0\right\},$$

$$R_{\boldsymbol{\sigma}}^2 = \left\{\Phi_{\boldsymbol{\sigma}} \,\middle|\, \int_{(\mathbb{R}^+)^{N_d}} \exp\left(-\Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma})\right)\max(1, a(\epsilon, \boldsymbol{\sigma}))\mu_0^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) < \infty, \text{ for } \epsilon \in [0, \epsilon_0)\right\},$$

where $a(\epsilon, \boldsymbol{\sigma}) := \sum_{i=1}^{N_d} \max\left(\sigma_i^{\epsilon}, \exp(\epsilon/\sigma_i)\right)$ and ϵ_0 is a fixed positive number. Usually, we may choose $\epsilon_0 \ll \min_{1 \le i \le N_d}\{\alpha_i^0, \beta_i^0\}$, where $\{\alpha_i^0, \beta_i^0\}_{i=1}^{N_d}$ are given in (2.5). Define

$$\mathcal{A}_u = \left\{\nu^u \in \mathcal{M}(\mathcal{H}_u) \,\middle|\, \begin{array}{l}\nu^u \text{ is equivalent to } \mu_0^u \text{ with (2.14) holding true,} \\ \text{and } \Phi_u \in R_u^1 \cap R_u^2\end{array}\right\},$$

$$\mathcal{A}_{\boldsymbol{\sigma}} = \left\{\nu^{\boldsymbol{\sigma}} \in \mathcal{M}(\mathcal{H}_{\boldsymbol{\sigma}}) \,\middle|\, \begin{array}{l}\nu^{\boldsymbol{\sigma}} \text{ is equivalent to } \mu_0^{\boldsymbol{\sigma}} \text{ with (2.15) holding true,} \\ \text{and } \Phi_{\boldsymbol{\sigma}} \in R_{\boldsymbol{\sigma}}^1 \cap R_{\boldsymbol{\sigma}}^2\end{array}\right\}.$$

Remark 2 *The definitions of R_u^1 and $R_{\boldsymbol{\sigma}}^1$ are slightly different from the corresponding definitions of $R_j^1 (j = 1, \ldots, M)$ introduced in (Jia et al., 2021b). The sets $R_k^1 (k = u, \boldsymbol{\sigma})$ defined here enlarge the original definition (without uniform bound), which makes the theory more appropriate. For completeness, we provide a short illustration of the modified general theory in Subsection 5.2 of the Appendix.*

Next, we present the key theorem that leads to an iterative algorithm.

Theorem 3 *Assume that the prior measure μ_0, noise measure, and posterior measure are defined in (2.6), (2.5), and (2.12), respectively. Let $\Phi(u, \boldsymbol{\sigma}; \boldsymbol{d})$ be defined in (2.7). If $\mathcal{A}_u$ and $\mathcal{A}_{\boldsymbol{\sigma}}$ are defined as above, then the problem (2.16) possesses a solution $\nu(du, d\boldsymbol{\sigma}) = \nu^u(du)\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma})$, which satisfies*

$$\frac{d\nu}{d\mu_0}(u, \boldsymbol{\sigma}) \propto \exp\left(-\Phi_u(u) - \Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma})\right), \tag{2.17}$$

where

$$\Phi_u(u) = \int_{(\mathbb{R}^+)^{N_d}} \Phi(u, \boldsymbol{\sigma}; \boldsymbol{d})\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) + \mathit{Const}, \tag{2.18}$$

$$\Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}) = \int_{\mathcal{H}_u} \Phi(u, \boldsymbol{\sigma}; \boldsymbol{d})\nu^u(du) + \mathit{Const}. \tag{2.19}$$

Here, "Const" denotes some general constant. Futhermore, we have

$$\nu^u(du) \propto \exp\left(-\Phi_u(u)\right)\mu_0^u(du), \quad \nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) \propto \exp\left(-\Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma})\right)\mu_0^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}).$$

To avoid a possible distraction from the presentation of the work, the proof of this theorem is given in Subsection 5.3 of the Appendix. In the following, we denote $\Phi(u, \boldsymbol{\sigma})$ instead of $\Phi(u, \boldsymbol{\sigma}; \boldsymbol{d})$ when there is no ambiguity from the context.

Remark 4 *In the general theory, the space of parameters is assumed to be a separable Hilbert space. The parameter $\boldsymbol{\sigma}$ resides in $(\mathbb{R}^+)^{N_d}$, which is not a Hilbert space. However, as stated in Remark 15 in (Jia et al., 2021b), a simple transformation can be adopted, e.g., $\sigma'_k = \log \sigma_k$ $(k = 1, \ldots, N_d)$. Then, the new parameter $\boldsymbol{\sigma}' = \{\sigma'_1, \ldots, \sigma'_{N_d}\}$ belongs to $\mathbb{R}^{N_d}$, which is a Hilbert space. All of the statements and verifications of $\boldsymbol{\sigma}$ can be equivalently transformed to the new variable $\boldsymbol{\sigma}'$. Therefore, we still use the parameter $\boldsymbol{\sigma}$ for simplicity of notation. By the above discussion, the reader may intuitively understand the connection between the finite- and infinite-dimensional theories.*

Using Theorem 3, we can present a classical iterative algorithm when the parameter u belongs to some infinite-dimensional separable Hilbert space $\mathcal{H}_u$.

Calculate $\Phi_u(u)$. A direct application of (2.18) yields

$$\begin{aligned}\Phi_u(u) &= \frac{1}{2}\int_{(\mathbb{R}^+)^{N_d}} \left(\|\boldsymbol{d} - Hu\|_\Sigma^2 + \sum_{k=1}^{N_d} \log \sigma_k\right) \nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) + \text{Const} \\ &= \frac{1}{2}\int_{(\mathbb{R}^+)^{N_d}} (\boldsymbol{d} - Hu)^T \Sigma^{-1} (\boldsymbol{d} - Hu)\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) + \text{Const}.\end{aligned} \tag{2.20}$$

Denoting

$$\Sigma_{\text{inv}}^* = \text{diag}\left(\mathbb{E}^{\nu^{\boldsymbol{\sigma}}}[\sigma_1^{-1}], \ldots, \mathbb{E}^{\nu^{\boldsymbol{\sigma}}}[\sigma_{N_d}^{-1}]\right), \tag{2.21}$$

we have

$$\Phi_u(u) = \frac{1}{2}\|\boldsymbol{d} - Hu\|^2_{\Sigma^*_{\text{inv}}} + \text{Const.} \tag{2.22}$$

Hence,

$$\frac{d\nu^u}{d\mu_0^u}(u) \propto \exp\left(-\frac{1}{2}\|\boldsymbol{d} - Hu\|^2_{\Sigma^*_{\text{inv}}}\right), \tag{2.23}$$

which implies that the approximate posterior measure of u is a Gaussian measure

$$\nu^u = \mathcal{N}(\bar{u}_p, \mathcal{C}_p), \tag{2.24}$$

where

$$\mathcal{C}_p^{-1} = H^*\Sigma^*_{\text{inv}}H + \mathcal{C}_0^{-1}, \quad \bar{u}_p = \mathcal{C}_p(H^*\Sigma^*_{\text{inv}}\boldsymbol{d} + \mathcal{C}_0^{-1}\bar{u}_0). \tag{2.25}$$

Calculate $\Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma})$. Following (2.19), we have

$$\begin{aligned}
\Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}) &= \frac{1}{2}\int_{\mathcal{H}_u}\left(\|\boldsymbol{d} - Hu\|^2_{\Sigma} + \sum_{k=1}^{N_d}\log\sigma_k\right)\nu^u(du) + \text{Const} \\
&= \frac{1}{2}\int_{\mathcal{H}_u}(\boldsymbol{d} - Hu)^T\Sigma^{-1}(\boldsymbol{d} - Hu)\nu^u(du) + \frac{1}{2}\sum_{k=1}^{N_d}\log\sigma_k + \text{Const} \\
&= \sum_{k=1}^{N_d}\frac{1}{2}\left(\int_{\mathcal{H}_u}(d_k - (Hu)_k)^2\nu^u(du)\frac{1}{\sigma_k} + \log\sigma_k\right) + \text{Const.}
\end{aligned} \tag{2.26}$$

Hence,

$$\frac{d\nu^{\boldsymbol{\sigma}}}{d\mu_0^{\boldsymbol{\sigma}}}(\boldsymbol{\sigma}) \propto \exp\left(-\sum_{k=1}^{N_d}\frac{1}{2}\left(\mathbb{E}^{\nu^u}[(d_k - (Hu)_k)^2]\frac{1}{\sigma_k} + \log\sigma_k\right)\right). \tag{2.27}$$

Recalling

$$\mu_0^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) \propto \prod_{k=1}^{N_d}\sigma_k^{-\alpha_k^0-1}\exp\left(-\frac{\beta_k^0}{\sigma_k}\right)d\boldsymbol{\sigma}, \tag{2.28}$$

we obtain

$$\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) \propto \prod_{k=1}^{N_d}\sigma_k^{-\alpha_k^0-\frac{3}{2}}\exp\left(-\frac{\beta_k^0 + \frac{1}{2}\mathbb{E}^{\nu^u}[(d_k - (Hu)_k)^2]}{\sigma_k}\right)d\boldsymbol{\sigma}, \tag{2.29}$$

which implies that the posterior distribution of each component of $\nu^{\boldsymbol{\sigma}}$ takes the form

$$\sigma_k \sim \text{IG}\left(\alpha_k^0 + \frac{1}{2}, \beta_k^0 + \frac{1}{2}\mathbb{E}^{\nu^u}[(d_k - (Hu)_k)^2]\right), \quad k = 1, \ldots, N_d. \tag{2.30}$$

The above illustrations confirm the feasibility of Algorithm 1, which is similar to the finite-dimensional case. However, it is worth mentioning that the infinite-dimensional formulation provides a general framework for conducting appropriate discretizations. To keep dimension-independent properties, the discretization of Algorithm 1 should be done carefully; for example, the adjoint operator H^* is usually not equal to the transpose of the discrete approximation of H. For more detailed discussions on this topic, please refer to (Jia et al., 2021b; Bui-Thanh et al., 2013; Petra et al., 2014; Wang et al., 2018; Bui-Thanh and Nguyen, 2016). A recent study on the Stein variational gradient descent algorithm, designed on the functional space for training DNNs (Wang et al., 2019), further indicates the infinite-dimensional formulation.

Algorithm 1 A classical VI algorithm

1: Give an initial guess μ_0 ($\bar{u}_0$, $\mathcal{C}_0$, p, and $\{\alpha_i^0, \beta_i^0\}_{i=1}^{N_d}$). Set the tolerance *tol* and let $k = 1$.
2: **Do**
3: Let $k = k + 1$
4: Calculate Σ_{inv}^{k*} by $\mathbb{E}^{\nu_{k-1}^{\boldsymbol{\sigma}}}[\sigma_i^{-1}](i = 1, \ldots, N_d)$
5: Update ν_k^u by

$$\mathcal{C}_p^{-1} = H^* \Sigma_{\text{inv}}^{k*} H + \mathcal{C}_0^{-1}, \quad u_k = \mathcal{C}_p (H^* \Sigma_{\text{inv}}^{k*} \boldsymbol{d} + \mathcal{C}_0^{-1} \bar{u}_0)$$

6: Calculate $\nu_k^{\boldsymbol{\sigma}}$ by

$$\nu_k^{\boldsymbol{\sigma}} = \prod_{i=1}^{N_d} \text{IG}(\alpha_i^k, \beta_i^k),$$

where

$$\alpha_i^k = \alpha_i^0 + \frac{1}{2}, \quad \beta_i^k = \beta_i^0 + \frac{1}{2} \mathbb{E}^{\nu_k^u}[(d_i - (Hu_k)_i)^2]$$

7: **Until** $\max\left(\|u_k - u_{k-1}\|/\|u_k\|, \|\boldsymbol{\sigma}_k - \boldsymbol{\sigma}_{k-1}\|/\|\boldsymbol{\sigma}_k\|\right) \leq tol$
8: Return $\nu_k^u(du)\nu_k^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma})$ as the solution.

Remark 5 *In the implementation of Algorithm 1, it is required to calculate* $\mathbb{E}^{\nu_k^u}[(d_i - (Hu_k)_i)^2]$ *for each* $i = 1, \ldots, N_d$. *Regarding these terms, we have*

$$\begin{aligned} \mathbb{E}^{\nu_k^u}[(d_i - (Hu_k)_i)^2] &= \int (\boldsymbol{d} - Hu_k)_i^2 \nu_k^u(du) + \int [H(u - u_k)]_i^2 \nu_k^u(du) \\ &= (\boldsymbol{d} - Hu_k)_i^2 + \boldsymbol{e}_i^T H \mathcal{C}_p H^* \boldsymbol{e}_i, \end{aligned} \tag{2.31}$$

where $\boldsymbol{e}_i$, $i = 1, \ldots, N_d$, *represents the standard vector basis in* $\mathbb{R}^{N_d}$ *and Proposition 1.18 in (Prato, 2006) is employed to derive the second equality. It can be seen from (2.31) that the adjoint equations, forward equations, and inverse of the operator* $\mathcal{C}_p$ *are needed with*

N_d iterations to update the parameters $\{\beta_i^k\}_{i=1}^{N_d}$. Especially, when calculating $\mathcal{C}_p u$ with u being a function (e.g., $H^*\boldsymbol{e}_i$), we need to evaluate $(H^*\Sigma_{inv}^{k*}H + \mathcal{C}_0^{-1})^{-1}u$. The operator $(H^*\Sigma_{inv}^{k*}H + \mathcal{C}_0^{-1})^{-1}$ cannot be calculated explicitly since, in discrete form, it will be a large dense matrix that is difficult to store and time-consuming to find its explicit inverse operator. Instead, we usually solve a linear equation $(H^*\Sigma_{inv}^{k*}H + \mathcal{C}_0^{-1})\tilde{u} = u$. The solution $\tilde{u}$ is the function we expected. By employing some iterative linear equation solvers, e.g., the conjugate gradient algorithm, we need to solve a lot of PDEs, i.e., for each iteration, we need to solve one forward and one adjoint equation (Jia et al., 2021b; Jin and Zou, 2010). Overall, the computational complexity will be very large when the number of measurement points N_d increases.

2.3 Variational inverting network

Let us provide the following observations of the classical IDVI algorithm:

- The entire prior and noise probability measure (general forms and all of the function- and vector-valued parameters) must be specified manually. This makes the use of the algorithm highly dependent on the subjective experiences of the practitioners.

- Under the settings of Subsections 2.1 and 2.2, the derived Algorithm 1 actually still cannot be employed for practical problems since the synthetic truth (i.e., prior mean function) is not known for new inverse problems.

- When the practitioner acquires new data, the entire iterative process should be started from the beginning to obtain a new estimate, which can be time-consuming for many applications.

Inspired by the recent work on denoising tasks under the finite-dimensional setting (Yue et al., 2019, 2020), we introduce some parametric forms of posterior probability measure by adopting DNNs. As illustrated in the introduction and Figure 1, we will rely on the general form of the posterior measures derived from the IDVI theory, which provides the probabilistic model (or called probabilistic constraint) for DNNs-based data-driven inverse methods. However, the function- and vector-valued parameters in the posterior measures will be replaced by well-designed DNNs that could incorporate the physical model into our method. Relying on this methodology, it is then hopeful that we can alleviate the problems of the classical IDVI algorithms mentioned above.

Based on the discussions in Subsection 2.2, we have obtained the general form of the approximate posterior measures, as seen in formulas (2.24) and (2.29). Therefore, we introduce the following parametric form of the approximate posterior probability measures:

$$\nu^u = \mathcal{N}(\bar{u}_p(\boldsymbol{d}; W_I), \mathcal{C}_p(\boldsymbol{d}; W_I)), \quad \nu^{\boldsymbol{\sigma}} = \prod_{i=1}^{N_d} \mathrm{IG}\left(\alpha_i(\boldsymbol{d}; W_S), \beta_i(\boldsymbol{d}; W_S)\right), \tag{2.32}$$

where $\{\alpha_i(\boldsymbol{d}; W_S)\}_{i=1}^{N_d}$, $\{\beta_i(\boldsymbol{d}; W_S)\}_{i=1}^{N_d}$ are represented by neural networks with parameters denoted by W_S, and $\{\bar{u}_p(\boldsymbol{d}; W_I), \mathcal{C}_p(\boldsymbol{d}; W_I)\}$ are represented by neural networks with parameters denoted by W_I (where $\mathcal{C}_p(\boldsymbol{d}; W_I)$ is a symmetric, positive definite, and trace class

operator). The neural networks with parameters W_S and W_I are referred to as SNet (sigma network) and INet (inverting network), respectively.

Let us provide intuitive ideas of how to incorporate physical models into the INet, which rely on some analysis of the posterior probability measures given by (2.24) and (2.29). It follows from the formula as shown in Example 6.23 of (Stuart, 2010) that the expression of the mean function (2.25) can be written into the following form:

$$\bar{u}_p = \bar{u}_p^1 + \bar{u}_p^2 = (H^*\Sigma_{\text{inv}}^* H + \mathcal{C}_0^{-1})^{-1} H^*\Sigma_{\text{inv}}^* \boldsymbol{d} + (H^*\Sigma_{\text{inv}}^* H + \mathcal{C}_0^{-1})^{-1}\mathcal{C}_0^{-1}\bar{u}_0. \tag{2.33}$$

Obviously, the posterior mean function contains two parts $\bar{u}_p^1$ and $\bar{u}_p^2$. We analyze these two parts separately and then join them to provide specific structures of the INet.

Now, let us illustrate some intuitive ideas about the data-informed part $\bar{u}_p^1$. Denote $\{e_j\}_{j=1}^{\infty}$ in (2.2) and let $\{\lambda_j\}_{j=1}^{\infty}$ be a positive sequence belonging to ℓ^2. In addition, we assume that the forward operator H can be decomposed into $S \circ F$, where

$$(Fu)(x) = \sum_{j=1}^{\infty} \lambda_j \langle u, e_j \rangle e_j(x) = \sum_{j=1}^{\infty} \lambda_j u_j e_j(x) \quad \forall\ u \in \mathcal{H}_u \tag{2.34}$$

and

$$(S \circ F)u = ((Fu)(x_1), \ldots, (Fu)(x_{N_d}))^T \quad \forall\ (x_1, \ldots, x_{N_d}) \in \mathbb{R}^{N_d}. \tag{2.35}$$

This is only needed in Theorem 6.

Theorem 6 *Assume that the forward operator H can be decomposed into $S \circ F$, where F and S are defined in (2.34) and (2.35), respectively. Then, we have*

$$\lim_{\max_{1\le i\le N_d} \beta_i^* \to 0} \|(H^*\Sigma_{\text{inv}}^* H + \mathcal{C}_0^{-1})^{-1}\mathcal{C}_0^{-1}\bar{u}_0\|_{\mathcal{H}_u} = 0, \tag{2.36}$$

where

$$\Sigma_{\text{inv}}^* = diag\left(\frac{\alpha_1^*}{\beta_1^*}, \ldots, \frac{\alpha_{N_d}^*}{\beta_{N_d}^*}\right) \tag{2.37}$$

with α_i^*, β_i^* $(i = 1, \ldots, N_d)$ being positive real numbers.

The proof is given in Subsection 5.3 of Appendix. It can be seen from Theorem 6 that the synthetic truth (prior mean) related term $\bar{u}_p^2$ will disappear as the noise goes to zero, i.e., the data $\boldsymbol{d}$ becomes to be the clean data. Hence, in order to fully extract the information of noisy data, we introduce a neural network named as DNet (denoising network) that aims to transform the noisy data into nearly clean data (data with low noise). The term $\bar{u}_p^1$ will be transformed into

$$\bar{u}_p^1 = (H^*\Sigma_{\text{inv}}^* H + \mathcal{C}_0)^{-1} H^*\Sigma_{\text{inv}}^* \text{DNet}(\boldsymbol{d}). \tag{2.38}$$

Since $(H^*\Sigma_{\text{inv}}^* H + \mathcal{C}_0)^{-1} H^*\Sigma_{\text{inv}}^*$ can be seen as a classical inverse method, we can introduce a **c**omputationally **e**fficient **c**lassical **inv**erse (CECInv) method (which may provide a quick

but noisy estimate) combined with a deep neural network called ENet (enhancing neural network). For simplicity, we call this the CECInv layer, which incorporates the physical model into our inversion method. In summary, the network structure is defined as follows:

$$\bar{u}_p^1 = \text{ENet}(\text{CECInv}(\text{DNet}(\boldsymbol{d}))). \tag{2.39}$$

Practically, the noisy data can hardly be transformed into clean data due to the limited training datasets. Considering the ill-posedness of the inverse problems, the clean data can only provide limited information to yield estimates with limited accuracy. Therefore, the synthetic truth (prior mean) related term $\bar{u}_p^2$ should also be taken into consideration.

During the training stage, we have the synthetic truth, i.e., the prior mean function $\bar{u}_0$. However, the synthetic truth is not available during the inference stage, and only the noisy data is available. Based on these considerations, we need to design a mechanism that can encode the prior information of the synthetic truth into the network parameters through the training process on these data with ground truth knowledge. Remembering the form of the term $\bar{u}_p^2$, we see that if the synthetic truth is much similar to the truth, then the prior variance should be very small. That is to say, we have

$$\bar{u}_p^2 \approx \bar{u}_0. \tag{2.40}$$

Remembering the formula (2.39), during the training stage, the noisy data is transformed into near-clean data. Then, ENet(CECInv($\cdot$)) provides an estimate that should be as similar as possible to the synthetic truth parameters in the training datasets. Obviously, we may recognize

$$\bar{u}_0 \approx \text{ENet}(\text{CECInv}(\text{DNet}(\boldsymbol{d}))). \tag{2.41}$$

Finally, considering (2.33), (2.39), (2.40), and (2.41), the network for predicting the posterior mean functions could rationally be designed as follow:

$$\bar{u}_p = \bar{u}_p^1 + \bar{u}_p^2 \approx \text{ENet}(\text{CECInv}(\text{DNet}(\boldsymbol{d}))) + \text{ENet}(\text{CECInv}(\text{DNet}(\boldsymbol{d}))), \tag{2.42}$$

which can be reduced to

$$\bar{u}_p = \text{ENet}(\text{CECInv}(\text{DNet}(\boldsymbol{d}))). \tag{2.43}$$

We incorporate the variances by adding an additional channel to the related neural networks to represent the function parameters involved in the covariance operator. From the above statements, we can see that the final structure of INet achieves our aim: the general probability model is specified by the variational inference method, the physical model is encoded in the CECInv layer, and the prior information of the training datasets is encoded in both the ENet and DNet (see formula (2.41)). For clarity, we depict a schematic diagram of INet in Figure 2. The general framework illustrated here requires no restrictions on the specific forms of the DNet, ENet, and SNet. In Section 3, we provide two numerical examples and the particular DNNs used in the present work.

After introducing the general inverse framework, named VINet (variational inverting network), we need to focus on designing algorithms that can efficiently train the involved DNNs. To formulate an optimization problem for the network parameters of INet and SNet, we need to find the variational lower bound under the infinite-dimensional setting as the loss functional to guide the DNN training. This requires the following basic assumption.

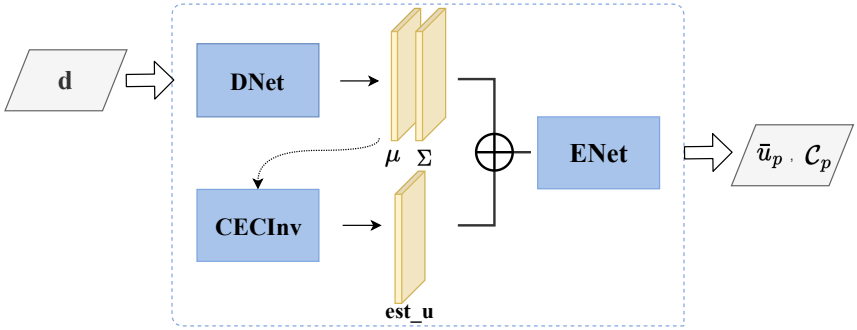

Figure 2: General structure of INet.

Assumption 7 *The approximate probability measure ν with components ν^u and $\nu^{\boldsymbol{\sigma}}$ is assumed to be equivalent to the prior probability measure μ_0, as well as the posterior probability measure $\mu^{\boldsymbol{d}}$.*

Regarding the logarithm of the marginal likelihood, we have

$$\begin{aligned}\log Z_{\boldsymbol{d}} &= \int_{\mathcal{H}_u\times(\mathbb{R}^+)^{N_d}} \log Z_{\boldsymbol{d}}\, \nu(du, d\boldsymbol{\sigma}) \\ &= \int_{\mathcal{H}_u\times(\mathbb{R}^+)^{N_d}} \log\left\{\frac{d\mu_0}{d\nu}\frac{d\nu}{d\mu^{\boldsymbol{d}}}\frac{1}{\sqrt{\det(\Sigma)}}\exp\left(-\frac{1}{2}\|\Sigma^{-1/2}(\boldsymbol{d}-Hu)\|^2\right)\right\}\nu(du, d\boldsymbol{\sigma}).\end{aligned}$$

Expanding the terms in the integrand of the above equality yields

$$\log Z_{\boldsymbol{d}} = \mathcal{L}(u, \boldsymbol{\sigma}; \boldsymbol{d}) + D_{\mathrm{KL}}(\nu||\mu^{\boldsymbol{d}}), \tag{2.44}$$

where

$$\mathcal{L}(u, \boldsymbol{\sigma}; \boldsymbol{d}) = \int \log\frac{d\mu_0}{d\nu} - \frac{1}{2}\log(\det(\Sigma)) - \frac{1}{2}\|\Sigma^{-1/2}(\boldsymbol{d}-Hu)\|^2\nu(du, d\boldsymbol{\sigma}). \tag{2.45}$$

Noting that the KL divergence between the variational approximate measure ν and the true posterior measure $\mu^{\boldsymbol{d}}$ is non-negative, we can thus get

$$\log Z_{\boldsymbol{d}} \geq \mathcal{L}(u, \boldsymbol{\sigma}; \boldsymbol{d}). \tag{2.46}$$

Since $\mathcal{L}(u, \boldsymbol{\sigma}; \boldsymbol{d})$ constitutes a lower bound of $\log Z_{\boldsymbol{d}}$, we call it evidence lower bound (ELBO) as in the finite-dimensional setting (Bishop, 2006). Under Assumption 7, we know that the optimization problems

$$\min_{W_I, W_S} D_{\mathrm{KL}}(\nu||\mu^{\boldsymbol{d}}) \tag{2.47}$$

and

$$\min_{W_I, W_D} -\mathcal{L}(u, \boldsymbol{\sigma}; \boldsymbol{d}) \tag{2.48}$$

are equivalent to each other. The ELBO can be decomposed into the following three terms:

$$\mathcal{L}(u, \boldsymbol{\sigma}; \boldsymbol{d}) = I_1 + I_2 + I_3, \tag{2.49}$$

where

$$I_1 = \int \frac{d\mu_0^u}{d\nu^u}\nu^u(du), \quad I_2 = \int \frac{d\mu_0^{\boldsymbol{\sigma}}}{d\nu^{\boldsymbol{\sigma}}}\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}), \tag{2.50}$$

$$I_3 = \int -\frac{1}{2}\log(\det(\Sigma)) - \frac{1}{2}\|\Sigma^{-1/2}(\boldsymbol{d} - Hu)\|^2\nu(du, d\boldsymbol{\sigma}). \tag{2.51}$$

Term I_1. Different from the finite-dimensional case, it is difficult to reduce the form of I_1. However, for some specific problems, the following reparameterization trick can be used to approximate it by Monte Carlo estimation:

$$I_1 \approx \frac{d\mu_0^u}{d\nu^u}(\tilde{u}), \tag{2.52}$$

where $\tilde{u} = \bar{u}_p(\boldsymbol{d}; W_I) + \mathcal{C}_p(\boldsymbol{d}; W_I)^{1/2}\eta$ with $\eta \sim \mathcal{N}(0, \mathrm{Id})$.

Term I_2. Since $\boldsymbol{\sigma}$ belongs to the finite-dimensional space, we may follow (Yue et al., 2019) and obtain

$$\begin{aligned}\int \log\frac{d\nu^{\boldsymbol{\sigma}}}{d\mu_0^{\boldsymbol{\sigma}}}\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) = \sum_{i=1}^{N_d}\Bigg\{ &\left(\alpha_i - \alpha_i^0 + 1\right)\psi(\alpha_i) + \alpha_i\left(\frac{\beta_i^0}{\beta_i} - 1\right) \\ &+ \left(\alpha_i^0 - 1\right)\left(\log\beta_i - \log\beta_i^0\right) + \left[\log\Gamma\left(\alpha_i^0 - 1\right) - \log\Gamma(\alpha_i)\right]\Bigg\},\end{aligned} \tag{2.53}$$

where $\Gamma(\cdot)$ and $\psi(\cdot)$ represent the Gamma and Degamma functions, respectively.

Term I_3. For the first term in I_3, we have

$$\begin{aligned}-\frac{1}{2}\int \log(\det(\Sigma))\nu(du, d\boldsymbol{\sigma}) &= -\frac{1}{2}\sum_{i=1}^{N_d}\int \log\sigma_i^2\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) \\ &= -\frac{1}{2}\sum_{i=1}^{N_d}\left(\log\beta_i - \psi(\alpha_i)\right).\end{aligned} \tag{2.54}$$

For the second term in I_3, a simple calculation yields

$$-\frac{1}{2}\int \|(\boldsymbol{d} - Hu)\|_\Sigma^2\nu(du, d\boldsymbol{\sigma}) = -\frac{1}{2}\int (\boldsymbol{d} - Hu)^T\Sigma_{\text{inv}}(\boldsymbol{d} - Hu)\nu^u(du), \tag{2.55}$$

where

$$\Sigma_{\text{inv}} = \operatorname{diag}\left(\mathbb{E}^{\nu^{\boldsymbol{\sigma}}}[\sigma_1^{-1}], \ldots, \mathbb{E}^{\nu^{\boldsymbol{\sigma}}}[\sigma_{N_d}^{-1}]\right) = \operatorname{diag}\left(\frac{\alpha_1}{\beta_1}, \ldots, \frac{\alpha_{N_d}}{\beta_{N_d}}\right). \tag{2.56}$$

Using (2.54)–(2.55) and the Monte Carlo estimate, we get

$$I_3 \approx -\frac{1}{2}\sum_{i=1}^{N_d}\left(\log\beta_i - \psi(\alpha_i)\right) - \frac{1}{2}(\boldsymbol{d} - H\tilde{u})^T\Sigma_{\text{inv}}(\boldsymbol{d} - H\tilde{u}), \tag{2.57}$$

where $\tilde{u} = \bar{u}_p(\boldsymbol{d}; W_I) + \mathcal{C}_p(\boldsymbol{d}; W_I)^{1/2}\eta$ with $\eta \sim \mathcal{N}(0, \mathrm{Id})$.

For a given training data set $\{\boldsymbol{d}_i, u_i\}_{i=1}^{N_e}$, where N_e is a positive integer, the following objective function, the negative ELBO on the entire training set, can be constructed to optimize the parameters of INet and SNet:

$$\min_{W_I, W_S} -\sum_{i=1}^{N_e} \mathcal{L}(u^{(i)}, \boldsymbol{\sigma}^{(i)}; \boldsymbol{d}^{(i)}), \tag{2.58}$$

where $u^{(i)}$, $\boldsymbol{\sigma}^{(i)}$, and $\boldsymbol{d}^{(i)}$ denote the corresponding variables for the i-th training pair.

Remark 8 *Here, we provide some intuitive explanations of the VINet. From formula (2.44), we know that the problems (2.47) and (2.48) are equivalent. If there is only one piece of training data, the optimization problem (2.48) will find the parameters W_I and W_S of DNNs that make the parameterized measure (2.32) similar to the approximate posterior measure calculated from the mean-field assumption based IDVI method. We can expect that the learned model from one piece of training data will have poor generalization properties. During the training stage, we actually have N_e pieces of training data and solve the minimization problem (2.58) that can be written as*

$$\min_{W_I, W_S} -\frac{1}{N_e}\sum_{i=1}^{N_e} \mathcal{L}(u^{(i)}, \boldsymbol{\sigma}^{(i)}; \boldsymbol{d}^{(i)}). \tag{2.59}$$

Hence, our training process intuitively finds common network parameters W_I and W_S that perform well on all of the training datasets on average. In this sense, we can intuitively recognize that the DNet, ENet, and SNet capture the information shared by all training datasets (prior and noise information) in some average sense. During the inference stage, there will be no synthetic truth. The trained VINet will combine the information extracted from the new data $\boldsymbol{d}$, the physical model encoded in the CECInv layer, and the prior information encoded in the DNet, SNet, and ENet to yield estimates of the function- and vector-valued parameters of the probability measure (2.32).

At the end of this subsection, we present a preliminary analysis of the estimation error of the mean function, which provides some intuitions for future study. The proof is given in Subsection 5.3 of the Appendix.

Theorem 9 *Let $u^\dagger$ be the background truth function, $\bar{u}_p$ be the estimated function given by (2.33), and $\mathbb{E}_0$ be the expectation corresponding to the distribution of the data. Then the following estimate holds:*

$$\begin{aligned}\mathbb{E}_0[\|\bar{u}_p(\boldsymbol{d}; W_I) - u^\dagger\|_{\mathcal{H}_u}] \le \mathbb{E}_0[\|\bar{u}_p(\boldsymbol{d}; W_I) - \bar{u}_p\|_{\mathcal{H}_u}] + \|\bar{u}_0 - u^\dagger\|_{\mathcal{H}_u} \\ + tr\left(\mathcal{C}_p H^* \Sigma_{\mathrm{inv}}^* \Sigma \Sigma_{\mathrm{inv}}^* H \mathcal{C}_p^*\right)^{1/2},\end{aligned} \tag{2.60}$$

where Σ is the background truth noise covariance matrix and Σ_{inv}^* is the diagonal matrix $diag\left(\alpha_1^*/\beta_1^*, \ldots, \alpha_{N_d}^*/\beta_{N_d}^*\right)$ defined in (2.21).

We have the following observations on the three terms appearing in (2.60). The first term on the right-hand side of (2.60) reflects the approximation error arising from learning by DNNs, which can be analyzed in detail, e.g., by introducing the concepts of sample error and approximation error (Cucker and Smale, 2001). The DNNs employed here need to learn a nonlinear operator between two separable Hilbert spaces, which, to the best of our knowledge, can hardly be analyzed under the current learning theory. However, there are some works on the learnability of the nonlinear operators (Chen and Chen, 1995) that should be useful for future investigations.

The second term accounts for the error induced by the synthetic function parameter used for training and the true background function. This term is difficult to analyze in its current form since $u^\dagger$ and $\bar{u}_0$ cannot be obtained in the inverse computational stage. However, we can assume that the residual of the synthetic function and the background truth can be bounded by some constant with high probability. The third term on the right-hand side of (2.60) represents the errors induced by noise and the prior covariance operator.

2.4 Parametric strategies

In the implementation of the algorithm, it is necessary to choose an appropriate prior probability measure μ_0 and approximate probability measure ν to ensure Assumption 7. This subsection focuses on a possible parametric strategy.

For the finite-dimensional parameter $\boldsymbol{\sigma}$, there are no singular issues with the prior and posterior probability measures. Therefore, we focus on the more problematic function parameter u. The major difficulty is that Gaussian measures defined on infinite-dimensional space are singular with each other when the covariance operators are scaled by different constants. It remains an open question as to how to resolve such a singularity problem in terms of theory and algorithms (Agapiou et al., 2014; Dunlop et al., 2017).

Let us introduce some assumptions proposed in (Dashti and Stuart, 2017) that inspire us to define appropriate covariance operators in the prior and posterior probability measures.

Assumption 10 *(Dashti and Stuart, 2017) The operator A, densely defined on the Hilbert space $\mathcal{H} = L^2(\Omega; \mathbb{R})$ where $\Omega \subset \mathbb{R}^d$ is a bounded domain with smooth boundary, is assumed to satisfy the following properties:*

1. *A is positive definite, self-adjoint and invertible;*
2. *The eigenfunctions $\{e_j\}_{j\in\mathbb{N}}$ of A form an orthonormal basis for $\mathcal{H}$;*
3. *The eigenvalues of A satisfy $\lambda_j \asymp j^{2/d}$;*
4. *There is $C > 0$ such that*

$$\sup_{j\in\mathbb{N}} \left(\|e_j\|_{L^\infty} + \frac{1}{j^{1/d}} Lip(e_j) \right) \leq C,$$

where $Lip(e_j)$ denotes the Lipschitz constant of $e_j, j = 1, \ldots, \infty$.

With Assumption 10, we now restrict the prior probability measure of u to be $\mu_0^u = \mathcal{N}(\bar{u}_0, \mathcal{C}_{\epsilon_0})$ with $\mathcal{C}_{\epsilon_0} = (\epsilon_0^{-1}\mathrm{Id} + \delta A^{\alpha/2})^{-2}$. Here, the parameters $\epsilon_0, \delta > 0$ are positive constants satisfying $\delta \ll \epsilon_0^{-1}$, and the parameter $\alpha > d/2$ controls the regularity properties of

the generated random samples. The mean function $\bar{u}_0$ is taken as the simulated background truth, which is available at the training stage but cannot be obtained at the inference stage.

For the posterior probability measure $\nu^u = \mathcal{N}(\bar{u}_p(\boldsymbol{d}; W_I), \mathcal{C}_p(\boldsymbol{d}; W_I))$, we specify the covariance operator as follows:

$$\mathcal{C}_p(\boldsymbol{d}; W_I) = (a(\boldsymbol{d}; W_I)\mathrm{Id} + \delta A^{\alpha/2})^{-2}, \tag{2.61}$$

where α and δ are taken the same as the prior measure, and $a(\boldsymbol{d}; W_I)$ is a positive function parameterized by the neural network with parameters W_I. In the following, we let $\mathcal{H}_u = \mathcal{H}$ which is defined in Assumption 10.

The following result illustrates that μ_0^u and ν^u defined above are equivalent under certain general conditions.

Theorem 11 *Assume that $\Omega \subset \mathbb{R}^d$ ($d \leq 3$) is a bounded domain with a smooth boundary, and the function $a(\boldsymbol{d}; W_I) \in L^\infty(\Omega)$ has positive lower and upper bounds, i.e., $0 < \underline{a} \leq a(\boldsymbol{d}; W_I)(x) \leq \bar{a}$ for all $x \in \Omega$ with $\underline{a}, \bar{a} \in \mathbb{R}^+$. Assume further that the mean functions $\bar{u}$ and $\bar{u}_p(\boldsymbol{d}; W_I)$ introduced in μ_0^u and ν^u belong to the Hilbert scale $\mathcal{C}_{\epsilon_0}^{1/2}\mathcal{H}_u$. Then, the approximate posterior measure ν^u is equivalent to the prior measure μ_0^u.*

The proof is provided in Subsection 5.3 of the Appendix.

3. Numerical Examples

In this section, we provide the specific settings of INet and SNet for inverse problems of PDEs. We then apply the VINet to two examples: a simple smoothing model, which is widely used for benchmark testing (Agapiou et al., 2014), and an inverse source problem of the Helmholtz equation.

3.1 Network structures

In Section 2, we introduce a general inverse framework called VINet, wherein any type of rational neural network can be chosen. Well-constructed neural networks typically perform better and require a smaller training dataset, which is crucial for practical applications.

The INet proposed in the general inference framework has three functionalities:

1. The network should be able to accurately transmit the information of the noises $\boldsymbol{\epsilon}$, which is assumed to be distributed according to some non-i.i.d. Gaussian distribution.

2. The network should be able to learn the prior information from the training data set, which can be used to provide high-quality estimations.

3. The forward evaluation of the network can be calculated more efficiently than the classical iterative algorithm.

Based on the three criteria above, we designed a specific INet structure in the previous section, which is illustrated in Figure 2. The network contains three components. The first component, named DNet, undertakes the denoising task, which is expected to learn the noise structure and provide estimated clean data. The second component, named CECInv,

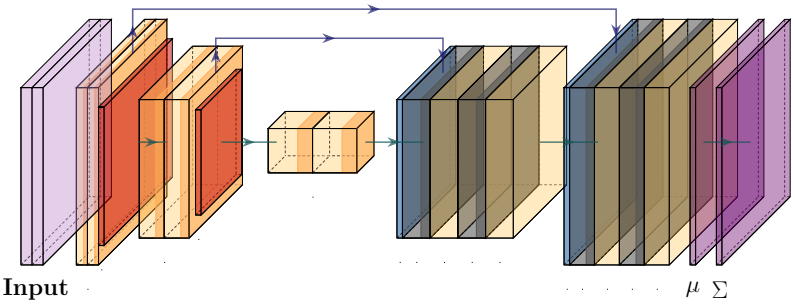

Figure 3: The tiny U-Net architecture employed for constructing DNet.

is a small-scale classical inversion, in which every inverse of sparse matrices can be explicitly calculated, and the results are some dense matrices. Clearly, only matrix multiplications are needed during the computations, which can be more efficiently implemented than solving linear equations. The third component, named ENet, can be viewed as an enhancing layer, which learns the residual of the inexact inversion obtained by the CECInv layer with the background truth. More details of the three components are listed below:

1. DNet is chosen to be a small U-Net architecture, as illustrated in Figure 3, containing three encoder blocks, two decoder blocks, and symmetric skip connections under each scale.

2. A small-scale classical inversion (which may be different for each problem) is employed as the CECInv layer, providing a low-resolution estimate (which may contain some artifacts).

3. For the ENet, we adopt two types of neural networks: U-Net with rescale layers (details are given in Remark 12) and Fourier neural network (FNO) (Li et al., 2020b). We will provide specific settings later, since different parameters are employed for the two numerical examples.

As for the SNet, it is adopted to infer the variational posterior parameters $\{\alpha_i(\boldsymbol{d}; W_S)\}_{i=1}^{\infty}$, $\{\beta_i(\boldsymbol{d}; W_S)\}_{i=1}^{\infty}$ from the noisy measurement data $\boldsymbol{d}$. Similar to the finite-dimensional denoising case (Yue et al., 2019), we use the DnCNN (Zhang et al., 2017) architecture with five layers, and the feature channels of each layer are set to be 64.

Combining INet and SNet, we illustrate the whole framework in Figure 4 with ENet has chosen to be the U-Net that contains rescale layers. It is easy to train our model based on the objective function presented in (2.58) by using the stochastic gradient descent algorithm or any variant optimization techniques. As for the finite-dimensional case, each term of (2.49) can be intuitively explained. The INet and SNet are updated from the terms I_1 and I_2, respectively, by controlling the discrepancy between the variational posteriors and the priors. Corresponding to the noise in the training dataset, the term I_3 couples INet and SNet together, generating gradients simultaneously during backpropagation.

For any new noisy data $\boldsymbol{d}$, the trained INet can be used to obtain the final estimated result directly and explicitly. Additionally, SNet can be used to easily infer the noise information, even with complicated non-i.i.d. configurations, by inputting the noisy data.

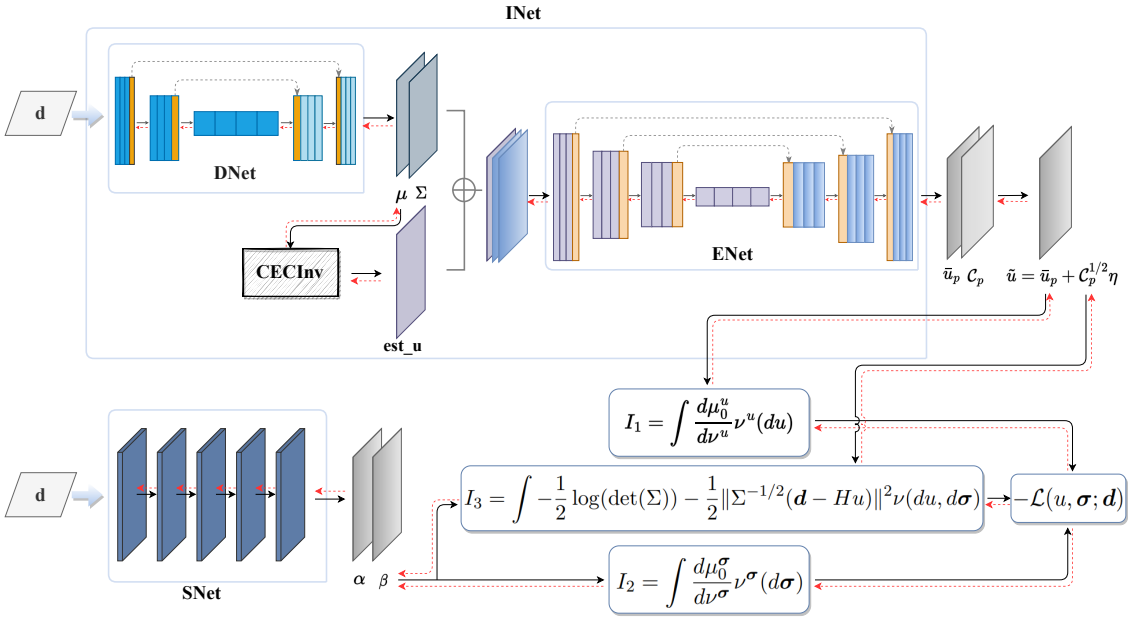

Figure 4: A particular example of the VINet framework (ENet is illustrated by a U-Net architecture).

Numerically, the related PDEs are solved using the finite element method, which is implemented with the open software FEniCS (Version 2019.1.0). Additional information on FEniCS can be found in (Logg et al., 2012). The related neural networks are implemented with the open software PyTorch (Version 1.12.1+cu115). The mesh of the finite element method is generated automatically by the FEniCS command *UnitSquareMesh(·)*. Since the mesh is regular, the finite difference discretization can also be adopted based on the same grid points. Therefore, we use the finite difference method to compute the integral formulas involved in Terms 1 and 3 of the loss functional (2.49). All programs were run on a computer with an Intel(R) Core(TM) i9-12900KF (CPU), a GeForce RTX 3080 (GPU) with 12 GB memory, and Ubuntu 20.04.1 LTS (OS).

Remark 12 *It should be mentioned that the ENet is a mapping between two infinite-dimensional spaces, which is typically specified as a neural network with discretization-invariant property, e.g., the neural network proposed in (Bhattacharya et al., 2021; Nelsen and Stuart, 2021; Li et al., 2020a,b). In the present work, we employed the FNO (Li et al., 2020b) as one possible choice of the ENet, which learns a fixed number of Fourier modes for each layer, making the whole network process discretization-invariant. Besides the FNO, we also used the U-Net as one option for the ENet. As is well-known, the U-Net structure does not have discretization-invariant property. To overcome this obstacle, we added rescale layers as the input and output layers. The core idea of the rescale layer is that the input function is viewed as being discretized on a mesh by the finite element method, so we can compute the function values on any different discretized meshes. With rescale layers, we projected the input functions on a fixed mesh (e.g., a mesh generated by UnitSquareMesh(·) command in FEniCS with mesh size* 150×150*) and then projected the output function on*

a desired mesh by a rescale layer. Numerical examples show that the U-Net with rescale layers processes the discretization-invariant property to some extent.

3.2 A simple smoothing model

3.2.1 Basic settings

Consider an inverse source problem of the elliptic equation

$$\begin{aligned} -\alpha\Delta w + w &= u \quad \text{in } \Omega, \\ \frac{\partial w}{\partial \boldsymbol{n}} &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{3.1}$$

where $\Omega = [0,1]^2 \subset \mathbb{R}^2$, $\alpha > 0$ is a positive constant, and $\boldsymbol{n}$ denotes the outward normal vector. The forward operator H is defined as follows:

$$Hu = (w(\boldsymbol{x}_1), w(\boldsymbol{x}_2), \cdots, w(\boldsymbol{x}_{N_d}))^T, \tag{3.2}$$

where $u \in \mathcal{H}_u := L^2(\Omega)$, w denotes the solution of (3.1), and $\boldsymbol{x}_i \in \Omega$ for $i = 1, \ldots, N_d$. In our implementations, the measurement points $\{\boldsymbol{x}_i\}_{i=1}^{N_d}$ are taken at the coordinates $\left\{\left(\frac{k}{20}, \frac{\ell}{20}\right)\right\}_{k,\ell=1}^{20}$. To avoid the inverse crime (Kaipio and Somersalo, 2005), we discretize the elliptic equation by the finite element method on a regular mesh (the grid points are uniformly distributed on the rectangular domain) with the number of grid points being equal to 400×400. All of the training and testing data pairs are generated based on this discretization. For revealing the impact of the discretization resolution, the related functions are discretized by finite element method on a series of regular mesh with the number of grid points $n = \{30 \times 30, 50 \times 50, 100 \times 100, 150 \times 150, 200 \times 200, 250 \times 250, 300 \times 300\}$ at the inverse stage.

For the operator A in Assumption 10 used in the prior probability measure, it is chosen to be the negative Laplace operator with Neumann boundary condition, which is similar to (Bui-Thanh et al., 2013). Hence, the operators in Subsection 2.4 can be specified explicitly as follows:

$$\mathcal{C}_{\epsilon_0} = (\epsilon_0^{-1}\mathrm{Id} + \delta(-\Delta)^{\alpha/2})^{-2}, \quad \mathcal{C}_p(\boldsymbol{d}; W_I) = (a(\boldsymbol{d}; W_I)\mathrm{Id} + \delta(-\Delta)^{\alpha/2})^{-2}.$$

To simplify the numerical implementations, we choose $\alpha = 2$, which makes the operators $\mathcal{C}_{\epsilon_0}$ and $\mathcal{C}_p(\boldsymbol{d}; W_I)$ to be discretized as illustrated in (Bui-Thanh et al., 2013). Since the fractional Laplace operator needs to be discretized differently (Bui-Thanh and Nguyen, 2016; Lischke et al., 2020), we leave the case $\alpha \neq 2$ for our future research.

Concerned with the CECInv layer, we employed a classical truncated singular value method (Engl et al., 1996). Specifically, the inverse method can be written in the form

$$R_{\lambda_m}\hat{\boldsymbol{d}} = q_{\lambda_m}(H^*H)H^*\hat{\boldsymbol{d}}, \tag{3.3}$$

where $\hat{\boldsymbol{d}}$ is the output of the DNet, and

$$q_{\lambda_m}(\lambda) = \begin{cases} \lambda^{-1}, & \lambda \geq \lambda_m, \\ 0, & \lambda < \lambda_m, \end{cases} \tag{3.4}$$

with λ_m denoting a predefined truncate level. In the above, the operator $q_{\lambda_m}(H^*H)$ appearing in (3.3) should be understood as the functional calculus for bounded self-adjoint operators, as described in Chapter 2 of (Engl et al., 1996). The PDEs were discretized on a regular mesh with the number of grid points being equal to 30×30, which is relatively small compared with the forward discretization. Based on such a small mesh, we can calculate the related inversion of the sparse matrices generated by the finite element method explicitly, i.e., there is no need to solve a lot of sparse linear equations.

In our experiments, we generate 1000 training function parameters u based on a Gaussian probability measure with mean function equal to constant 2 and covariance operator given by

$$\mathcal{C}^{-1} = \alpha^2(\mathrm{Id} - \nabla \cdot (\Theta \nabla \cdot))^2, \tag{3.5}$$

where $\Theta := \mathrm{diag}(10, 1)$ and $\alpha = 0.01$. For testing the proposed methods, we generate two types of testing datasets with 100 function parameters u for each one. Specifically, type 1 test datasets are generated by a Gaussian probability measure with the same settings as the training datasets. Type 2 test datasets are generated by a Gaussian probability measure with the mean function given by $2 + 0.2\sin(2\pi x)\sin(2\pi y)$ and the same covariance operator as the training datasets. Since, as mentioned in Remark 1, we may not know the ideal background truth practically, type 2 test datasets are designed to test the generalization abilities of the proposed VINet framework.

Previous studies (Yue et al., 2019, 2020) have implied that multiplicative noises can be removed efficiently by a non-i.i.d. noise model. Therefore, multiplicative noises are added to the clean data $\boldsymbol{d}_c$ to generate noisy data $\boldsymbol{d}$ in order to test the performance of the proposed VINet on non-i.i.d. noise cases. In the training stage, we add noises according to $\boldsymbol{d} = \boldsymbol{d}_c + a(\boldsymbol{\eta} \odot \boldsymbol{d}_c)$, where $\odot$ is an element-wise product, $\boldsymbol{\eta} \sim \mathcal{N}(0, \mathrm{Id})$, and $a \sim U[-0.1, 0.1]$. In the testing stage, we take $a = 0.1$, i.e., $\boldsymbol{d} = \boldsymbol{d}_c + 0.1(\boldsymbol{\eta} \odot \boldsymbol{d}_c)$, where $\boldsymbol{\eta} \sim \mathcal{N}(0, \mathrm{Id})$.

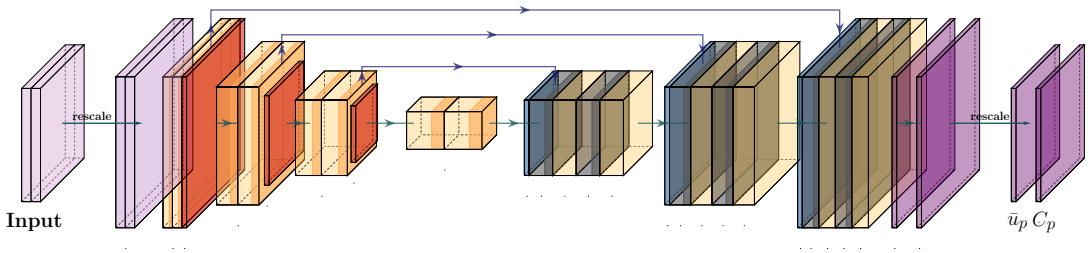

Figure 5: The U-Net architecture is employed for constructing ENet. The input discretized functions are projected onto a fixed mesh by a rescaling layer. Then, through a conventional U-Net, the output functions are obtained on the fixed mesh. Finally, the output functions are projected onto the required mesh by a rescaling layer.

On the one hand, the ENet has been chosen to be a U-Net, as illustrated in Figure 5, which contains four encoder blocks, three decoder blocks, symmetric skip connections under each scale, and two rescale blocks as the input and output layers. For simplicity, we refer to this as U-ENet. On the other hand, we also take ENet to be an FNO, as illustrated

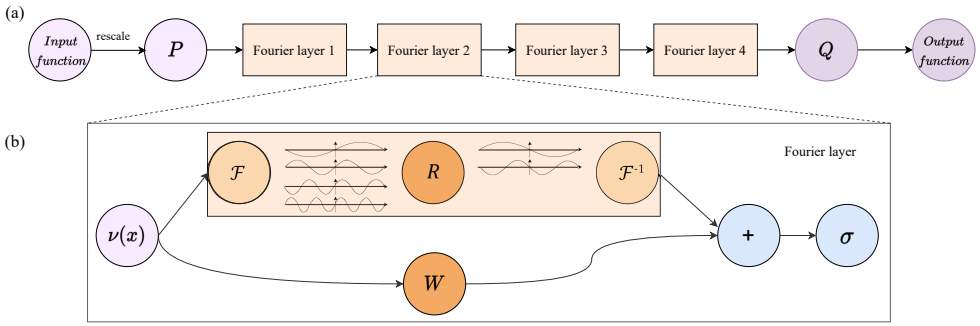

Figure 6: The FNO architecture is employed for constructing ENet (this figure is nearly the same as Figure 2 in (Li et al., 2020b)). (a) **The full architecture of FNO**: Input functions are lifted to a higher-dimensional channel space by a neural network P. Four Fourier layers are employed to enhance the results of the CECInv layer. Finally, the functions are projected back to the target dimensions (2 in our setting) by a neural network Q. (b) **Fourier layers**: Upper part: the Fourier transform $\mathcal{F}$ is applied; a linear transform R is implemented on the lower Fourier modes; then the inverse Fourier transform $\mathcal{F}^{-1}$ is applied. Lower part: a local linear transform W is applied.

in Figure 6, which contains one lift layer, four Fourier integral operator layers, and one project layer. We set the maximal number of Fourier modes to be twelve for each Fourier integral operator, and the original inputs are lifted to a higher dimensional representation with a dimension equal to thirty-two. These settings are the same as the original paper of FNO (Li et al., 2020b) for two-dimensional problems. For simplicity, we refer to this as FNO-ENet. Lastly, we should mention that the input layer of ENet contains two channels: the output of the CECInv layer and the variance function obtained from DNet. The output layer of ENet also contains two channels: the mean function and the function parameter in the covariance operator.

For training the VINet, we first train the DNet and SNet, then train the ENet and SNet with the trained DNet (parameters are fixed), which makes the two components more interpretable and reduces the training time. For this example of DNet, we use the Adam optimizer to train for 70 epochs with an initial learning rate of 0.0001, halved every 10 epochs. For the U-ENet and FNO-ENet, we also use the Adam optimizer to train for 15 epochs with an initial learning rate of 0.0001, halved every 10 epochs. The whole training times of this example were 1789.89 and 1536.15 seconds (approximately 30 and 27 minutes) for VINet with U-ENet and FNO-ENet, respectively.

3.2.2 Numerical results

We compare the results of VINet, the mean-field based variational inference method (MFVI) exhibited in Algorithm 1, and the truncated singular value decomposition (TSVD) method

shown in formula (3.3). For the MFVI method, the estimated function u highly depends on the particular choices of the prior measure. Here, we assume that the distribution generating the truth is known, which means the statistical properties of the truth are known exactly for the MFVI method. For comparison, we define the relative error as follows:

$$\text{relative error} = \frac{\|\tilde{u} - u_{\text{truth}}\|_{L^2}}{\|u_{\text{truth}}\|_{L^2}}, \tag{3.6}$$

where $\tilde{u}$ is the estimated function.

For the TSVD method, its performance depends on the predefined truncation level. To determine the optimal truncation level λ_m, we set the truncation level equal to $2, 1.5, 1, 0.5$, and 0.1. Then we applied the TSVD method to the type 1 test datasets, which provided the optimal truncation level based on the average relative errors. Detailed results are given in Table 1. From the table, we can see that the optimal truncation level is 1.0, no matter which discrete levels are employed. Therefore, we always specify the truncation level to be 1.0 in the following. In our opinion, the reason the discrete levels are not highly influential is that the measured noisy data $\boldsymbol{d}$ can only provide limited information, and the background true parameters are smooth in some sense. For the next numerical example, the discrete level will be more influential due to the poor regularity of the parameter u. One point should be mentioned: we employed the discrete strategy illustrated in (Bui-Thanh et al., 2013) to calculate the eigen-system, i.e., the operator H will never be constructed (only Hu can be calculated), so a randomized algorithm named double pass (Halko et al., 2011; Villa et al., 2021) has been employed that does not require H and H^* to be constructed explicitly.

Table 1: Average relative errors of the estimates obtained by the TSVD.

Truncate level	2.0	1.5	1.0	0.5	0.1
Relative error ($n = 30^2$)	0.1527	0.1523	**0.1484**	0.2020	1.1417
Relative error ($n = 50^2$)	0.1484	0.1484	**0.1439**	0.1945	1.1743
Relative error ($n = 100^2$)	0.1489	0.1488	**0.1448**	0.1939	1.3359
Relative error ($n = 150^2$)	0.1493	0.1493	**0.1455**	0.1942	1.3299
Relative error ($n = 200^2$)	0.1494	0.1494	**0.1456**	0.1942	1.4759
Relative error ($n = 250^2$)	0.1495	0.1495	**0.1482**	0.1942	1.4752
Relative error ($n = 300^2$)	0.1495	0.1495	**0.1482**	0.1944	1.4752

From the iteration procedure of Algorithm 1, it is easily seen that the MFVI method is computationally expensive (see Remark 5). Let us denote N_{ite} as the iterative number of linear equation solvers employed to solve $(H^*\Sigma_{\text{inv}}^{k*}H + \mathcal{C}_0^{-1})\tilde{u} = u$ mentioned in Remark 5. Let k_{max} be the maximum iterative number of Algorithm 1. In order to reduce the computational time, we employ the methods given in (Jia et al., 2021b), which calculate

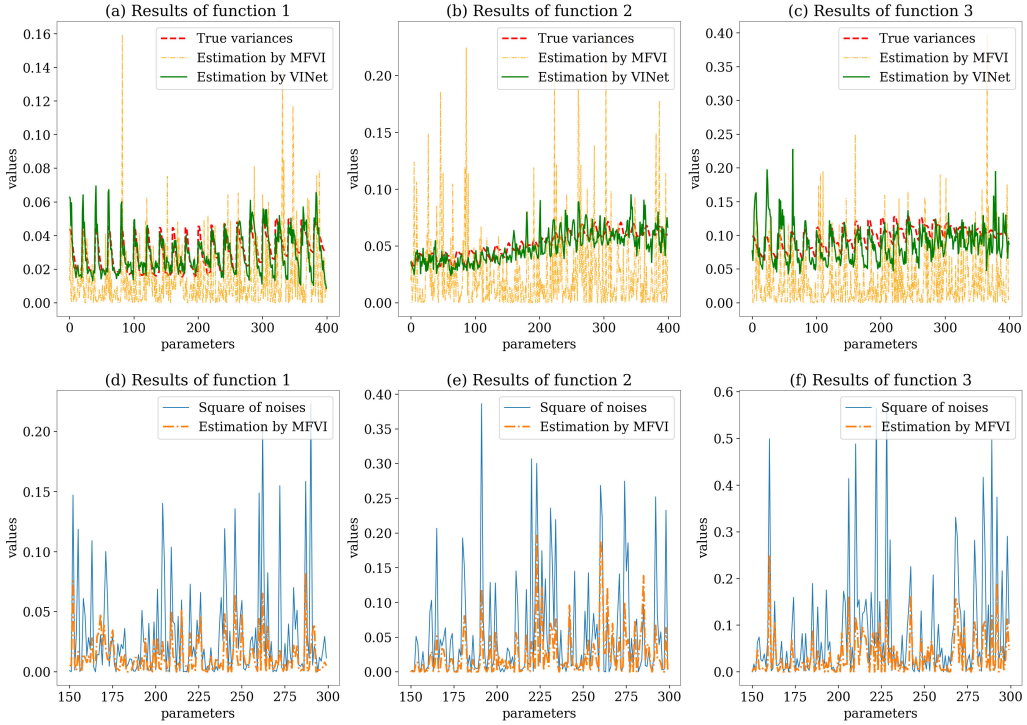

Figure 7: Comparison of the true noise variances and estimated noise variances by the MFVI and VINet (output of SNet). (a) (b) (c) Noise variances of all parameters for three different testing examples. (d) (e) (f) Noise variances estimated by MFVI and the square real noises of some parameters.

the mean function u_k in Algorithm 1 with a fine mesh and then project the mean function to a rough mesh (mesh size $n = 50 \times 50$) to reduce the computational complexity for updating β. Although the computational complexity has been reduced, we still need to solve approximately $(2N_{ite} + 2)k_{max}$ number of PDEs, e.g., if $N_{ite} = 100$ and $k_{max} = 20$, there are 4040 number of PDEs that need to be solved. Actually, the above illustration is just a rough estimate, and we only control the errors for calculating u_k but not the iteration number N_{ite} since the accuracy of u_k has a large impact on the performance of MFVI. Hence the number N_{ite} may be larger than 100. Considering the large computational complexity, we only apply the MFVI method to the first 50 types 1 and 2 testing examples. Since MFVI is not a learning based method, we can expect that the average performance on the first 50 testing examples is similar to the average performance on the whole testing examples.

Before presenting the numerical results, we should mention that the VINets with U-ENet and FNO-ENet were all trained when the related function parameters u were discretized on a regular mesh with a mesh size of 100×100. All other inversion results on different meshes were produced by the same trained VINet. In the following, let us visually compare the results obtained by TSVD, MFVI, and VINet when the mesh size is equal to 150×150. Then, we will provide a detailed comparison of these methods with different mesh sizes.

Let us first compare the estimates of the noise variances given by the MFVI and VINet. Because type 2 test datasets deviate from the training datasets (more challenging than type

1 ones), we only show the results of type 2 test datasets to illustrate the effectiveness of the proposed VINet. For interested readers, we exhibit the results of type 1 test datasets in the Appendix. In Figure 7, we depict the estimated noise variances by the MFVI and VINet and the background true noise variances for three randomly selected testing examples. In the first line of Figure 7, the estimated variances of all 400 measurement points are given, with the dashed red line representing the true noise variances, the dash-dotted light orange line representing the estimation given by MFVI, and the solid green line representing the estimation given by the SNet in VINet. Obviously, the SNet provides evidently more accurate estimates of the noise variances for all of the three type 2 test datasets. In the second line of Figure 7, we use the solid blue line and dash-dotted orange line to represent the square real noises, i.e., $(\boldsymbol{d} - \boldsymbol{d}_c)^2$, and the estimations given by MFVI, respectively. Obviously, the MFVI method only catches some features of the square real noises, but not the true noise variances $0.1^2\boldsymbol{d}_c^2$. From this figure, we see that the real noise variances $0.1^2\boldsymbol{d}_c^2$ can be visually much different from one particular realization $(\boldsymbol{d} - \boldsymbol{d}_c)^2$. Hence, it is a difficult task to estimate $0.1^2\boldsymbol{d}_c^2$ from the noisy data. Based on one piece of noisy data, the MFVI method can hardly give a reasonable estimate. However, by learning from the training datasets, the SNet captures the main features of the true noise variances.

Secondly, let us show the background truth and estimates obtained by the TSVD, MFVI, VINet with U-ENet, and VINet with FNO-UNet for one randomly selected data from the type 2 testing datasets. As before, the results of type 1 testing datasets are provided in the Appendix. In Figure 8, we exhibit the true function in sub-figure (a) and the estimated functions by the TSVD, MFVI, VINet with U-ENet, and VINet with FNO-ENet in sub-figures (b)-(e), respectively. Obviously, the TSVD and MFVI methods can only provide a rough estimate of the true parameter. Comparatively, no matter whether using U-ENet or FNO-ENet, the proposed VINet captures more detailed structures of the true parameter. In addition, we show the estimated variance functions of VINet with log-scale in sub-figures (f) and (g). For this example, compared with the VINet with U-ENet, the VINet with FNO-ENet gives a less accurate estimate of the dark part (lower right corner) of the true parameter. The region of the darkest part in sub-figure (e) is larger than the darkest region in sub-figures (a) and (d). The variance function show in sub-figure (g) just characterize this inaccuracy, i.e., for the region of darkest part, the values of the variance function in sub-figure (g) is higher than other region.

At last, let us provide a detailed comparison of these methods with various discrete meshes. In Table 2, we list the average relative errors of the estimates obtained by the TSVD, MFVI, VINet with U-ENet, and VINet with FNO-ENet for type 2 test datasets when the mesh sizes are equal to 30×30, 50×50, 100×100, 150×150, 200×200, 250×250, and 300×300. From Table 2, we see that all four methods could provide stable estimates with different discretized dimensions. It should be mentioned that the VINets with both U-ENet and FNO-ENet were trained when the discrete mesh size was $n = 100 \times 100$. The relative errors of the proposed VINets were much smaller than those of the TSVD and MFVI methods. The TSVD method could only provide an estimate of the function parameter and the MFVI method gave the estimates with the highest relative errors. From Figure 8, we actually found that the two VINets provided more details compared with the TSVD and MFVI methods. In Table 3, we also provided the recorded computing times on CPU since the TSVD and MFVI methods were tested on CPU. If we ran VINet

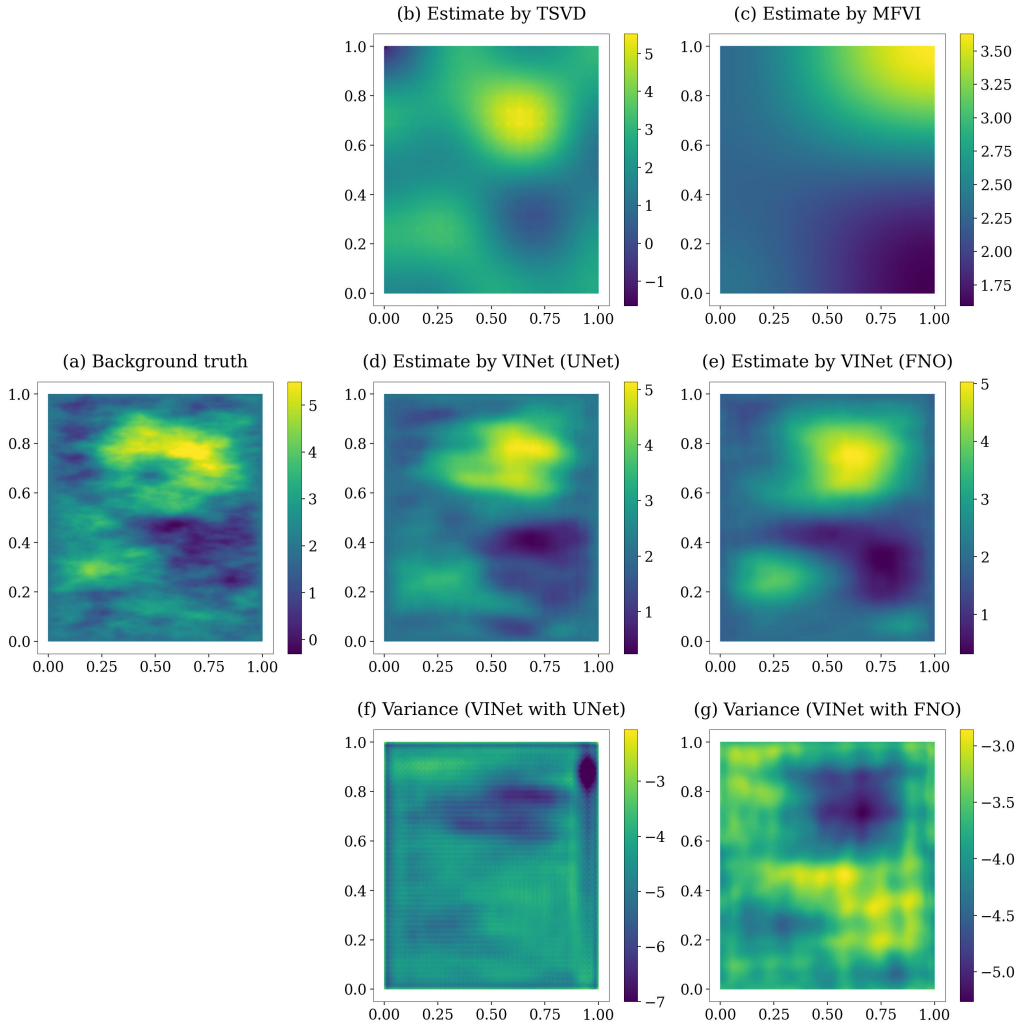

Figure 8: (a) Background true function randomly selected from type 2 testing datasets; (b) Estimate obtained by the TSVD method; (c) Estimate obtained by the MFVI method; (d) Estimate obtained by the VINet with U-ENet; (e) Estimate obtained by the VINet with FNO-ENet; (f) Variance function obtained by the VINet with U-ENet shown in log-scale; (g) Variance function obtained by the VINet with FNO-ENet shown in log-scale.

on GPU, the running time would be nearly ten times faster than that shown in Table 3. Obviously, in the inference stage, the two VINets were hundreds or thousands of times faster than the TSVD and MFVI methods. For the VINets with U-ENet and FNO-ENet, we took 1789.89 and 1536.15 seconds (approximately 30 and 27 minutes) to train the neural networks, respectively. The training time plus the inference time was even smaller than that of the MFVI with mesh sizes $n = \{250 \times 250, 300 \times 300\}$.

Table 2: Average relative errors of the estimates obtained by the TSVD, MFVI, VINet (U-ENet), and VINet (FNO-ENet) with mesh sizes $n = \{30 \times 30, 50 \times 50, 100 \times 100, 150 \times 150, 200 \times 200, 250 \times 250, 300 \times 300\}$ for the type 2 test datasets.

	TSVD	MFVI	VINet (U-ENet)	VINet (FNO-ENet)
Relative error ($n = 30^2$)	0.1484	0.3287	0.0881	0.0802
Relative error ($n = 50^2$)	0.1439	0.3191	0.0940	0.0849
Relative error ($n = 100^2$)	0.1448	0.3218	0.0980	0.0881
Relative error ($n = 150^2$)	0.1455	0.3279	0.0989	0.0888
Relative error ($n = 200^2$)	0.1456	0.3614	0.0996	0.0894
Relative error ($n = 250^2$)	0.1482	0.3535	0.0997	0.0894
Relative error ($n = 300^2$)	0.1482	0.3520	0.0999	0.0896

Table 3: Average computing times of the TSVD, MFVI, VINet (U-ENet), and VINet (FNO-ENet) with mesh sizes $n = \{30 \times 30, 50 \times 50, 100 \times 100, 150 \times 150, 200 \times 200, 250 \times 250, 300 \times 300\}$ for the type 2 test datasets. The computing time is measured in seconds.

	TSVD	MFVI	VINet (U-ENet)	VINet (FNO-ENet)
Average time ($n = 30^2$)	0.48	25.69	0.0296	0.0122
Average time ($n = 50^2$)	1.10	52.72	0.0296	0.0125
Average time ($n = 100^2$)	5.65	330.72	0.0298	0.0203
Average time ($n = 150^2$)	15.21	655.51	0.0299	0.0336
Average time ($n = 200^2$)	31.71	1092.62	0.0305	0.0436
Average time ($n = 250^2$)	55.39	2071.76	0.0305	0.0594
Average time ($n = 300^2$)	86.82	2921.92	0.0311	0.0729

3.3 An inverse source problem

3.3.1 Basic settings

In this subsection, we consider an inverse source problem for the Helmholtz equation with multi-frequency data which was studied in (Bao et al., 2015)[2]. Specifically speaking, we

2. Deep learning based methods are employed for statistical inverse problems under the finite-dimensional setting for CT image reconstruction, PET image reconstruction, and MRI reconstruction (Arridge et al., 2019). For these problems, the finite-dimensional formulation is widely employed. However, to the best of our knowledge, there seems little such types of work focused on multi-frequency inverse source problems governed by the Helmholtz equation (one of the typical and important mathematical models

consider the following Helmholtz equation:

$$\begin{aligned} &\Delta w + \kappa^2 w = u \quad \text{in } \mathbb{R}^2, \\ &\partial_r w - i\kappa w = o(r^{-1/2}) \quad \text{as } r = |x| \to \infty, \end{aligned} \tag{3.7}$$

where κ is the wavenumber, w is the acoustic field, and u is the acoustic source supported in an open bounded domain $\Omega = [0,1]^2$. To simulate the problem defined on the infinity domain $\mathbb{R}^2$, we adopt the perfectly matched layer (PML) technique and reformulate the scattering problem in a bounded domain. Let $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$. Denote D as a rectangle containing Ω, and let d_1 and d_2 be the thickness of the PML layers in the x and y axes, respectively. Let ∂D be the boundary of D. Define $s_1(x) = 1 + i\sigma_1(x)$ and $s_2(y) = 1 + i\sigma_2(y)$, where σ_1, σ_2 are positive continuous functions satisfying $\sigma_1(x) = 0, \sigma_2(y) = 0$ in Ω. For more details of the PML, please refer to (Bao et al., 2010).

Following the general PML technique, we can deduce the truncated PML problem:

$$\begin{aligned} &\nabla \cdot (s\nabla w) + \kappa^2 s_1 s_2 w = u \quad \text{in } D, \\ &w = 0 \quad \text{on } \partial D, \end{aligned} \tag{3.8}$$

where the diagonal matrix $s = \mathrm{diag}(s_2(y)/s_1(x), s_1(x)/s_2(y))$. The forward operator related to κ is defined by the Helmholtz equation $H_\kappa(u) = (w(\boldsymbol{x}_1), \ldots, w(\boldsymbol{x}_{N_d}))^T$ with $\{\boldsymbol{x}_i\}_{i=1}^{N_d} \in \partial\Omega$ and $u \in \mathcal{H}_u := L^2(\Omega)$. Since we consider the multi-frequency case, i.e., a series of wavenumbers $0 < \kappa_1 < \kappa_2 < \cdots < \kappa_{N_f} < \infty$ are considered, the forward operator has the following form:

$$Hu = (H_{\kappa_1}(u), \ldots, H_{\kappa_{N_f}}(u)) \in \mathbb{R}^{N_d \times N_{N_f}}. \tag{3.9}$$

Similar to the simple smoothing model, we generate the training and testing data based on a fine mesh with the number of grid points equal to 500×500. In the inverse computational stage, the related functions are discretized by the finite element method on a series of regular meshes with the number of grid points $n = \{30 \times 30, 100 \times 100, 150 \times 150, 200 \times 200, 250 \times 250, 300 \times 300\}$. Concerning the operators $\mathcal{C}_{\epsilon_0}$ and $\mathcal{C}_p$, we adopt the same settings as those in Subsection 3.2 for the simple smoothing model.

For the CECInv layer, we employ the classical recursive linearization method (RLM) reviewed in (Bao et al., 2015) with $\kappa = 10, 20, 30, 40, 48$. The PDEs are discretized on a regular mesh with the number of grid points equal to 30×30, which makes it possible to directly compute the inverse of all the sparse matrices generated by the finite element method. Obviously, for a source function, the data generated by PDEs on a fine mesh with grid points equal to 500×500 are different from the data obtained by PDEs on a coarse mesh, especially when the wavenumber is large. For this example, we take the data generated by PDEs on the fine mesh with random Gaussian noise as the noisy input data. The accurate data are recognized as the data generated by solving PDEs with the coarse mesh, i.e., a mesh with size $n = 30 \times 30$. Hence, the DNet and SNet actually learn the modeling error and random noise together in this example. Please refer to (Calvetti et al., 2018; Jia et al., 2019) for more relevant references on model error learning algorithms.

usually defined on infinite-dimensional space (Bao et al., 2015)). So we compare the performance of the proposed method with the classical recursive linearization method (RLM), which should be sufficient to reveal the mechanism and superiority of the proposed method.

Before presenting the numerical results, we clarify how the training and testing datasets are constructed. Similar to the strategies employed in (Jia et al., 2019), we introduce the following formula:

$$\tilde{u}(x,y) := \sum_{k=1}^{3} a_k^1 \exp\Big(- a_k^2(x-a_k^3)^2 - a_k^4(y-a_k^5)^2 \Big), \tag{3.10}$$

where

$$a_k^1 \sim U[-1,1], \quad a_k^2, a_k^4 \sim U[50,60], \quad a_k^3, a_k^5 \sim U[0.3,0.7],$$

with $k = 1,2,3$ and $U[a,b]$ being the uniform distribution between a and b. To demonstrate the effectiveness of the proposed method, we consider the following nonsmooth source function:

$$u(x,y) = \begin{cases} \tilde{u}(x,y) & \text{for } |\boldsymbol{x}-\boldsymbol{x}_0| < r_1, \\ -0.5 & \text{for } r_2 \le |\boldsymbol{x}-\boldsymbol{x}_0| \le r_1, \\ 0.0 & \text{for } |\boldsymbol{x}-\boldsymbol{x}_0| > r_2, \end{cases} \tag{3.11}$$

where $\boldsymbol{x}_0 = (0.5, 0.5)^T$, $r_1 \sim U[0.32, 0.4]$ and $r_2 \sim U[0.42, 0.48]$. Based on Eq. (3.11), we generate 1000 source functions for training and 100 source functions for testing. For testing the generalization properties, we also test the performance of the trained model on the following source function:

$$u^\dagger(x,y) = u(x,y) + \tilde{u}(x,y), \tag{3.12}$$

where u and $\tilde{u}$ are two functions generated according to Eq. (3.11) and Eq. (3.10), respectively. The functions defined by (3.12) and (3.11) exhibit similar properties but the function defined in (3.12) can never be generated from Eq. (3.11), which mimics the following situation: the constructed learning examples usually cannot contain completely correct information, i.e., prior knowledge may exhibit bias. In the following, we call the 100 test source functions generated from Eq. (3.11) and Eq. (3.12) as type 1 and type 2 test datasets, respectively. One thing should be emphasized is that the training datasets can be constructed by employing more sophisticated methods and are not restricted to the presented simple parametric form.

Similar to the simple smooth example, the ENet has been chosen to be either U-Net or FNO. If we choose ENet to be a U-Net, we will employ the same settings as in Subsection 3.2 and we will still call the ENet U-ENet in the following. When choosing ENet to be a FNO, we also take the same network structures but set the maximal number of Fourier modes to be 32 for each Fourier integral operator, and the original inputs are lifted to a higher dimensional representation with a dimension equal to 64. As before, we call ENet with FNO structure FNO-ENet in this section. Different from the simple smooth example, the current CECInv layer uses the measured data of 5 different wavenumbers, i.e., $\kappa = 10, 20, 30, 40, 48$. Due to the small number of wavenumbers, it is better to employ all of the 5 estimated functions obtained by different wavenumbers. In addition, the variance function of the data obtained from DNet needs to be propagated to the ENet. Hence, the input layer of ENet

for the current case has 6 channels (the 6 functions are projected on the same mesh) and the output layer contains 2 channels, i.e., the mean function and function parameter in the covariance operator.

The training procedure is similar to that in Subsection 3.2. We will first train DNet and SNet, and then train ENet and SNet with the trained DNet. For the first stage of training, we used the Adam optimizer to train for 80 epochs with an initial learning rate of 0.0001, which is halved every 10 epochs. For training U-ENet and FNO-ENet, we also used the Adam optimizer to train for 25 epochs with an initial learning rate of 0.0001, which is halved every 10 epochs. The total training times for this example were 33950 and 39687 seconds (approximately 9.4 and 11 hours) for VINet with U-ENet and FNO-ENet, respectively. The training time was much longer than that of the simple smooth model. The reasons for this are twofold: a large discrete dimension was employed, which made the computation of forward problems time consuming (during the training stage, we need to solve forward PDEs); for the multi-frequency problem, the measured data were more complex, and the function parameters had sharp edges which required more training loops.

3.3.2 Numerical results

We compare the results of the classical recursive linearization method (RLM) (Bao et al., 2015), VINet with U-ENet, and VINet with FNO-ENet. The RLM uses multiple frequency data, i.e., data obtained at a series of wavenumbers. Low frequency data make the RLM stable with respect to the initialization (accurate initial function is usually hard to determine) and the high frequency data ensure the accuracy of the RLM. For our experiments, the data for each wavenumber was added 5% random Gaussian noise. For the RLM, we employed two types of data, i.e., data $\boldsymbol{d}_{48}$ obtained when the wavenumbers are $\kappa = 1, \cdots, 48$, and data $\boldsymbol{d}_{80}$ obtained when the wavenumbers are $\kappa = 1, \cdots, 80$. For the VINets with U-ENet or FNO-ENet, we used 1000 training data pairs that contain the 1000 training source functions generated by formula (3.11) and the data $\boldsymbol{d}_{48}$ generated by solving PDEs on a fine mesh, i.e., mesh with size $n = 500 \times 500$. For training DNet, we also needed the data $\tilde{\boldsymbol{d}}_{48}$ generated by solving Helmholtz equations on a mesh with mesh size $n = 30 \times 30$, which is a cheap computational procedure due to the small mesh size (all sparse linear solvers transform to matrix multiplications). The DNet in this example actually learns to remove two types of noises: 5% random noises and the discrepancies of different discrete models at measurement points. Lastly, we should emphasize that the VINets with U-ENet and FNO-ENet were all trained when the mesh size was $n = 150 \times 150$. All of the results for other meshes were generated by VINets through the rescale layers of U-ENet or the FNO-ENet directly.

As in Subsection 3.2.2, we only provide the results for type 2 testing datasets (which have a different generation mechanism than the training datasets) and leave the results of type 1 testing datasets (which have the same generation mechanism as the training datasets) in the Appendix. Firstly, let us provide some visual comparisons to acquire some intuitive impressions about the estimates of different methods. In Figure 9, we show the results given by DNet with input data $\boldsymbol{d}_{48}$ that is randomly selected from the type 2 test datasets. In all of the sub-figures of Figure 9, we use the dashed blue line, dashed orange line, and solid green line to represent the clean data (solution of the rough forward

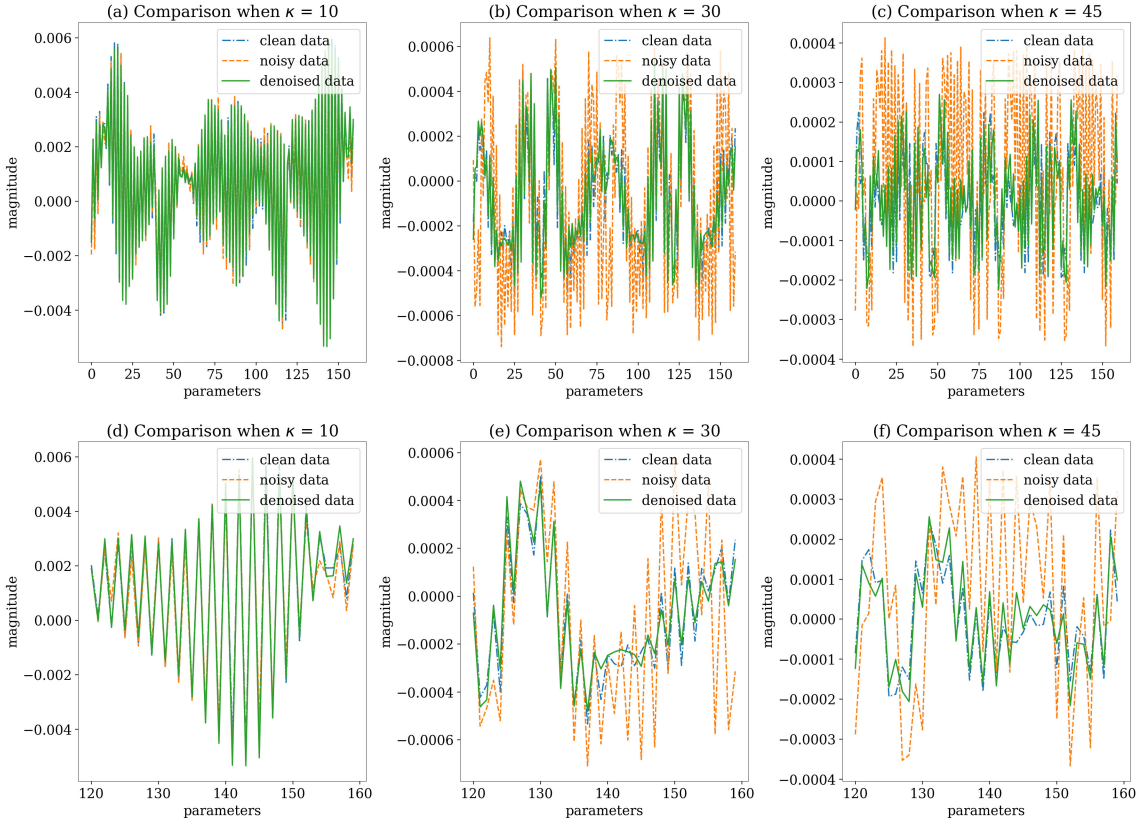

Figure 9: Noisy data (data generated by an accurate forward solver with random Gaussian noise), clean data (data generated by a rough forward solver), and denoised data (the output of the DNet) for a randomly selected source function u from type 2 test datasets are shown in (a) data when the wavenumber $\kappa = 10$; (b) data when the wavenumber $\kappa = 30$; (c) data when the wavenumber $\kappa = 45$; (d), (e), and (f) are partly enlarged versions of (a), (b), and (c), respectively.

solver with mesh size $n = 30 \times 30$), noisy data (solution of the accurate forward solver with mesh size $n = 500 \times 500$), and the output of the DNet, respectively. Sub-figures (a) (b) and (c) provide comparisons when the wavenumbers are chosen to be 10, 30, and 45, respectively. To provide more details, we exhibit sub-figures (d) (e) and (f) that give the partly enlarged versions of sub-figures (a) (b) and (c), respectively. When the wavenumber is small, i.e., $\kappa = 10$, we see that the clean data, noisy data, and output of DNet exhibit no significant differences. The reason is that we only need a rough forward solver when the wavenumbers are small. As the wavenumbers increase, we need a larger mesh to obtain an accurate forward solver. From sub-figures (b), (c), (e), and (f), we can clearly find that the noisy data deviate significantly from the clean data and the outputs of DNet are visually similar to the clean data, which reflects that DNet has learned the sophisticated model error mechanism between the accurate and rough forward solvers.

In Figure 10, we show the results of one randomly selected test example from the type 2 test datasets. In sub-figure (a), we exhibit the background true source function. In sub-

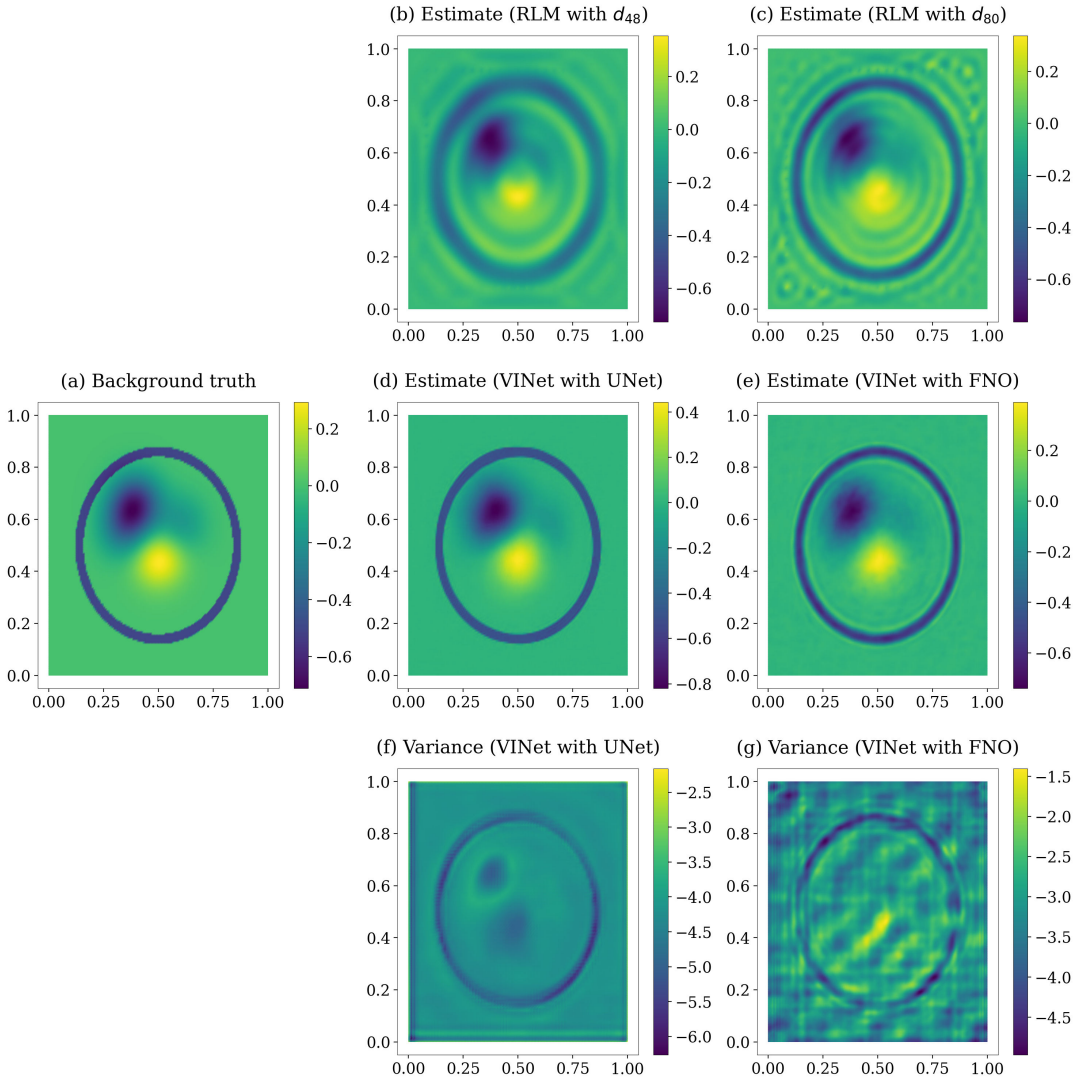

Figure 10: (a) One of the background true function in type 2 test datasets; (b) Estimate obtained by RLM with $\boldsymbol{d}_{48}$; (c) Estimate obtained by RLM with $\boldsymbol{d}_{80}$; (d) Estimate obtained by VINet with U-ENet; (e) Estimate obtained by VINet with FNO-ENet; (f) Estimated variance function by VINet with U-ENet shown in log-scale; (g) Estimated variance function by VINet with FNO-ENet shown in log-scale.

figures (b) and (c), we provide the estimated source functions of RLM with data $\boldsymbol{d}_{48}$ and $\boldsymbol{d}_{80}$, respectively. It is obvious that the classical RLM can hardly handle the discontinuity, so the sharp circle cannot be accurately estimated even when the highest wavenumber is set to 80. In sub-figures (d) and (e), we see that the VINets with U-ENet and FNO-ENet both yield evidently more accurate estimates compared to the classical RLM. In sub-figures (f) and (g), we show the variance functions obtained from the VINets with U-ENet and FNO-ENet in log-scale. The high values of variance functions appear at the changing parts

Table 4: Average relative errors of the estimates obtained by the RLM (using data $\boldsymbol{d}_{48}$ and $\boldsymbol{d}_{80}$), VINet with U-ENet, and VINet with FNO-ENet with mesh sizes $n = \{30 \times 30, 100 \times 100, 150 \times 150, 200 \times 200, 250 \times 250, 300 \times 300\}$ for the type 2 test datasets.

	RLM($\boldsymbol{d}_{48}$)	RLM($\boldsymbol{d}_{80}$)	VINet(U-ENet)	VINet(FNO-ENet)
Relative error ($n = 30^2$)	0.3670	0.3515	-	-
Relative error ($n = 100^2$)	0.2727	0.2763	0.0693	0.0969
Relative error ($n = 150^2$)	0.2623	0.2414	0.0630	0.0401
Relative error ($n = 200^2$)	0.2596	0.2246	0.0625	0.0410
Relative error ($n = 250^2$)	0.2581	0.2191	0.0630	0.0425
Relative error ($n = 300^2$)	0.2575	0.2149	0.0645	0.0450

of the source function, especially at the discontinuous boundary, which is reasonable since discontinuous boundaries are intuitively harder to estimate.

Next, let us give a detailed comparison of RLM and VINet with various discrete meshes. In Table 4, we show the averaged relative errors on different meshes obtained by RLM with $\boldsymbol{d}_{48}$, RLM with $\boldsymbol{d}_{80}$, VINet with U-ENet, and VINet with FNO-ENet, respectively. From the table, it can be observed that the discrete level indeed influences the performance of the classical RLM; the relative errors of the results obtained by RLM are larger than 1% when the discrete mesh size is smaller than $n = 200 \times 200$. Using higher frequency data $\boldsymbol{d}_{80}$, the inversion results indeed have lower relative errors. However, due to the limited measured data (we only assume the measurement points are equally spaced along the boundary with a number equal to 80), we see that the inverse accuracy increases slowly when more wavenumbers are employed. When the VINets with U-ENet and FNO-ENet are employed, the relative errors are around 6% and 4%, respectively. Both types of VINets provide stable inversion results for different meshes, except the VINet with FNO-ENet when the mesh size is equal to 100×100. This can be rationally explained by the fact that the function parameter u has a discontinuous sharp boundary which leads to larger numerical errors when the discrete dimension is small (compared with the frequency truncate level of FNO). When the function parameter u has better regularity, there seems to be no such phenomenon, as seen in Table 2.

At last, in Table 5, we give the computing times based on averages of the type 2 test datasets. Obviously, the inference times of both types of VINet are significantly shorter than those of the classical RLM. It should be mentioned that all of the computation in the inference stage is performed on the CPU (not the GPU) since RLM seems more suitable for running on the CPU. If we calculate the inference of VINet on GPU, the inference times of VINet will be around 0.015 seconds for both types of VINets. For training the VINets with U-ENet and FNO-ENet, we take 33950 and 39687 seconds (approximately 9.4 and 11 hours) respectively. The training stage needs only to be calculated once, and the learned computationally cheap model can be repeatedly employed for similar inversion

Table 5: Average computing times of the RLM (using data $\boldsymbol{d}_{48}$ and $\boldsymbol{d}_{80}$), VINet with U-ENet, and VINet with FNO-ENet with mesh sizes $n = \{40 \times 40, 100 \times 100, 150 \times 150, 200 \times 200, 250 \times 250, 300 \times 300\}$ for the type 2 test datasets. The computing time is measured in seconds.

	RLM($\boldsymbol{d}_{48}$)	RLM($\boldsymbol{d}_{80}$)	VINet(U-ENet)	VINet(FNO-ENet)
Average time ($n = 30^2$)	0.71	1.26	-	-
Average time ($n = 100^2$)	17.27	29.09	0.1929	0.3882
Average time ($n = 150^2$)	53.21	83.32	0.1795	0.4011
Average time ($n = 200^2$)	126.01	210.33	0.1799	0.4313
Average time ($n = 250^2$)	245.93	407.71	0.1800	0.4697
Average time ($n = 300^2$)	495.78	827.83	0.1801	0.5295

tasks. Therefore, using large training times to exchange much slower and more accurate inversion results is reasonable.

4. Conclusion

In this paper, we have studied inverse problems governed by PDEs with non-i.i.d. noise assumptions under the Bayesian analysis framework. We have provided a rigorous illustration of Bayes' formula under the non-i.i.d. noise setting with the conjugate prior measure. To construct an efficient approximate sampling algorithm, we have deduced the mean-field based variational inference (MFVI) method with infinite-dimensional parameters. However, this deduced algorithm is time-consuming as a large amount of computationally expensive PDEs need to be solved. To address this issue, we have further constructed a parametric form of posterior measures based on an intuitive analysis of the approximate measure proposed in the MFVI method and proposed a general inference framework for inverse problems named VINet. By introducing measure equivalence conditions, we have generalized the evidence lower bound argument to the infinite-dimensional setting, which yields a computable training method. We have also provided a possible parametric strategy of the parametric posterior measure, which reflects the equivalent measure assumption that can be satisfied by designing parametric strategies appropriately. Finally, we have provided a possible specific network structure that yields a concrete VINet. The VINet has been applied to two inverse problems that are governed by a simple elliptic equation and the Helmholtz equation. For both inverse problems, the VINet was able to learn prior information of interested functions and noises from training examples efficiently and provide proper estimates quickly.

The current VINet framework is proposed based on the analysis of the MFVI method under the conjugate prior measure, which restricts the form of the posterior measures. Especially for nonlinear inverse problems, the VINet can only manage linearized problems and provide an approximate probability measure. As for the MFVI, the approximate probability measure may be inaccurate for some highly nonlinear problems. Hence, generalizing the

semi-conjugate VI methods to the infinite-dimensional setting and constructing the corresponding DNN-based inference method is worthy of investigation in future work. For the current preliminary studies on numerical aspects, we only employ the U-Net with a rescale layer and FNO, which is not designed for particular inverse problems. For specific inverse problems such as inverse scattering problems, special neural networks (Khoo and Ying, 2019) may be employed to construct the INet in our general statistical inference framework. For the CECInv layer, we only choose a rough discretized classical inverse algorithm for simplicity. It is interesting to choose the CECInv layer as some classical inverse methods with surrogates, e.g., polynomial approximations (Li and Marzouk, 2014) and Fourier neural operator (Li et al., 2020b). We will conduct further investigations on these issues in our future research.

5. Appendix

5.1 Bayesian well-posedness theory

For inverse problems, it is necessary to find a probability measure $\mu^{\boldsymbol{d}}$ on $\mathcal{H}$ (a separable Hilbert space), which is known as the posterior probability measure, and is specified by its density with respect to a prior probability measure μ_0. Here, we assume the data $\boldsymbol{d} \in \mathbb{R}^{N_d}$ as in the main text. The Bayes' formula on the Hilbert space is defined by

$$\frac{d\mu^{\boldsymbol{d}}}{d\mu_0}(x) = \frac{1}{Z_{\mu^{\boldsymbol{d}}}} \exp\big(-\Phi(x;\boldsymbol{d})\big), \tag{5.1}$$

where $\Phi(\cdot;\boldsymbol{d}) : \mathcal{H} \to \mathbb{R}$ is a continuous function, and $\exp\big(-\Phi(\cdot;\boldsymbol{d})\big)$ is integrable with respect to μ_0. The constant $Z_{\mu^{\boldsymbol{d}}}$ is chosen to ensure that $\mu^{\boldsymbol{d}}$ is a probability measure. Under the current setting, Assumption 1 and Theorems 15–16 in (Dashti and Stuart, 2017) can be stated as follows.

Assumption 13 *(Assumption 1 in (Dashti and Stuart, 2017)) Assume that $\Phi \in C(\mathcal{H} \times \mathbb{R}^{N_d}; \mathbb{R})$. Further assume that there are functions $M_i : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$, $i = 1, 2$, which are monotonic and non-decreasing separately in each argument, and with M_2 strictly positive. For all $x \in \mathcal{H}$ and $\boldsymbol{d}, \boldsymbol{d}_1, \boldsymbol{d}_2 \in B_{\mathbb{R}^{N_d}}(0, r) := \{\boldsymbol{d} \in \mathbb{R}^{N_d} : \|\boldsymbol{d}\|_{\mathbb{R}^{N_d}} < r\}$, the following holds:*

$$\Phi(x;\boldsymbol{d}) \geq -M_1(r, \|x\|_{\mathcal{H}}),$$
$$|\Phi(x;\boldsymbol{d}_1) - \Phi(x;\boldsymbol{d}_2)| \leq M_2(r, \|x\|_{\mathcal{H}})\|\boldsymbol{d}_1 - \boldsymbol{d}_2\|_{\mathbb{R}^d}.$$

Theorem 14 *(Theorem 15 in (Dashti and Stuart, 2017)) Let Assumption 13 hold. Assume that $\mu_0(\mathcal{H}) = 1$ and that $\mu_0(\mathcal{H} \cap B) > 0$ for some bounded set B in $\mathcal{H}$. Additionally, assume that for every fixed $r > 0$, $\exp\left(M_1(r, \|x\|_{\mathcal{H}})\right) \in L^1_{\mu_0}(\mathcal{H}; \mathbb{R})$, where $L^1_{\mu_0}(\mathcal{H}; \mathbb{R})$ represents $\mathcal{H}$-valued integrable functions under the measure μ_0. Then, for every $\boldsymbol{d} \in \mathbb{R}^{N_d}$, $Z_{\mu^{\boldsymbol{d}}}$ is positive and finite, and the probability measure $\mu^{\boldsymbol{d}}$ given by (5.1) is well-defined.*

Theorem 15 *(Theorem 16 in (Dashti and Stuart, 2017)) Let Assumption 13 hold. Assume that $\mu_0(\mathcal{H}) = 1$ and that $\mu_0(\mathcal{H} \cap B) > 0$ for some bounded set B in $\mathcal{H}$. Additionally, assume that, for every fixed $r > 0$,*

$$\exp\left(M_1(r, \|x\|_{\mathcal{H}})\right)\left(1 + M_2(r, \|x\|_{\mathcal{H}})^2\right) \in L^1_{\mu_0}(\mathcal{H}; \mathbb{R}).$$

Then, there is a $C = C(r) > 0$ such that, for all $\boldsymbol{d}, \boldsymbol{d}' \in B_{\mathbb{R}^{N_d}}(0, r)$,

$$d_{Hell}(\mu^{\boldsymbol{d}}, \mu^{\boldsymbol{d}'}) \leq C\|\boldsymbol{d} - \boldsymbol{d}'\|_{\mathbb{R}^{N_d}},$$

where $d_{Hell}(\cdot, \cdot)$ is the Hellinger distance between two probability measures.

5.2 The infinite-dimensional variational inference theory

In this section, we provide a brief introduction to the infinite-dimensional variational inference theory with the mean-field assumption. For interested readers, we suggest reading the paper (Jia et al., 2021b), which comprehensively describes the full ideas and theories of this topic. It should be emphasized that we have slightly improved the statement of Theorem 11 in (Jia et al., 2021b) to make the theory more applicable to practical problems. See Theorem 17 below for details.

Remembering the Bayes' formula (5.1) shown in Subsection 5.1 and denoting $\Phi(x) := \Phi(x; \boldsymbol{d})$, the variational inference method can be formulated as solving the following optimization problem:

$$\arg\min_{\nu \in \mathcal{A}} D_{\mathrm{KL}}(\nu || \mu^{\boldsymbol{d}}), \tag{5.2}$$

where $\mathcal{A} \subset \mathcal{M}(\mathcal{H})$ is a set of measures. Here $\mathcal{M}(\mathcal{H})$ is the set of Borel probability measures on $\mathcal{H}$.

For a fixed positive constant M, we assume the variable $x = (x_1, \ldots, x_M)$. Restricted to the settings in the main text, the constant $M = 2$ and $x = (u, \boldsymbol{\sigma})$. Generally, the Hilbert space $\mathcal{H}$ and set $\mathcal{A}$ can be specified as

$$\mathcal{H} = \prod_{j=1}^{M} \mathcal{H}_j, \qquad \mathcal{A} = \prod_{j=1}^{M} \mathcal{A}_j, \tag{5.3}$$

where $\mathcal{H}_j, j = 1, \cdots, M$ are separable Hilbert spaces and $\mathcal{A}_j \subset \mathcal{M}(\mathcal{H}_j)$. Denote $\nu := \prod_{i=1}^{M} \nu^i$ by a probability measure satisfying $\nu(dx) = \prod_{i=1}^{M} \nu^i(dx)$. Based on these assumptions, we can rewrite the minimization problem (5.2) as

$$\arg\min_{\nu^i \in \mathcal{A}_i} D_{\mathrm{KL}}\Big(\prod_{i=1}^{M} \nu^i \Big|\Big| \mu^{\boldsymbol{d}} \Big) \tag{5.4}$$

for suitable sets $\mathcal{A}_i$ with $i = 1, 2, \cdots, M$. Now, we introduce the approximate probability measure ν given in (5.2) which is equivalent to μ_0, defined by

$$\frac{d\nu}{d\mu_0}(x) = \frac{1}{Z_\nu} \exp\big(-\Phi_\nu(x)\big). \tag{5.5}$$

Comparing with the finite-dimensional case, we may assume that the potential $\Phi_\nu(x)$ can be decomposed as

$$\exp\left(-\Phi_\nu(x)\right) = \prod_{j=1}^{M} \exp\left(-\Phi_\nu^j(x_j)\right), \tag{5.6}$$

where $x = (x_1, \cdots, x_M)$. As illustrated in (Jia et al., 2021b), this approach does not incorporate the parameters in the prior probability measure into the hierarchical Bayes' model. Therefore, the following assumption needs to be introduced.

Assumption 16 *Let us introduce a reference probability measure*

$$\mu_r(dx) = \prod_{j=1}^{M} \mu_r^j(dx_j), \tag{5.7}$$

which is equivalent to the prior probability measure with the following relation:

$$\frac{d\mu_0}{d\mu_r}(x) = \frac{1}{Z_0} \exp(-\Phi^0(x)). \tag{5.8}$$

For each $j \in \{1, 2, \cdots, M\}$, there exists a predefined continuous function $a_j(\epsilon, x_j)$ satisfying $\mathbb{E}^{\mu_r^j}[a_j(\epsilon, \cdot)] < \infty$, where $\epsilon \in (0, \epsilon_0^j)$ with ϵ_0^j being a small positive number and $x_j \in \mathcal{H}_j$. Additionally, we assume that the approximate probability measure ν is equivalent to the reference measure μ_r and the Radon–Nikodym derivative of ν with respect to μ_r is given by

$$\frac{d\nu}{d\mu_r}(x) = \frac{1}{Z_r} \exp\Big(- \sum_{j=1}^{M} \Phi_j^r(x_j) \Big). \tag{5.9}$$

Following Assumption 16, we know that the approximate measure can be decomposed as $\nu(dx) = \prod_{j=1}^{M} \nu^j(dx_j)$ with

$$\frac{d\nu^j}{d\mu_r^j} = \frac{1}{Z_r^j} \exp\big(- \Phi_j^r(x_j)\big), \tag{5.10}$$

where $Z_r^j = \mathbb{E}^{\mu_r^j}\big(\exp\big(- \Phi_j^r(x_j)\big)\big)$ ensures that ν^j is a probability measure. For $j = 1, 2, \ldots, M$, let $\mathcal{Z}_j$ be defined as a Hilbert space embedded in $\mathcal{H}_j$. Then, for $j = 1, 2, \cdots, M$, we introduce

$$\mathrm{R}_j^1 = \Big\{ \Phi_j^r \,\Big|\, \sup_{1/N \le \|x_j\|_{\mathcal{Z}_j} \le N} \Phi_j^r(x_j) < \infty \text{ for all } N > 0 \Big\},$$

$$\mathrm{R}_j^2 = \Big\{ \Phi_j^r \,\Big|\, \int_{\mathcal{H}_j} \exp\big(-\Phi_j^r(x_j)\big) \max(1, a_j(\epsilon, x_j)) \mu_r^j(dx_j) < \infty, \text{ for } \epsilon \in [0, \epsilon_0^j) \Big\},$$

where ϵ_0^j and $a_j(\cdot, \cdot)$ are defined as in Assumption 16. With these preparations, we can define $\mathcal{A}_j$ $(j = 1, 2, \cdots, M)$ as follows:

$$\mathcal{A}_j = \left\{ \nu^j \in \mathcal{M}(\mathcal{H}_j) \,\middle|\, \begin{array}{l} \nu^j \text{ is equivalent to } \mu_r^j \text{ with (5.10) holding true,} \\ \text{and } \Phi_j^r \in \mathrm{R}_j^1 \cap \mathrm{R}_j^2 \end{array} \right\}. \tag{5.11}$$

Now, we state the main theorem that yields practical iterative algorithms.

Theorem 17 *Assume that the approximate probability measure in problem (5.4) satisfies Assumption 16. For $j = 1, 2, \cdots, M$, we define $T_N^j = \{x_j \,|\, 1/N \le \|x_j\|_{\mathcal{Z}_j} \le N\}$, where N is a positive constant. Each reference measure μ_r^j is assumed to satisfy $\sup_N \mu_r^j(T_N^j) = 1$. In addition, we assume*

$$\sup_{x_i \in T_N^i} \int_{\prod_{j\neq i} \mathcal{H}_j} \big(\Phi^0(x) + \Phi(x)\big)\, 1_A(x) \prod_{j\neq i} \nu^j(dx_j) < \infty \tag{5.12}$$

and

$$\int_{\mathcal{H}_i} \exp\bigg(- \int_{\prod_{j\neq i} \mathcal{H}_j} (\Phi^0(x) + \Phi(x)) 1_{A^c}(x) \prod_{j\neq i} \nu^j(dx_j)\bigg) M_i(x) \mu_r^i(dx_i) < \infty, \tag{5.13}$$

where $A := \{x \,|\, \Phi^0(x) + \Phi(x) \ge 0\}$, and $M_i(x) := \max\left(1, a_i(\epsilon, x_i)\right), i, j = 1, 2, \cdots, M$. Then, problem (5.4) admits a solution $\nu = \prod_{j=1}^M \nu^j \in \mathcal{M}(\mathcal{H})$ satisfying

$$\frac{d\nu}{d\mu_r} \propto \exp\bigg(-\sum_{i=1}^M \Phi_i^r(x_i)\bigg), \tag{5.14}$$

where

$$\Phi_i^r(x_i) = \int_{\prod_{j\neq i} \mathcal{H}_j} \bigg(\Phi^0(x) + \Phi(x)\bigg) \prod_{j\neq i} \nu^j(dx_j) + Const \tag{5.15}$$

and

$$\nu^i(dx_i) \propto \exp\big(-\Phi_i^r(x_i)\big)\mu_r^i(dx_i). \tag{5.16}$$

It should be pointed out that this theorem and the meanings of R_j^1 and R_j^2 $(j = 1, \ldots, M)$ are slightly different from the statements given in (Jia et al., 2021b). The proof of Theorem 11 in (Jia et al., 2021b) can be taken step by step to prove Theorem 17. In fact, the present version can be seen as an improved version of Theorem 11 in (Jia et al., 2021b), which can be verified more easily for practical problems.

5.3 Proof details

Proof of Theorem 3

Proof To prove the theorem, it suffices to verify the conditions given in Theorem 17. Specifically speaking, we need to verify the following four inequalities:

$$T_1 = \sup_{u \in T_N^u} \int_{(\mathbb{R}^+)^{N_d}} \Phi(u, \boldsymbol{\sigma}) \nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) < \infty,$$

$$T_2 = \sup_{\boldsymbol{\sigma} \in T_N^{\boldsymbol{\sigma}}} \int_{\mathcal{H}_u} \Phi(u, \boldsymbol{\sigma}) \nu^{\boldsymbol{\sigma}} \nu^u(du) < \infty,$$

$$T_3 = \int_{(\mathbb{R}^+)^{N_d}} \exp\bigg(- \int_{\mathcal{H}_u} \Phi(u, \boldsymbol{\sigma}) \nu^u(du)\bigg) \max(1, a(\epsilon, \boldsymbol{\sigma})) \mu_0^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) < \infty,$$

$$T_4 = \int_{\mathcal{H}_u} \exp\bigg(- \int_{(\mathbb{R}^+)^{N_d}} \Phi(u, \boldsymbol{\sigma}) \nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma})\bigg) \max(1, \|u\|_{\mathcal{H}_u}^2) \mu_0^u(du) < \infty.$$

For term T_1, we have

$$\begin{aligned}
T_1 \leq & \sup_{u\in T_N^u} \frac{1}{2}\int_{(\mathbb{R}^+)^{N_d}} \left(\sum_{k=1}^{N_d}\frac{1}{\sigma_k}\|\boldsymbol{d}-Hu\|_2^2+\sum_{k=1}^{N_d}\log\sigma_k\right)\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) \\
\leq & \left(1+\sup_{u\in T_N^u}\|\boldsymbol{d}-Hu\|_2^2\right)\int_{(\mathbb{R}^+)^{N_d}}\sum_{k=1}^{N_d}\left(\frac{1}{\sigma_k}+\log\sigma_k\right)\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) \\
\leq & C\int_{(\mathbb{R}^+)^{N_d}}\sum_{k=1}^{N_d}\left(\exp\left(\frac{\epsilon'}{\sigma_k}\right)+\sigma_k^{\epsilon'}\right)\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) \\
\leq & C\int_{(\mathbb{E}^+)^{N_d}}\exp\left(-\Phi_{\boldsymbol{\sigma}}(\boldsymbol{\sigma})\right)\max(1,a(2\epsilon',\boldsymbol{\sigma}))\mu_0^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma})<\infty,
\end{aligned}$$

where $\epsilon'<\epsilon_0/2$ and C is a generic constant that may be different from line to line.

For term T_2, we have

$$T_2 \leq \sup_{\boldsymbol{\sigma}\in T_N^{\boldsymbol{\sigma}}}\frac{1}{2}\sum_{k=1}^{N_d}\left(\frac{1}{\sigma_k}+\log\sigma_k\right)\int_{\mathcal{H}_u}\left(1+\|\boldsymbol{d}-Hu\|_2^2\right)\nu^u(du).$$

Considering $\nu^u\in\mathcal{A}_u$, we get

$$\int_{\mathcal{H}_u}\left(1+\|\boldsymbol{d}-Hu\|_2^2\right)\nu^u(du)\leq C\int_{\mathcal{H}_u}(1+\|u\|_{\mathcal{H}_u}^2)\nu^u(du)<\infty$$

and

$$\sum_{k=1}^{N_d}\left(\frac{1}{\sigma_k}+\log\sigma_k\right)\leq\sum_{k=1}^{N_d}\left(\frac{1}{\sigma_k}+\sigma_k\right)\leq 2NN_d<\infty,$$

which indicates $T_2<\infty$.

Using the estimate

$$\exp\left(-\int_{\mathcal{H}_u}\frac{1}{2}\left(\|d-Hu\|_\Sigma^2+\sum_{k=1}^{N_d}\log\sigma_k\right)\nu^u(du)\right)\leq\exp\left(-\frac{1}{2}\sum_{k=1}^{N_d}\log\sigma_k\right),$$

we can estimate T_3 as follows:

$$\begin{aligned}
T_3 \leq & \int_{(\mathbb{R}^+)^{N_d}}\prod_{k=1}^{N_d}\sigma_k^{-1/2}\max\left(1,a(\epsilon,\boldsymbol{\sigma})\right)\mu_0^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma}) \\
\leq & \sqrt{N_d}\int_{(\mathbb{R}^+)^{N_d}}\sum_{k=1}^{N_d}\sigma_k^{-1/2}\max(1,a(\epsilon,\boldsymbol{\sigma}))\mu_0^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma})<\infty,
\end{aligned}$$

where the last inequality follows from the properties of inverse Gamma distribution. Notice that

$$\begin{aligned}
&\exp\left(-\frac{1}{2}\int_{(\mathbb{R}^+)^{N_d}}\|\boldsymbol{d}-Hu\|_2^2+\sum_{k=1}^{N_d}\log\sigma_k\nu^{\boldsymbol{\sigma}}(d\boldsymbol{\sigma})\right) \\
&\qquad\leq\exp\left(-\frac{1}{2}\sum_{k=1}^{N_d}\int_{(\mathbb{R}^+)^{N_d}}\log\sigma_k\nu^{\boldsymbol{\sigma}}(\boldsymbol{\sigma})\right)<\infty.
\end{aligned}$$

Then, for term T_4, we have

$$T_4 \leq C \int_{\mathcal{H}_u} \max(1, \|u\|^2_{\mathcal{H}_u}) \mu_0^u(du) < \infty. \tag{5.17}$$

The proof is completed by noting the general theory shown in Subsection 5.2 in the Appendix. ■

Proof of Theorem 6

Proof Let us firstly give the explicit form of the operator H^*. Taking $\boldsymbol{f} = (f_1, \ldots, f_{N_d})^T \in \mathbb{R}^{N_d}$ and $g = \sum_{k=1}^{\infty} g_k e_k \in \mathcal{H}_u$, we have

$$\langle H^* \boldsymbol{f}, g \rangle_{L^2} = \langle \boldsymbol{f}, Hg \rangle_{\ell^2} = \sum_{k=1}^{\infty} \lambda_k g_k \langle \boldsymbol{f}, Se_k \rangle_{\ell^2} = \sum_{k=1}^{\infty} \lambda_k g_k \sum_{i=1}^{N_d} f_i e_k(x_i),$$

which implies

$$H^* \boldsymbol{f} = \sum_{k=1}^{\infty} \left(\sum_{i=1}^{N_d} f_i \lambda_k e_k(x_i) \right) e_k(x). \tag{5.18}$$

Then, we have

$$\begin{aligned} H^* \Sigma^*_{\text{inv}} H \bar{u}_0 &= \sum_{k=1}^{\infty} \lambda_k \bar{u}_{0k} H^* \Sigma^*_{\text{inv}} S e_k \\ &= \sum_{k=1}^{\infty} \lambda_k \bar{u}_{0k} H^* \left(e_k(x_1) \alpha_1^* / \beta_1^*, \ldots, e_k(x_{N_d}) \alpha_{N_d}^* / \beta_{N_d}^* \right)^T \\ &= \sum_{j=1}^{\infty} \left(\sum_{i=1}^{N_d} \frac{\alpha_i^*}{\beta_i^*} \sum_{k=1}^{\infty} \lambda_k \bar{u}_{0k} e_k(x_i) \lambda_j e_j(x_i) \right) e_j(x). \end{aligned} \tag{5.19}$$

With these preparations, we find that

$$\begin{aligned} &(H^* \Sigma^*_{\text{inv}} H + \mathcal{C}_0^{-1})^{-1} \mathcal{C}_0^{-1} \bar{u}_0 = \\ &\qquad \sum_{j=1}^{\infty} \frac{\alpha_j^{-1}}{\sum_{i=1}^{N_d} \frac{\alpha_i^*}{\beta_i^*} \sum_{k=1}^{\infty} \lambda_k \frac{\bar{u}_{0k}}{\bar{u}_{0j}} e_k(x_i) \lambda_j e_j(x_i) + \alpha_j^{-1}} \bar{u}_{0j} e_j(x), \end{aligned} \tag{5.20}$$

which obviously implies the desired conclusion. ■

Proof of Theorem 9

Proof Based on Eq. (2.34), we have

$$\begin{aligned} \bar{u}_p =& \mathcal{C}_p \left[(H^* \Sigma^*_{\text{inv}} H + \mathcal{C}_0^{-1}) u^\dagger + H^* \Sigma^*_{\text{inv}} \boldsymbol{\epsilon} + \mathcal{C}_0^{-1} (\bar{u}_0 - u^\dagger) \right] \\ =& u^\dagger + \mathcal{C}_p \left[H^* \Sigma^*_{\text{inv}} \boldsymbol{\epsilon} + \mathcal{C}_0^{-1} (\bar{u}_0 - u^\dagger) \right], \end{aligned} \tag{5.21}$$

which yields

$$\begin{aligned}\mathbb{E}_0\left[\|\bar{u}_p - u^\dagger\|_{\mathcal{H}}\right] \leq & \mathbb{E}_0\left[\|\mathcal{C}_p H^* \Sigma_{\text{inv}}^* \boldsymbol{\epsilon}\|_{\mathcal{H}_u}\right] + \|\mathcal{C}_p \mathcal{C}_0^{-1}(\bar{u}_0 - u^\dagger)\|_{\mathcal{H}_u} \\ \leq & \mathbb{E}_0\left[\|\mathcal{C}_p H^* \Sigma_{\text{inv}}^* \boldsymbol{\epsilon}\|_{\mathcal{H}_u}^2\right]^{1/2} + \|\bar{u}_0 - u^\dagger\|_{\mathcal{H}_u} \\ = & \text{tr}\left(\mathcal{C}_p H^* \Sigma_{\text{inv}}^* \Sigma \Sigma_{\text{inv}}^* H \mathcal{C}_p^*\right)^{1/2} + \|\bar{u}_0 - u^\dagger\|_{\mathcal{H}_u}.\end{aligned} \tag{5.22}$$

Now, we have

$$\mathbb{E}_0[\|\bar{u}_p(\boldsymbol{d}; W_I) - u^\dagger\|_{\mathcal{H}_u}] \leq \mathbb{E}_0[\|\bar{u}_p(\boldsymbol{d}; W_I) - \bar{u}_p\|_{\mathcal{H}_u}] + \mathbb{E}_0[\|\bar{u}_p - u^\dagger\|_{\mathcal{H}_u}], \tag{5.23}$$

which yields the required estimate by inserting the above inequality (5.22). ■

Proof of Theorem 11

Proof Because the operator A is assumed to be a self-adjoint and positive definite, it holds for appropriate f that

$$\langle(\underline{a} + \delta A^{\alpha/2})f, f\rangle \leq \langle(a + \delta A^{\alpha/2})f, f\rangle \leq \langle(\bar{a} + \delta A^{\alpha/2})f, f\rangle, \tag{5.24}$$

where we denote $a := a(\boldsymbol{d}; W_I)$ and omit all of the identity operators. For concision, we omit the identity operators in the proof when there are no ambiguities from the context. Obviously, we also have

$$\langle(\underline{a} + \delta A^{\alpha/2})^2 f, f\rangle \leq \langle(a + \delta A^{\alpha/2})^2 f, f\rangle \leq \langle(\bar{a} + \delta A^{\alpha/2})^2 f, f\rangle, \tag{5.25}$$

when choosing appropriate f in each estimate. According to Theorem 2.25 (Feldman-Hajek theorem) in (Prato and Zabczyk, 2014), we only need to prove the following two results:

1. $\mathcal{C}_{\epsilon_0}^{1/2}\mathcal{H} = \mathcal{C}_p^{1/2}\mathcal{H} = \mathcal{H}_0$,
2. the operator $T := (\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}^{1/2})(\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}^{1/2})^* - \text{Id}$ is a Hilbert-Schmidt operator on $\bar{\mathcal{H}}_0$.

First, we prove (1). Taking arbitrary $u \in \mathcal{C}_{\epsilon_0}^{1/2}\mathcal{H}$ gives

$$u = \mathcal{C}_{\epsilon_0}^{1/2} v \quad \text{for some } v \in \mathcal{H}. \tag{5.26}$$

The variable can be rewritten as $u = \mathcal{C}_p^{1/2}\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}^{1/2} v$. Considering (5.25), we have

$$\begin{aligned}\|\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}^{1/2} v\|_{\mathcal{H}}^2 = & \|(a + \delta A^{\alpha/2})(\epsilon_0^{-1} + \delta A^{\alpha/2})^{-1} v\|_{\mathcal{H}}^2 \\ \leq & \|(\bar{a} + \delta A^{\alpha/2})(\epsilon_0^{-1} + \delta A^{\alpha/2})^{-1} v\|_{\mathcal{H}}^2 \\ = & \sum_{k=1}^{\infty} \left(\frac{\bar{a} + \delta \lambda_k^{\alpha/2}}{\epsilon_0^{-1} + \delta \lambda_k^{\alpha/2}}\right)^2 v_k^2 \\ \leq & C\|v\|_{\mathcal{H}}^2 < \infty,\end{aligned} \tag{5.27}$$

which implies $u \in \mathcal{C}_p^{1/2}\mathcal{H}$. Conversely, we assume $u \in \mathcal{C}_p^{1/2}\mathcal{H}$ with $u = \mathcal{C}_p^{1/2}v$ for some $v \in \mathcal{H}$. Similarly, we can rewrite $u = \mathcal{C}_{\epsilon_0}^{1/2}\mathcal{C}_{\epsilon_0}^{-1/2}\mathcal{C}_p^{1/2}v$. In the following, we give estimates of $\mathcal{C}_{\epsilon_0}^{-1/2}\mathcal{C}_p^{1/2}v$ which is more complex than the estimate (5.27). Obviously, we have

$$\|\mathcal{C}_{\epsilon_0}^{-1/2}\mathcal{C}_p^{1/2}v\|_{\mathcal{H}}^2 = \sum_{k,\ell=1}^{\infty} v_k v_\ell \langle \mathcal{C}_{\epsilon_0}^{-1/2}\mathcal{C}_p^{1/2}e_k, \mathcal{C}_{\epsilon_0}^{-1/2}\mathcal{C}_p^{1/2}e_\ell\rangle_{\mathcal{H}} = \sum_{k,\ell=1}^{\infty} v_k v_\ell B_{k,\ell}. \tag{5.28}$$

For the term $B_{k,\ell}$, we have

$$\begin{aligned} B_{k,\ell} =& \langle (a-\epsilon_0^{-1})\mathcal{C}_p^{1/2}e_k, (a-\epsilon_0^{-1})\mathcal{C}_p^{1/2}e_\ell\rangle_{\mathcal{H}} + 2\langle(\epsilon_0^{-1}-a)\mathcal{C}_p^{1/2}e_k, e_\ell\rangle + \langle e_k, e_\ell\rangle \\ =& B_{k,\ell}^1 + B_{k,\ell}^2 + B_{k,\ell}^3. \end{aligned} \tag{5.29}$$

For the term $B_{k,\ell}^1$, we have

$$\begin{aligned} B_{k,\ell}^1 \leq& C\|(a+\delta A^{\alpha/2})^{-1}e_k\|_{\mathcal{H}}\|(a+\delta A^{\alpha/2})^{-1}e_\ell\|_{\mathcal{H}} \\ \leq& C\|(\underline{a}+\delta A^{\alpha/2})^{-1}e_k\|_{\mathcal{H}}\|(\underline{a}+\delta A^{\alpha/2})^{-1}e_\ell\|_{\mathcal{H}} \\ =& C\frac{1}{\underline{a}+\delta\lambda_k^{\alpha/2}}\frac{1}{\underline{a}+\delta\lambda_\ell^{\alpha/2}}. \end{aligned} \tag{5.30}$$

Hence, we further obtain

$$\sum_{k,\ell=1}^{\infty} B_{k,\ell}^1 \leq C\left[\sum_{k=1}^{\infty}\left(v_k^2 + \left(\frac{1}{\underline{a}+\delta\lambda_k^{\alpha/2}}\right)^2\right)\right]^2 < \infty. \tag{5.31}$$

For the term $B_{k,\ell}^2$, we have the following estimate

$$\begin{aligned} \sum_{k,\ell=1}^{\infty} v_k v_\ell B_{k,\ell}^2 =& 2\sum_{k}^{\infty} v_k \langle(\epsilon_0^{-1}-a)\mathcal{C}_p^{1/2}e_k, \sum_{\ell=1}^{\infty} v_\ell e_\ell\rangle_{\mathcal{H}} \\ \leq& C\sum_{k=1}^{\infty}\left(v_k^2 + \left(\frac{1}{\underline{a}+\delta\lambda_k^{\alpha/2}}\right)^2\right)\|v\|_{\mathcal{H}} < \infty. \end{aligned} \tag{5.32}$$

For the term $B_{k,\ell}^3$, we find

$$\sum_{k,\ell=1}^{\infty} v_k v_\ell B_{k,\ell}^2 \leq \|v\|_{\mathcal{H}}^2 < \infty. \tag{5.33}$$

Inserting estimates from (5.31) to (5.33) into (5.28), we arrive at

$$\|\mathcal{C}_{\epsilon_0}^{-1/2}\mathcal{C}_p^{1/2}v\|_{\mathcal{H}}^2 < \infty. \tag{5.34}$$

Hence, the proof of (1) is completed. Before going further, we denote $\langle\cdot,\cdot\rangle_{\mathcal{H}_0} := \langle\cdot,\cdot\rangle_{\mathcal{C}_{\epsilon_0}^{1/2}\mathcal{H}} = \langle(\epsilon_0+\delta A^{\alpha/2})\cdot, (\epsilon_0+\delta A^{\alpha/2})\cdot\rangle_{\mathcal{H}}$. For proving (2), we introduce $\tilde{e}_j = \frac{1}{\epsilon_0^{-1}+\delta\lambda_j^{\alpha/2}}e_j$ for $j =$

$1, 2, \ldots$, which is an orthonormal basis on $\mathcal{C}_{\epsilon_0}^{1/2}\mathcal{H}$. Following simple calculations, we obtain

$$\begin{aligned}\sum_{j=1}^{\infty}\langle T\tilde{e}_j, T\tilde{e}_j\rangle_{\mathcal{H}_0} =& \sum_{j=1}^{\infty}\langle(\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}\mathcal{C}_p^{-1/2} - \mathrm{Id})\tilde{e}_j, (\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}\mathcal{C}_p^{-1/2} - \mathrm{Id})\tilde{e}_j\rangle_{\mathcal{H}_0} \\ =& \sum_{j=1}^{\infty}(I_j^1 + I_j^2 + I_j^3),\end{aligned} \tag{5.35}$$

where

$$I_j^1 = \langle(\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}\mathcal{C}_p^{-1/2} - \mathcal{C}_{\epsilon_0}^{1/2}\mathcal{C}_p^{-1/2})\tilde{e}_j, (\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}\mathcal{C}_p^{-1/2} - \mathcal{C}_{\epsilon_0}^{1/2}\mathcal{C}_p^{-1/2})\tilde{e}_j\rangle_{\mathcal{H}_0},$$

$$I_j^2 = 2\langle(\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}\mathcal{C}_p^{-1/2} - \mathcal{C}_{\epsilon_0}^{1/2}\mathcal{C}_p^{-1/2})\tilde{e}_j, (\mathcal{C}_{\epsilon_0}^{1/2}\mathcal{C}_p^{-1/2} - \mathrm{Id})\tilde{e}_j\rangle_{\mathcal{H}_0},$$

$$I_j^3 = \langle(\mathcal{C}_{\epsilon_0}^{1/2}\mathcal{C}_p^{-1/2} - \mathrm{Id})\tilde{e}_j, (\mathcal{C}_{\epsilon_0}^{1/2}\mathcal{C}_p^{-1/2} - \mathrm{Id})\tilde{e}_j\rangle_{\mathcal{H}_0}.$$

Since I_j^2 can be bounded by I_j^1 combined with I_j^3, we only focus on the estimates of I_j^1 and I_j^3 in the following.

Estimate of I_j^1: Because

$$\begin{aligned}\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}^{1/2} - \mathrm{Id} =& (a + \delta A^{\alpha/2})(\epsilon_0^{-1} + \delta A^{\alpha/2})^{-1} - \mathrm{Id} \\ =& \left(a + \delta A^{\alpha/2} - \epsilon_0^{-1} - \delta A^{\alpha/2}\right)(\epsilon_0^{-1} + \delta A^{\alpha/2})^{-1} \\ =& (a - \epsilon_0^{-1})(\epsilon_0^{-1} + \delta A^{\alpha/2})^{-1},\end{aligned} \tag{5.36}$$

we have

$$\begin{aligned}I_j^1 =& \langle(\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}^{1/2} - \mathrm{Id})\mathcal{C}_{\epsilon_0}^{1/2}\mathcal{C}_p^{-1/2}\tilde{e}_j, (\mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}^{1/2} - \mathrm{Id})\mathcal{C}_{\epsilon_0}^{1/2}\mathcal{C}_p^{-1/2}\tilde{e}_j\rangle_{\mathcal{H}_0} \\ =& \langle(a - \epsilon_0^{-1})B\tilde{e}_j, (a - \epsilon_0^{-1})B\tilde{e}_j\rangle_{\mathcal{H}_0} \\ =& \langle\mathcal{C}_{\epsilon_0}^{-1/2}[(a - \epsilon_0^{-1})B\tilde{e}_j], \mathcal{C}_{\epsilon_0}^{-1/2}[(a - \epsilon_0^{-1})B\tilde{e}_j]\rangle_{\mathcal{H}} \\ \leq& 2I_j^{11} + 2I_j^{12},\end{aligned} \tag{5.37}$$

where $B\tilde{e}_j := \mathcal{C}_{\epsilon_0}\mathcal{C}_p^{-1/2}\tilde{e}_j$ and

$$\begin{aligned}I_j^{11} &= \langle(\epsilon_0^{-1} + \delta A^{\alpha/2})[aB\tilde{e}_j], (\epsilon_0^{-1} + \delta A^{\alpha/2})[aB\tilde{e}_j]\rangle_{\mathcal{H}}, \\ I_j^{12} &= \epsilon_0^{-2}\langle(\epsilon_0^{-1} + \delta A^{\alpha/2})B\tilde{e}_j, (\epsilon_0^{-1} + \delta A^{\alpha/2})B\tilde{e}_j\rangle_{\mathcal{H}}.\end{aligned}$$

For the term I_j^{11}, we have

$$\begin{aligned}I_j^{11} =& \langle(\epsilon_0^{-1} + \delta A^{\alpha/2})\sum_{k=1}^{\infty}\langle aB\tilde{e}_j, e_k\rangle_{\mathcal{H}}e_k, (\epsilon_0^{-1} + \delta A^{\alpha/2})\sum_{\ell=1}^{\infty}\langle aB\tilde{e}_j, e_\ell\rangle_{\mathcal{H}}e_\ell\rangle_{\mathcal{H}} \\ =& \sum_{k=1}^{\infty}\langle aB\tilde{e}_j, e_k\rangle_{\mathcal{H}}^2\left(\frac{1}{\epsilon_0^{-1} + \delta\lambda_k^{\alpha/2}}\right)^2 \\ =& \left(\frac{1}{\epsilon_0^{-1} + \delta\lambda_j^{\alpha/2}}\right)^2\sum_{k=1}^{\infty}\left(\frac{1}{\epsilon_0^{-1} + \delta\lambda_k^{\alpha/2}}\right)^2\langle e_j, \mathcal{C}_p^{-1/2}\mathcal{C}_{\epsilon_0}(ae_k)\rangle_{\mathcal{H}}^2.\end{aligned} \tag{5.38}$$

Because

$$\begin{aligned}\sum_{j=1}^{\infty}\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_j^{\alpha/2}}\right)^2\langle e_j,(a+\delta A^{\alpha/2})(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-2}(ae_k)\rangle_{\mathcal{H}}^2\\ \leq C\|ae_k\|_{\mathcal{H}}^2\sum_{j=1}^{\infty}\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_j^{\alpha/2}}\right)^2<\infty,\end{aligned}\tag{5.39}$$

we obtain

$$\sum_{j=1}^{\infty}I_j^{11}\leq C\|ae_k\|_{\mathcal{H}}^2\sum_{j=1}^{\infty}\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_j^{\alpha/2}}\right)^2\sum_{k=1}^{\infty}\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_k^{\alpha/2}}\right)^2<\infty.\tag{5.40}$$

For the term I_j^{12}, we have

$$\begin{aligned}I_j^{12}=&\langle(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}(a+\delta A^{\alpha/2})\tilde{e}_j,(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}(a+\delta A^{\alpha/2})\tilde{e}_j\rangle_{\mathcal{H}}\\ \leq&2\langle(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}(a\tilde{e}_j),(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}(a\tilde{e}_j)\rangle_{\mathcal{H}}\\ &+2\delta^2\langle(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}A^{\alpha/2}\tilde{e}_j,(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}A^{\alpha/2}\tilde{e}_j\rangle_{\mathcal{H}}.\end{aligned}\tag{5.41}$$

For the first term on the right-hand side of the above inequality, we have

$$\begin{aligned}&\langle(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}(a\tilde{e}_j),(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}(a\tilde{e}_j)\rangle_{\mathcal{H}}\\ =&\langle(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}\sum_{k=1}^{\infty}\langle a\tilde{e}_j,e_k\rangle e_k,(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}\sum_{\ell=1}^{\infty}\langle a\tilde{e}_j,e_\ell\rangle e_\ell\rangle_{\mathcal{H}}\\ =&\sum_{k=1}^{\infty}\langle a\tilde{e}_j,e_k\rangle_{\mathcal{H}}^2\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_k^{\alpha/2}}\right)^2\\ \leq&\bar{a}^2\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_j^{\alpha/2}}\right)^2\sum_{k=1}^{\infty}\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_k^{\alpha/2}}\right)^2.\end{aligned}\tag{5.42}$$

For the second term on the right-hand side of (5.41), we have

$$\langle(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}A^{\alpha/2}\tilde{e}_j,(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}A^{\alpha/2}\tilde{e}_j\rangle_{\mathcal{H}}\leq C\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_j^{\alpha/2}}\right)^2.\tag{5.43}$$

Substituting estimates (5.42) and (5.43) into (5.41), we obtain

$$\sum_{j=1}^{\infty}I_j^{12}\leq C\sum_{j=1}^{\infty}\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_j^{\alpha/2}}\right)^2<\infty.\tag{5.44}$$

Combining estimates (5.37), (5.40), and (5.44) leads to

$$\sum_{j=1}^{\infty}I_j^1<\infty.\tag{5.45}$$

Estimate of I_j^3: For the term I_j^3, we find that

$$\begin{aligned} I_j^3 =&\langle(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}[(a-\epsilon_0^{-1})\tilde{e}_j],(\epsilon_0^{-1}+\delta A^{\alpha/2})^{-1}[(a-\epsilon_0^{-1})\tilde{e}_j]\rangle_{\mathcal{H}_0}\\ =&\langle(a-\epsilon_0^{-1})\tilde{e}_j,(a-\epsilon_0^{-1})\tilde{e}_j\rangle_{\mathcal{H}}\\ \leq& C\left(\frac{1}{\epsilon_0^{-1}+\delta\lambda_j^{\alpha/2}}\right)^2, \end{aligned} \tag{5.46}$$

which implies

$$\sum_{j=1}^{\infty} I_j^3 < \infty. \tag{5.47}$$

Combining estimates of I_j^1 and I_j^3, we obtain

$$\sum_{j=1}^{\infty}\langle T\tilde{e}_j, T\tilde{e}_j\rangle_{\mathcal{H}_0} < \infty, \tag{5.48}$$

which indicates that T is a Hilbert–Schmidt operator. Hence, the proof is completed. ■

5.4 More numerical results of simple smoothing model

In this subsection, let us provide the results for TSVD, MFVI, and VINet obtained by using the type 1 test datasets generated from the same probability measure as the training datasets.

In Figure 11, we depict the estimated noise variances by the MFVI and VINet and the background true noise variances for three randomly selected testing examples. In the first line of Figure 11, the estimated variances of all 400 measurement points are given with the dashed red line representing the true noise variances, the dashed-dotted light orange line representing the estimation given by MFVI, and the solid green line representing the estimation given by the SNet. As in the main test, the SNet provides accurate estimates of the noise variances for all of the three type 1 test datasets. In the second line of Figure 11, we use the solid blue line and dashed-dotted orange line to represent the square real noises, i.e., $(\boldsymbol{d}-\boldsymbol{d}_c)^2$, and the estimations given by MFVI, respectively. Obviously, the MFVI method only captures some features of the square real noises but not the true noise variances.

In Figure 12, we show the background truth and estimates obtained by the TSVD, MFVI, VINet with U-ENet, and VINet with FNO-ENet for one randomly selected data from the type 1 testing datasets. Similar conclusions can be drawn as in the main text: the TSVD and MFVI methods can only capture some major trends of the truth, while the two types of VINets provide more detailed structures.

At last, in Table 6, we provide a detailed comparison of TSVD, MFVI, VINet with U-ENet, and VINet with FNO-ENet with various discrete meshes. Similar to the results in the main text, the relative errors of the two VINets are lower than those of the TSVD and MFVI methods. All of the methods are stable with respect to different discrete meshes.

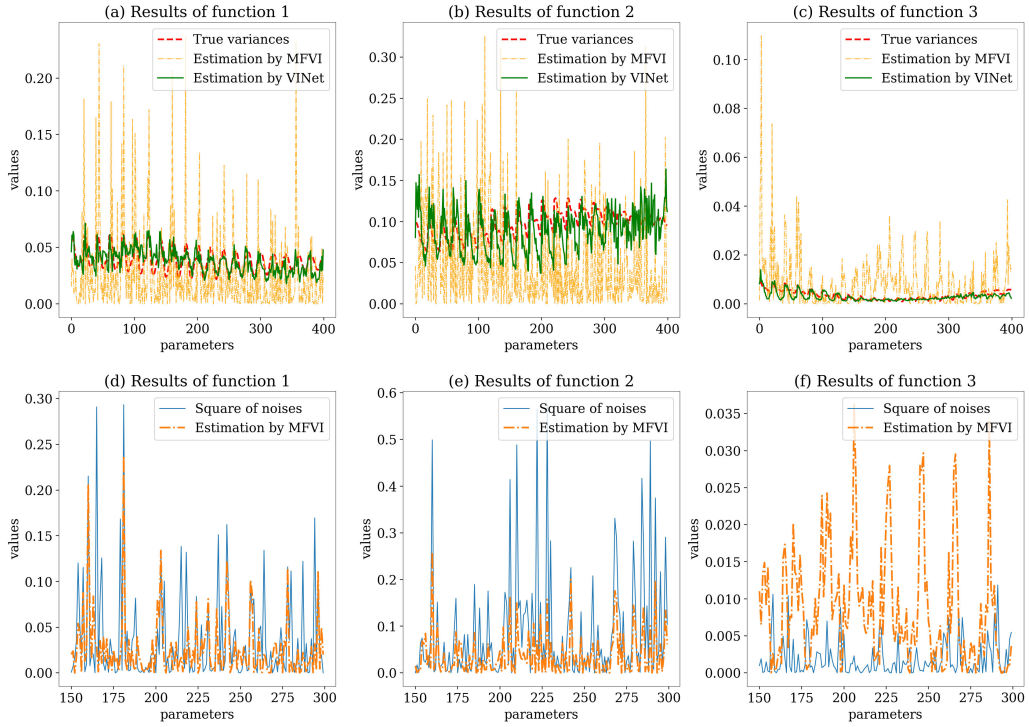

Figure 11: Comparison of the true noise variances and estimated noise variances by the MFVI and VINet (output of SNet). (a) (b) (c) Noise variances of all parameters for three different testing examples. (d) (e) (f) Noise variances estimated by MFVI and the square real noises of some parameters.

Table 6: Average relative errors of the estimates obtained by the TSVD, MFVI, VINet (U-ENet), and VINet (FNO-ENet) with mesh sizes $n = \{30 \times 30, 50 \times 50, 100 \times 100, 150 \times 150, 200 \times 200, 250 \times 250, 300 \times 300\}$ for the type 1 test datasets.

	TSVD	MFVI	VINet (U-ENet)	VINet (FNO-ENet)
Relative error ($n = 30^2$)	0.1406	0.2301	0.0878	0.0786
Relative error ($n = 50^2$)	0.1413	0.2328	0.0933	0.0822
Relative error ($n = 100^2$)	0.1457	0.2499	0.0972	0.0854
Relative error ($n = 150^2$)	0.1481	0.2504	0.0982	0.0865
Relative error ($n = 200^2$)	0.1494	0.2371	0.0989	0.0870
Relative error ($n = 250^2$)	0.1498	0.2499	0.0991	0.0871
Relative error ($n = 300^2$)	0.1499	0.2406	0.0991	0.0871

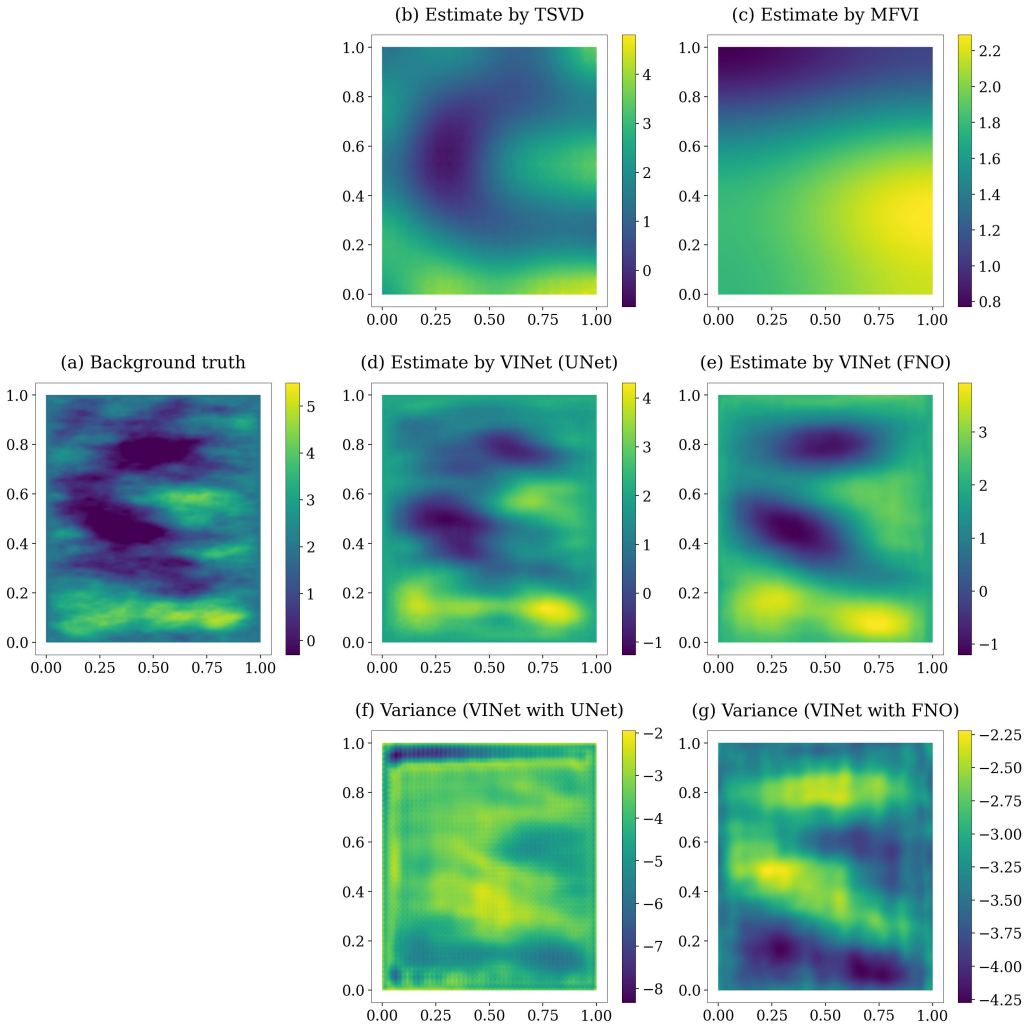

Figure 12: (a) Background true function randomly selected from type 1 testing datasets; (b) Estimate obtained by the TSVD method; (c) Estimate obtained by the MFVI method; (d) Estimate obtained by the VINet with U-ENet; (e) Estimate obtained by the VINet with FNO-ENet; (f) Variance function obtained by the VINet with U-ENet shown in log-scale; (g) Variance function obtained by the VINet with FNO-ENet shown in log-scale.

5.5 More numerical results of inverse source problem

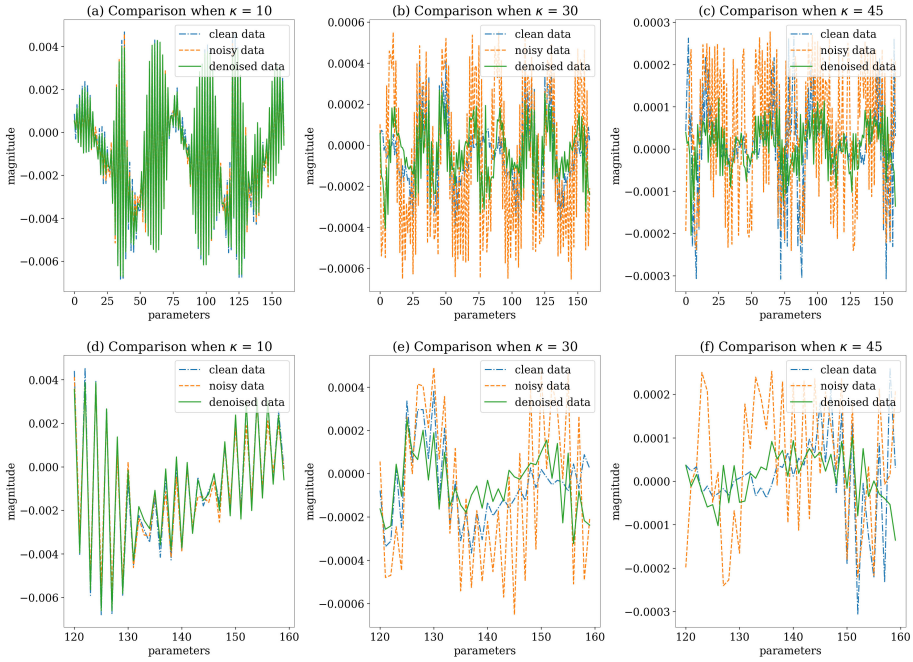

Figure 13: Noisy data (data generated by accurate forward solver with random Gaussian noise), clean data (data generated by rough forward solver), and denoised data (the output of the DNet) for a randomly selected source function u from type 1 test datasets. (a) Data when the wavenumber $\kappa = 10$; (b) Data when the wavenumber $\kappa = 30$; (c) Data when the wavenumber $\kappa = 45$; (d)(e)(f) Partly enlarged versions of (a), (b), and (c), respectively.

As in the main text, we first show the results of DNet. In Figure 13, we use the dash-dotted blue line, dashed orange line, and solid green line to represent the clean data, noisy data, and the output of the DNet, respectively. Sub-figures (a), (b), and (c) provide comparisons when the wavenumbers are chosen to be 10, 30, and 45. To provide more detail, we exhibit sub-figures (d), (e), and (f) that demonstrate the partly enlarged versions of sub-figures (a), (b), and (c), respectively. Similar to the main text, the noisy data deviate largely from the clean data and the outputs of DNet are visually similar to the clean data when the wavenumber $\kappa = 30, 45$. Hence, the DNet learns the sophisticated model error mechanism between the accurate and rough forward solvers.

In Figure 14, we exhibit the results of one randomly selected test example from the type 1 test datasets. Similar results can be obtained as in the main text. The classical RLM can hardly handle the discontinuity, even when the highest wavenumber is set to be 80. VINets with U-ENet and FNO-ENet both yield more accurate estimations.

Finally, we give a detailed comparison for RLM and VINet with various discrete meshes. In Table 7, we show the average relative errors on different meshes obtained by RLM with $\boldsymbol{d}_{48}$, RLM with $\boldsymbol{d}_{80}$, VINet with U-ENet, and VINet with FNO-ENet, respectively. We

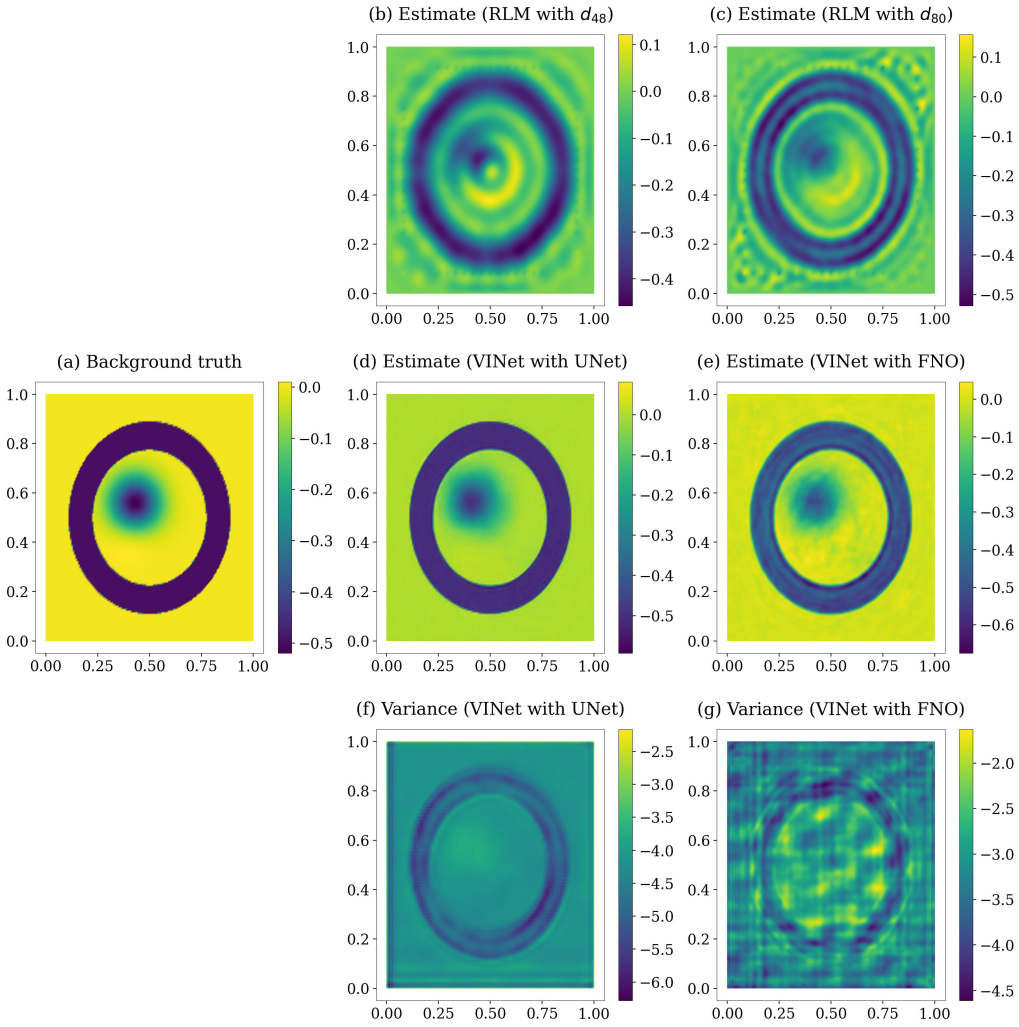

Figure 14: (a) One of the background true function in type 1 test datasets; (b) Estimate obtained by RLM with $\boldsymbol{d}_{48}$; (c) Estimate obtained by RLM with $\boldsymbol{d}_{80}$; (d) Estimate obtained by VINet with U-ENet; (e) Estimate obtained by VINet with FNO-ENet; (f) Estimated variance function by VINet with U-ENet shown in log-scale; (g) Estimated variance function by VINet with FNO-ENet shown in log-scale.

Table 7: Average relative errors of the estimates obtained by the RLM (using data $\boldsymbol{d}_{48}$ and $\boldsymbol{d}_{80}$), VINet with U-ENet, and VINet with FNO-ENet with mesh sizes $n = \{30 \times 30, 100 \times 100, 150 \times 150, 200 \times 200, 250 \times 250, 300 \times 300\}$ for the type 1 test datasets.

	RLM($\boldsymbol{d}_{48}$)	RLM($\boldsymbol{d}_{80}$)	VINet(U-ENet)	VINet(FNO-ENet)
Relative error ($n = 30^2$)	0.3349	0.3250	-	-
Relative error ($n = 100^2$)	0.2310	0.2321	0.0474	0.1163
Relative error ($n = 150^2$)	0.2114	0.1932	0.0426	0.0443
Relative error ($n = 200^2$)	0.2087	0.1728	0.0406	0.0449
Relative error ($n = 250^2$)	0.2084	0.1653	0.0405	0.0462
Relative error ($n = 300^2$)	0.2100	0.1624	0.0452	0.0489

can see that the discrete level indeed influences the performance of the classical RLM as illustrated in the main text. By comparing Tables 5 and 7, we see that the relative errors of VINet with FNO-ENet are both around 4%, but the relative errors of VINet with U-ENet are around 4% and 6%, respectively. Hence, under the current setting, the VINet with U-ENet seems to have weaker generalization properties compared with the VINet with FNO-ENet.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFA0713900 in part by the National Natural Science Foundation of China (NSFC) Project under Grants 12271428, 12090020, and 12090021, in part by the NSF grant DMS-2208256, in part by the Macao Science and Technology Development Fund under Grant 061/2020/A2, and in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2023-JC-QN-0035.

References

J. Adler and O. Öktem. Deep Bayesian inversion. arXiv:1811.05910, 2018.

S. Agapiou, S. Larsson, and A. M. Stuart. Posterior contraction rates for the Bayesian approach to linear ill-posed inverse problems. *Stoch. Proc. Appl.*, 123(10):3828–3860, 2013.

S. Agapiou, J. M. Bardsley, O. Papaspiliopoulos, and A. M. Stuart. Analysis of the Gibbs sampler for hierarchical inverse problems. *SIAM/ASA J. Uncertainty Quantification*, 2: 511–544, 2014.

S. Agapiou, O. Papaspiliopoulos, D. Sanz-Alonso, and A. M. Stuart. Importance sampling: intrinsic dimension and computational cost. *Stat. Sci.*, 32(3):405–431, 2017.

S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. Solving inverse problems using data-driven models. *Acta Numer.*, 28:1–174, 2019.

G. Bao, S. N. Chow, P. Li, and H. Zhou. Numerical solution of an inverse medium scattering problem with a stochastic source. *Inverse Probl.*, 26(7):074014, 2010.

G. Bao, P. Li, J. Lin, and F. Triki. Inverse scattering problems with multi-frequencies. *Inverse Probl.*, 31(9):093001, 2015.

J. O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer, New York, second edition, 1980.

A. Beskos, G. Roberts, A. Stuart, and J. Voss. MCMC method for diffusion bridges. *Stoch. Dynam.*, 08(03):319–350, 2008.

A. Beskos, A. Jasra, E. A. Muzaffer, and A. M. Stuart. Sequential Monte Carlo methods for Bayesian elliptic inverse problems. *Stat. Comput.*, 25:727–737, 2015.

K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart. Model reduction and neural networks for parametric PDEs. *SMAI J. Comput. Math.*, 7:121–157, 2021.

Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2006.

T. Bui-Thanh and Q. P. Nguyen. FEM-based discretization-invariant MCMC methods for PDE-constrained Bayesian inverse problems. *Inverse Probl. Imag.*, 10(4):943–975, 2016.

T. Bui-Thanh, O. Ghattas, J. Martin, and G. Stadler. A computational framework for infinite-dimensional Bayesian inverse problems part I: The linearized case, with application to global seismic inversion. *SIAM J. Sci. Comput.*, 35(6):A2494–A2523, 2013.

D. Calvetti, M. M. Dunlop, E. Somersalo, and A. M. Stuart. Iterative updating of model error for Bayesian inversion. *Inverse Probl.*, 34(2):025008, 2018.

T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE T. Neural. Network*, 6(4):904–917, 1995.

S. L. Cotter, M. Dashti, J. C. Robinson, and A. M. Stuart. Bayesian inverse problems for functions and applications to fluid mechanics. *Inverse Probl.*, 25(11):115008, 2009.

S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. MCMC methods for functions: modifying old algorithms to make them faster. *Stat. Sci.*, 28(3):424–446, 2013.

F. Cucker and S. Smale. On the mathematical foundations of learning. *B. Am. Math. Soc.*, 39(1):1–49, 2001.

T. Cui, K. J. H. Law, and Y. M. Marzouk. Dimension-independent likelihood-informed MCMC. *J. Comput. Phys.*, 304:109–137, 2016.

M. Dashti and A. M. Stuart. The Bayesian approach to inverse problems. *Handbook of Uncertainty Quantification*, pages 311–428, 2017.

Masoumeh Dashti, Stephen Harris, and Andrew Stuart. Besov priors for Bayesian inverse problems. *Inverse Probl. Imag.*, 6(2):183–200, 2012.

N. Davoudi, X. L. Deán-Ben, and D. Razansky. Deep learning optoacoustic tomography with sparse data. *Nat. Mach. Intell.*, 1:453–460, 2019.

M. M. Dunlop, M. A. Iglesias, and A. M. Stuart. Hierarchical Bayesian level set inversion. *Stat. Comput.*, 27:1555–1584, 2017.

M. M. Dunlop, D. Slepčcev, A. M. Stuart, and M. Thorpe. Large data and zero noise limits of graph-based semi-supervised learning algorithms. *Appl. Comput. Harmon. A.*, 49(2): 655–697, 2020.

H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer Academic Publishers, Netherlands, 1996.

L. C. Evans. *Partial Differential Equations*. American Mathematical Society, Providence, RI, second edition, 2010.

Z. Feng and J. Li. An adaptive independence sampler MCMC algorithm for Bayesian inferences of functions. *SIAM J. Sci. Comput.*, 40(3):A1310–A1321, 2018.

S. Ghosal and A. v. d. Vaart. *Fundamentals of Nonparametric Bayesian Inference*. Cambridge University Press, United States of America, 2017.

M. Giordano and R. Nickl. Consistency of Bayesian inference with Gaussian process priors in an elliptic inverse problem. *Inverse Probl.*, 36(8):085001, 2020.

E. Haber and L. Tenorio. Learning regularization functionals–a supervised training approach. *Inverse Probl.*, 19(3):611–626, 2003.

N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

F. Hoffmann, B. Hosseini, Z. Ren, and A. M. Stuart. Consistency of semi-supervised learning algorithms on graphs: probit and one-hot methods. *J. Mach. Learn. Res.*, 21(186):1–55, 2020.

Bamdad Hosseini and Nilima Nigam. Well-posed Bayesian inverse problems: Priors with exponential tails. *SIAM/ASA J. Uncertainty Quantification*, 5(1):436–465, 2017.

J. Jia, J. Peng, and J. Gao. Bayesian approach to inverse problems for functions with a variable index Besov prior. *Inverse Probl.*, 32(8):085006, 2016.

J. Jia, S. Yue, J. Peng, and J. Gao. Infinite-dimensional Bayesian approach for inverse scattering problems of a fractional Helmholtz equation. *J. Funct. Anal.*, 275(9):2299–2332, 2018.

J. Jia, B. Wu, J. Peng, and J. Gao. Recursive linearization method for inverse medium scattering problems with complex mixture Gaussian error learning. *Inverse Probl.*, 35(7): 075003, 2019.

J. Jia, J. Peng, and J. Gao. Posterior contraction for empirical Bayesian approach to inverse problems under non-diagonal assumption. *Inverse Probl. Imag.*, 15(2):201–228, 2021a.

J. Jia, Q. Zhao, Z. Xu, D. Meng, and Y. Leung. Variational Bayes' method for functions with applications to some inverse problems. *SIAM J. Sci. Comput.*, 43(1):A355–A383, 2021b.

J. Jia, P. Li, and D. Meng. Stein variational gradient descent on infinite-dimensional space and applications to statistical inverse problems. *SIAM J. Numer. Anal.*, 60(4):2225–2252, 2022.

B. Jin and J. Zou. Hierarchical Bayesian inference for ill-posed problems via variational method. *J. Comput. Phys.*, 229(19):7317–7343, 2010.

J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Springer-Verlag, New York, 2005.

J. P. Kaipio, T. Huttunen, T. Luostari, T. Lähivaara, and P. B. Monk. A Bayesian approach to improving the born approximation for inverse scattering with high-contrast materials. *Inverse Problems*, 35(8):084001, 2019.

H. Kekkonen, M. Lassas, and S. Siltanen. Posterior consistency and convergence rates for Bayesian inversion with hypoelliptic operators. *Inverse Probl.*, 32(8):085005, 2016.

Y. Khoo and L. Ying. Switchnet: a neural network model for forward and inverse scattering problems. *SIAM J. Sci. Comput.*, (5):A3182–A3201, 2019.

D. P. Kingma. *Variational inference & deep learning: A new synthesis*. PhD thesis, University of Amsterdam, 2017.

B. T. Knapik, A. van Der Vaart, and J. H. van Zanten. Bayesian inverse problems with Gaussian priors. *Ann. Statist.*, 39:2626–2657, 2011.

N. Kovachki, S. Lanthaler, and S. Mishra. On universal approximation and error bounds for Fourier neural operators. *J. Mach. Learn. Res.*, 22:1–76, 2021.

M. Lassas and S. Siltanen. Can one use total variation prior for edge-preserving Bayesian inversion? *Inverse Probl.*, 20(5):1537, 2004.

Matti Lassas, Eero Saksman, and Samuli Siltanen. Discretization-invariant Bayesian inversion and besov space priors. *Inverse Probl. Imag.*, 3(1):87–122, 2009.

J. Latz. On the well-posedness of Bayesian inverse problems. *SIAM/ASA J. Uncertainty Quantification*, 8(1):451–482, 2020.

J. Li and Y. M. Marzouk. Adaptive constuction of surrogates for the Bayesian solution of inverse problems. *SIAM J. Sci. Comput.*, 36(3):A1410–A1435, 2014.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar. Neural operator: graph kernel network for partial differential equations. arXiv:2003.03485, 2020a.

Z. Li, N. B. Kovachki, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *ICLR*, 2020b.

A. Lischke, G. Pang, M. Gulian, F. Song, C. Glusa, X. Zheng, Z. Mao, W. Cai, M. M. Meerschaert, M. Ainsworth, and G. E. Karniadakis. What is the fractional Laplacian? A comparative review with new results. *J. Comput. Phys.*, 404(1):109009, 2020.

A Logg, K. A. Mardal, and G. N. Wells. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

N. H. Nelsen and A. M. Stuart. The random feature model for input-output maps between Banach spaces. *SIAM J. Sci. Comput.*, 243:A3212–A3243, 2021.

N. Petra, J. Martin, G. Stadler, and O. Ghattas. A computational framework for infinite-dimensional Bayesian inverse problems, part II: Stochastic Newton MCMC with application to ice sheet flow inverse problems. *SIAM J. Sci. Comput.*, 36(4):A1525–A1555, 2014.

F. J. Pinski, G. Simpson, A. M. Stuart, and H. Weber. Kullback-Leibler approximation for probability measures on infinite dimensional space. *SIAM J. Math. Anal.*, 47(6): 4091–4122, 2015a.

F. J. Pinski, G. Simpson, A. M. Stuart, and H. Weber. Algorithms for Kullback-Leibler approximation of probability measures in infinite dimensions. *SIAM J. Sci. Comput.*, 37 (6):A2733–A2757, 2015b.

G. D. Prato. *An Introduction to Infinite-Dimensional Analysis*. Springer-Verlag, Berlin Heidelberg, 2006.

G. D. Prato and J. Zabczyk. *Stochastic Equations in Infinite Dimensions*. Cambridge University Press, Cambridge, second edition, 2014.

A. M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numer.*, 19:451–559, 2010.

S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional variational Bayesian neural networks. In *ICLR*, 2019.

B. Szabó, A. W. van der Vaart, and J. H. van Zanten. Frequentist coverage of adaptive nonparametric Bayesian credible sets. *Ann. Statist.*, 43(4):1391–1428, 2015.

N. G. Trillos, Z. Kaplan, T. Samakhoana, and D. Sanz-Alonso. On the consistency of graph-based Bayesian semi-supervised learning and the scalability of sampling algorithms. *J. Mach. Learn. Res.*, 21(28):1–47, 2020.

U. Villa, N. Petra, and O. Ghattas. HIPPYlib: An Extensible Software Framework for Large-Scale Inverse Problems Governed by PDEs: Part I: Deterministic Inversion and Linearized Bayesian Inference. *ACM Trans. Math. Softw.*, 47(2):1–34, 2021.

Sebastian J. Vollmer. Posterior consistency for Bayesian inverse problems through stability and regression results. *Inverse Probl.*, 29(12):125011, 2013.

K. Wang, T. Bui-Thanh, and O. Ghattas. A randomized maximum a posteriori method for posterior sampling of high dimensional nonlinear Bayesian inverse problems. *SIAM J. Sci. Comput.*, 40(1):A142–A171, 2018.

Z. Wang, T. Ren, J. Zhu, and B. Zhang. Function space particle optimization for Bayesian neural networks. In *ICLR*, 2019.

J. Yoo, A. Wahab, and J. C. Ye. A mathematical framework for deep learning in elastic source imaging. *SIAM J. Appl. Math.*, 78(5):2791–2818, 2018.

Z. Yue, H. Yong, Q. Zhao, L. Zhang, and D. Meng. Variational denoising network: toward blind noise modeling and removal. In *NeurIPS*, 2019.

Z. Yue, H. Yong, Q. Zhao, L. Zhang, and D. Meng. Variational image restoration network. *ArXiv*, 2008.10796, 2020.

C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt. Advances in variational inference. *IEEE T. Pattern Anal.*, 41(8):2008–2026, 2018.

K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. *IEEE T. Image Process.*, 26(7):3142–3155, 2017.

Q. Zhou, T. Yu, X. Zhang, and J. Li. Bayesian inference and uncertainty quantification for medical image reconstruction with poisson data. *SIAM J. Imaging Sci.*, 13(1):29–52, 2020.