

Accurate, scalable, and efficient Bayesian optimal experimental design with derivative-informed neural operators

Jinwoo Go^a, Peng Chen^a

^a*School of Computational Science and Engineering, Georgia Institute of Technology, 30332 GA, Atlanta, USA*

Abstract

We consider optimal experimental design (OED) problems in selecting the most informative observation sensors to estimate model parameters in a Bayesian framework. Such problems are computationally prohibitive when the parameter-to-observable (PtO) map is expensive to evaluate, the parameters are high-dimensional, and the optimization for sensor selection is combinatorial and high-dimensional. To address these challenges, we develop an accurate, scalable, and efficient computational framework based on derivative-informed neural operators (DINOs). The derivative of the PtO map is essential for accurate evaluation of the optimality criteria of OED in our consideration. We take the key advantage of DINOs, a class of neural operators trained with derivative information, to achieve high approximate accuracy of not only the PtO map but also, more importantly, its derivative. Moreover, we develop scalable and efficient computation of the optimality criteria based on DINOs and propose a modified swapping greedy algorithm for its optimization. We demonstrate that the proposed method is scalable to preserve the accuracy for increasing parameter dimensions and achieves high computational efficiency, with an over $1000\times$ speedup accounting for both offline construction and online evaluation costs, compared to high-fidelity Bayesian OED solutions for a three-dimensional nonlinear convection-diffusion-reaction example with tens of thousands of parameters.

Keywords: Optimal experimental design, Bayesian inverse problem, Derivative-informed neural operators, Dimension reduction, Uncertainty quantification, Greedy algorithm

1. Introduction

Mathematical modeling and computational simulation become increasingly important for understanding and optimizing complex systems in many scientific and engineering fields. In practical applications, various uncertainties arise in the modeling and simulation either because of the lack of knowledge of the systems (epistemic uncertainty) or the inherent randomness in the systems (aleatoric uncertainty) that cannot be reduced. To reduce the epistemic uncertainty commonly represented as uncertain model parameters, it is essential to acquire knowledge or data of the systems by proper experimental design and calibrate the model parameters by either deterministic or statistical/Bayesian inference from the experimental or observational data. However, these experiments may be very expensive to conduct or take an enormous time or space. This is where optimal experimental design (OED) comes into play. OED designs experiments to acquire the most informative data under budget/time/space constraints [1, 2, 3, 4, 5].

In this work, we consider OED for Bayesian inverse problems constrained by large-scale models described by partial differential equations (PDEs) under infinite/high-dimensional uncertain parameters, see [6, 3, 7, 8, 9, 5, 10, 11] and references therein. For the design optimality criteria, we consider both variance-based alphabetic optimality [3, 7, 8], and information theory-based optimality criterion known as mutual information or expected information gain (EIG) [9, 5, 11]. The variance-based optimality criteria seek

Email addresses: jgo31@gatech.edu (Jinwoo Go), pchen402@gatech.edu (Peng Chen)

to minimize the uncertainty represented by summary statistics of the posterior covariance, such as A-optimality (the trace of posterior covariance) and D-optimality (the determinant of posterior covariance), while the information theory-based EIG seeks to maximize the average information gained from all possible realizations of the experimental data. Mathematically, the EIG is defined as the expectation of Kullback–Leibler divergence between the posterior and the prior distributions over the data or marginal likelihood distribution, which involves the integration with respect to both the posterior and data distributions.

Several challenges are faced in solving OED with these optimality criteria for PDE-constrained large-scale Bayesian inverse problems, including: (1) each evaluation of the optimality criteria requires the computation of a parameter-to-observable (PtO) map—the map from the uncertain parameters to the observational quantity of interest of the system—at many data and parameter samples; (2) each computation of the PtO map involves a solution of the model, which can be very expensive for large-scale PDE models; (3) the dimensions of the uncertainty parameter space and the experiment design space can be very high or infinite as considered in our work, which brings the curse-of-dimensionality, i.e., the computational/sampling complexity increase rapidly (exponentially) with respect to the dimensions, for many conventional methods; and (4) the combinatorial optimization with respect to the experimental design may be highly non-convex, as in the example of optimal placement of sensors from a large number of candidate sensor locations.

To address these challenges, various computational methods have been developed over the last decade, including (1) sparse polynomial chaos approximation of PtO maps [4, 12], (2) Laplace approximation of non-Gaussian posterior distributions [13, 9, 14, 15, 16], (3) low-rank approximation of prior-preconditioned Hessian of the data misfit term [3, 7, 13, 17, 18, 19], (4) reduced order models [20, 21, 22] and deep neural networks [11] that serve as surrogate models of the PDEs or PtO maps, (5) variational inference and neural estimation in a fast approximation of the EIG or mutual information [23, 24, 25, 26], and (6) efficient optimization methods based on gradient information [3, 13, 12], greedy algorithms [27, 28, 21, 22], and swapping greedy algorithms [10, 5, 11].

In this work, we consider the Laplace approximation of the posterior distribution [13, 9, 14, 15, 16] and the low-rank approximation of the prior-preconditioned Hessian of the data misfit term [3, 7, 13, 17, 18, 19]. Though achieving significant computational reduction, such approximations are still infeasible for large-scale PDE models with high-dimensional parameters. This is because (1) a Maximum-A-Posteriori (MAP) point needs to be computed in the Laplace approximation for each observed data and each given experimental design, which requires the solution of a large-scale PDE-constrained high-dimensional optimization problem; (2) the evaluation of the optimality criteria involves a low-rank factorization of the large-scale prior-preconditioned Hessian of the misfit at each MAP point; (3) both the MAP points and the low-rank factorizations need to be computed numerous times, e.g., millions of times for the combinatorial optimization of the high-dimensional experimental design by a greedy algorithm.

Contributions. We propose to address these computational challenges by developing an accurate, scalable, and efficient computational framework based on the derivative-informed neural operator (DINO) [29]. We make the key observation that not only the PtO map but also its derivative (Jacobian) with respect to the parameters have to be evaluated numerous times for the Laplace and low-rank approximations. This motivates us to build a fast surrogate of the PtO map that is accurate not only for the map but also for its derivative, which is the essential property of DINO that trains a neural network to reduce the errors of both the map and its derivative. To achieve the computational scalability of DINO with increasing parameter and observation dimensions, we employ dimension reduction techniques by projecting the input and output to proper low-dimensional subspaces. We identify the most informative input subspaces for the MAP point approximation and the Jacobian approximation as a derivative-informed input subspace (DIS). Thanks to the dimension reduction, we can considerably reduce the size of the neural network and use a relatively small number of training data of both the PtO map and its projected Jacobian to achieve high approximation accuracy. We also propose to compute the projected Jacobian data in the input and output subspaces instead of the full Jacobian, which only requires the solutions of a small number of the PDEs and its adjoint systems for each Jacobian sample. This effectively alleviates the computation and storage cost for the Jacobian data and only requires marginal cost compared to that for generating the PtO map data.

Using the DINO evaluation of the PtO map and its projected Jacobian, we further derive the formulations to (1) compute the MAP point by solving a low-dimensional optimization problem using automatic

differentiation with reliable derivatives, and (2) solve a small-scale eigenvalue problem in the input subspace for the low-rank approximation of the Hessian of the misfit. These computations enable efficient evaluations of the optimality criteria for each data and corresponding experimental design with any given number of sensors. To this end, we propose a modification of a swapping greedy algorithm developed in [5] to solve the Bayesian OED optimization problem by a different initialization of the sensors using a greedy algorithm. We demonstrate the high accuracy, scalability, and efficiency of our approach with two examples of two-dimensional (2D) and three-dimensional (3D) PDE models, and with hundreds of observables and tens of thousands of parameters. Specifically, we achieve $80\times$ speedup compared to the high-fidelity Bayesian OED solution for the 2D model and $1148\times$ speedup for the 3D model, including both online evaluation and offline construction costs of the surrogates.

Related work. Our work is more closely related to the recent work [5] in using Laplace and low-rank approximations and [11] in using surrogate models than other works listed above. We point out the significant difference that in [5] the Laplace approximation is not constructed at the MAP point but rather at the prior sample point because of the prohibitive computational cost. Moreover, the authors assumed that observation noise is uncorrelated to enable an offline-online decomposition of the low-rank approximation. In [11], a neural network surrogate was constructed to compute only the PtO map and the EIG by a double loop Monte Carlo (DLMC) sampling. The surrogate was not constructed using the additional Jacobian information, thus was not accurate and not used to compute the Jacobian and the related A-/D-optimality criteria as well as the Laplace approximation based EIG. It is limited to relatively large observation noise because of the pollution of the neural network approximation errors in the accurate computation of the likelihood function, which is crucial for the DLMC sampling. In contrast, our work is not subject to these limitations. However, we remark that our method is limited to the low-rankness or fast spectral decay of the PtO map and the Hessian of the data misfit term as in [5, 11]. The fast spectral decay is the intrinsic property of many high-dimensional problems and has been exploited in various other contexts besides Bayesian OED, e.g., Bayesian inference [30, 31, 32, 33, 34, 35, 36], model reduction for sampling and deep learning [37, 38, 39, 40], and optimization under uncertainty [41, 42, 43, 44].

The rest of the paper is organized as follows: We present Bayesian OED problems with three optimality criteria constrained by PDEs to infer infinite-dimensional parameters from noisy observations in Section 2. This is followed by Section 3 in solving Bayesian OED problems by high-fidelity approximation using finite element discretization, Laplace, and low-rank approximations of the posterior. Section 4 introduces DINO surrogates incorporating proper dimension reduction techniques, efficient computation of the optimality criteria, and a modified swapping greedy algorithm to accelerate the solution of the Bayesian OED problems. We then present the numerical results in Section 5 to demonstrate the accuracy, scalability, and efficiency of the proposed method in solving Bayesian OED problems constrained by large-scale PDEs. Finally, we conclude the paper in Section 6.

2. Problem statement

We present Bayesian inverse problems constrained by PDEs with infinite-dimensional parameters, Bayesian OED problems in the context of optimal sensor placement, and the challenges for their solutions.

2.1. Bayesian inverse problem

We consider Bayesian inverse problems to infer uncertain parameters in mathematical models described by PDEs written in an abstract form as

$$\mathcal{R}(u, m) = 0 \text{ in } \mathcal{V}', \quad (1)$$

where $u \in \mathcal{V}$ is the state variable in a separable Banach space \mathcal{V} defined in physical domain $D \in \mathbb{R}^d$, e.g., in dimension $d = 1, 2, 3$; m represents the uncertain parameter. Specifically, we consider m as a random field parameter in a Hilbert space \mathcal{M} defined in D , which is infinite-dimensional. The operator $\mathcal{R} : \mathcal{V} \times \mathcal{M} \rightarrow \mathcal{V}'$ represents the strong form of the PDE, where \mathcal{V}' is the dual of \mathcal{V} .

Let $\mathcal{B} : \mathcal{V} \rightarrow \mathbb{R}^d$ denote an observation operator that maps the state variable u to a vector of d_s -dimensional observation functionals, with $d_s \in \mathbb{N}$. We define a corresponding PtO map $\mathcal{F} : \mathcal{M} \rightarrow \mathbb{R}^{d_s}$ as $\mathcal{F}(m) := \mathcal{B}(u(m))$. In particular, we consider the observation made in d_s sensor locations $(\mathbf{x}_1, \dots, \mathbf{x}_{d_s})$ with $\mathbf{x}_i \in D$ indicating the i -th sensor location in domain D . In the OED problem, we can only choose $r_s \in \mathbb{N}$ out of the d_s candidate sensors because of the budget constraint, where typically $d_s \gg r_s$. We use a design matrix $\xi \in \mathbb{R}^{d_s \times r_s}$ to represent the selection of r_s sensors out of the d_s candidates, with $\xi_{ij} = 1$ if the j -th sensor is selected at the i -th location \mathbf{x}_i , and $\xi_{ij} = 0$ if not. Moreover, we consider that one location can not be placed with more than one sensor. Mathematically, we define the admissible design matrix set Ξ as

$$\Xi = \left\{ \xi \in \mathbb{R}^{d_s \times r_s} : \xi_{ij} \in \{0, 1\}, \sum_{i=1}^{d_s} \xi_{ij} = 1, \sum_{j=1}^{r_s} \xi_{ij} \in \{0, 1\} \right\}. \quad (2)$$

For a given design matrix $\xi \in \Xi$, we denote the corresponding PtO map as $\mathcal{F}_\xi = \xi \mathcal{F} : \mathcal{M} \rightarrow \mathbb{R}^{r_s}$. We consider that the noisy observation data \mathbf{y} is corrupted with an additive observation noise ϵ , given as

$$\mathbf{y} = \mathcal{F}_\xi(m) + \epsilon, \quad (3)$$

where we assume the observation noise obeys Gaussian distribution $\epsilon \sim \mathcal{N}(\mathbf{0}, \Gamma_{\text{noise}})$ with zero mean and covariance $\Gamma_{\text{noise}} \in \mathbb{R}^{r_s \times r_s}$. Under this assumption, the likelihood of the data \mathbf{y} for a parameter m and design matrix ξ is given by

$$\pi_{\text{like}}(\mathbf{y}|m, \xi) = \frac{1}{\sqrt{(2\pi)^{r_s} |\Gamma_{\text{noise}}|}} \exp(-\Phi(\mathbf{y}, m, \xi)), \quad (4)$$

where $|\Gamma_{\text{noise}}|$ denotes the determinant of Γ_{noise} , and the potential function $\Phi(m, \mathbf{y}, \xi)$ represents a data-model misfit term given as

$$\Phi(\mathbf{y}, m, \xi) = \frac{1}{2} \|\mathbf{y} - \mathcal{F}_\xi(m)\|_{\Gamma_{\text{noise}}^{-1}}^2 = \frac{1}{2} (\mathbf{y} - \mathcal{F}_\xi(m))^T \Gamma_{\text{noise}}^{-1} (\mathbf{y} - \mathcal{F}_\xi(m)). \quad (5)$$

Given data \mathbf{y} in (3), the inverse problem is to infer the parameter m constrained by the model (1). In a Bayesian framework, the inference is to update the probability distribution of the parameter m from its prior distribution μ_{prior} to the posterior distribution μ_{post} conditioned on the observation data \mathbf{y} and design matrix ξ . We consider a Gaussian prior $\mu_{\text{prior}} = \mathcal{N}(m_{\text{prior}}, \mathcal{C}_{\text{prior}})$ with mean m_{prior} and covariance operator $\mathcal{C}_{\text{prior}} : \mathcal{M} \rightarrow \mathcal{M}'$. In particular, we use a common choice of Matérn covariance operator $\mathcal{C}_{\text{prior}} = \mathcal{A}^{-\alpha}$ [45], where $\mathcal{A} = -\gamma \Delta + \delta I$ with homogeneous Neumann boundary condition, Δ is the Laplacian, I denotes the identity, and the constants $\alpha, \gamma, \delta > 0$ collectively determine the smoothness, correlation, and variance of the parameter. We require $\alpha > d/2$ for the covariance to be of trace class [46], where d is the dimension of the domain D . Given this prior, Bayes' rule states that the posterior of the parameter written in the form of Radon–Nikodym derivative satisfies

$$\frac{d\mu_{\text{post}}^{\mathbf{y}, \xi}(m)}{d\mu_{\text{prior}}(m)} = \frac{1}{\pi(\mathbf{y}|\xi)} \pi_{\text{like}}(\mathbf{y}|m, \xi), \quad (6)$$

where $\pi(\mathbf{y}|\xi)$ is known as model evidence or marginal likelihood distribution given by

$$\pi(\mathbf{y}|\xi) = \int_{\mathcal{M}} \pi_{\text{like}}(\mathbf{y}|m, \xi) d\mu_{\text{prior}}(m), \quad (7)$$

which is typically computationally intractable in infinite or high dimensions.

2.2. Bayesian optimal experimental design

The Bayesian OED problem in our context seeks the optimal placement of r_s sensors out of d_s candidate locations that provide maximum information to infer the parameter. For this task, we consider several widely

used optimality criteria, including the variance-based A-optimality criterion and D-optimality criterion, and the information theory-based EIG or mutual information.

The variance-based A-/D-optimality criteria are defined for a Laplace approximation of the posterior distribution given by $\hat{\mu}_{\text{post}} = \mathcal{N}(m_{\text{MAP}}, \mathcal{C}_{\text{post}})$. Here, m_{MAP} denotes the MAP point defined as

$$m_{\text{MAP}}(\mathbf{y}, \xi) := \arg \min_{m \in \mathcal{M}} \frac{1}{2} \|\mathbf{y} - \mathcal{F}_{\xi}(m)\|_{\Gamma_{\text{noise}}^{-1}}^2 + \frac{1}{2} \|m - m_{\text{prior}}\|_{\mathcal{C}_{\text{prior}}^{-1}}^2, \quad (8)$$

and the covariance operator $\mathcal{C}_{\text{post}}$ is given by

$$\mathcal{C}_{\text{post}} = (\mathcal{H}_{\text{misfit}}(m_{\text{MAP}}) + \mathcal{C}_{\text{prior}}^{-1})^{-1}, \quad (9)$$

where $\mathcal{H}_{\text{misfit}}(m_{\text{MAP}})$ is Hessian of data-model misfit term $\Phi(m, \mathbf{y}, \xi)$ with respect to m evaluated at m_{MAP} . A common choice for $\mathcal{H}_{\text{misfit}}(m_{\text{MAP}})$ in infinite-/high-dimensional Bayesian OED is to use its Gauss–Newton approximation

$$\mathcal{H}_{\text{misfit}}(m_{\text{MAP}}) \approx \mathcal{H}_{\text{misfit}}^{\text{GN}}(m_{\text{MAP}}) := \nabla_m \mathcal{F}_{\xi}(m_{\text{MAP}})^T \Gamma_{\text{noise}}^{-1} \nabla_m \mathcal{F}_{\xi}(m_{\text{MAP}}), \quad (10)$$

with the Jacobian $\nabla_m \mathcal{F}_{\xi}(m_{\text{MAP}}) : \mathcal{M} \rightarrow \mathbb{R}^{r_s}$ evaluated at the MAP point m_{MAP} . This corresponds to a Fisher information matrix-based optimality criterion [47]. The Gauss–Newton approximation of the Hessian is exact if the PtO map $\mathcal{F}_{\xi}(m)$ is linear with respect to m , or it can be considered as a linear approximation of $\mathcal{F}_{\xi}(m)$ at the MAP point, i.e.,

$$\mathcal{F}_{\xi}(m) \approx \mathcal{F}_{\xi}(m_{\text{MAP}}) + \nabla_m \mathcal{F}_{\xi}(m_{\text{MAP}})(m - m_{\text{MAP}}). \quad (11)$$

We note that for the A-/D-optimality criteria, we need to obtain the MAP point m_{MAP} by solving the optimization problem (8) and compute the Jacobian $\nabla_m \mathcal{F}_{\xi}(m_{\text{MAP}})$ at the MAP point, both of which are constrained by the PDE (1) and depend on the realization of the data \mathbf{y} and the design matrix ξ .

To this end, we can define the Bayesian OED problem with the A-optimality criterion as

$$\xi_A^* = \arg \min_{\xi \in \Xi} \mathbb{E}_{\pi(\mathbf{y}|\xi)} [\text{trace}(\mathcal{C}_{\text{post}}^{\mathbf{y}, \xi}(m_{\text{MAP}}^{\mathbf{y}, \xi}))], \quad (12)$$

and D-optimality criterion as

$$\xi_D^* = \arg \min_{\xi \in \Xi} \mathbb{E}_{\pi(\mathbf{y}|\xi)} [\det(\mathcal{C}_{\text{prior}}^{-1} \mathcal{C}_{\text{post}}^{\mathbf{y}, \xi}(m_{\text{MAP}}^{\mathbf{y}, \xi}))], \quad (13)$$

where $\text{trace}(\cdot)$ and $\det(\cdot)$ denote the trace and determinant. The covariance operator $\mathcal{C}_{\text{post}}^{\mathbf{y}, \xi}(m_{\text{MAP}}^{\mathbf{y}, \xi})$ is defined in (9), with the superscript representing the dependence on the data \mathbf{y} observed at the sensors determined by the design matrix ξ . The expectation $\mathbb{E}_{\pi(\mathbf{y}|\xi)}[\cdot]$ is taken with respect to the distribution of the data \mathbf{y} conditioned on ξ . This expectation can be evaluated by the double integral $\mathbb{E}_{m \sim \mu_{\text{prior}}} \mathbb{E}_{\mathbf{y} \sim \pi_{\text{like}}(\cdot|m, \xi)}[\cdot]$.

Alternatively, the Bayesian OED with the information theory-based EIG seeks to maximize the expected information gained from the sensors, i.e.,

$$\xi_{\text{EIG}}^* = \arg \max_{\xi \in \Xi} \mathbb{E}_{\pi(\mathbf{y}|\xi)} \left[D_{\text{KL}}(\mu_{\text{post}}^{\mathbf{y}, \xi} || \mu_{\text{prior}}) \right], \quad (14)$$

where the information gain is measured by the Kullback–Leibler (KL) divergence between the posterior distribution $\mu_{\text{post}}^{\mathbf{y}, \xi}$ and the prior distribution μ_{prior} , defined as

$$D_{\text{KL}}(\mu_{\text{post}}^{\mathbf{y}, \xi} || \mu_{\text{prior}}) = \mathbb{E}_{m \sim \mu_{\text{post}}^{\mathbf{y}, \xi}} \left[\log \left(\frac{d\mu_{\text{post}}^{\mathbf{y}, \xi}(m)}{d\mu_{\text{prior}}(m)} \right) \right]. \quad (15)$$

Under the assumption of Gaussian prior $\mu_{\text{prior}} = \mathcal{N}(m_{\text{prior}}, \mathcal{C}_{\text{prior}})$ and using the Laplace approximation of the posterior $\hat{\mu}_{\text{prior}}^{\mathbf{y}, \xi} = \mathcal{N}(m_{\text{MAP}}^{\mathbf{y}, \xi}, \mathcal{C}_{\text{post}}^{\mathbf{y}, \xi})$, the analytical form of the KL divergence is given by [7]

$$D_{\text{KL}}(\hat{\mu}_{\text{prior}}^{\mathbf{y}, \xi} || \mu_{\text{prior}}) = \frac{1}{2} \log \det(I + \mathcal{C}_{\text{prior}} \mathcal{H}_{\text{misfit}}^{\mathbf{y}, \xi}) - \frac{1}{2} \text{trace}(\mathcal{H}_{\text{misfit}}^{\mathbf{y}, \xi} \mathcal{C}_{\text{post}}^{\mathbf{y}, \xi}) + \frac{1}{2} \|m_{\text{MAP}}^{\mathbf{y}, \xi} - m_{\text{prior}}\|_{\mathcal{C}_{\text{prior}}^{-1}}^2. \quad (16)$$

Several computational challenges arise in solving the Bayesian OED problems with these optimality criteria, including: (1) the dimension of the uncertain parameter space and the design space can be infinite or very high, which brings the curse of dimensionality, i.e., the computational/sampling complexity increases exponentially with respect to the dimensions; (2) each evaluation of the optimality criteria (trace (12), determinant (13), EIG (14)) requires numerous expensive PDE solves for the computation of the PtO map \mathcal{F}_ξ and its Jacobian $\nabla_m \mathcal{F}_\xi$; (3) the optimization with respect to experimental design ξ is combinatorial and high-dimensional, which may require a large number of evaluations of the optimality criteria.

3. High-fidelity approximations

In this section, we follow [13, 5, 45, 48] and present high-fidelity approximations using a finite element method to discretize the infinite-dimensional parameter field and solve the Bayesian OED problem based on Laplace and low-rank approximations. We highlight the specific computational challenges of high-fidelity approximations as the motivation and basics to develop our computational framework in Section 4. We also take the high-fidelity approximations of all the related quantities as the reference to demonstrate the accuracy, scalability, and efficiency of our proposed method in Section 5.

3.1. High-fidelity discretization

The random field parameter m lives in the infinite-dimensional Hilbert space \mathcal{M} . We discretize m by a finite element method in a finite-dimensional space $\mathcal{M}_{d_m} \subset \mathcal{M}$ of dimension d_m with piecewise continuous Lagrange polynomial basis $\{\phi_j\}_{j=1}^{d_m}$ such that $\phi_j(x_i) = \delta_{ij}$. Let h denote the mesh size of the discretization, we approximate the parameter by $m_h \in \mathcal{M}_{d_m}$ defined as

$$m_h(x) = \sum_{j=1}^{d_m} m_j \phi_j(x), \quad (17)$$

where $\mathbf{m} = (m_1, m_2, \dots, m_{d_m})^T \in \mathbb{R}^{d_m}$ is a coefficient vector of m_h . The prior distribution for this discretized parameter \mathbf{m} is Gaussian $\mu(\mathbf{m}) = \mathcal{N}(\mathbf{m}_{\text{prior}}, \Gamma_{\text{prior}})$ with mean vector $\mathbf{m}_{\text{prior}}$ and covariance matrix Γ_{prior} as discretized from the mean m_{prior} and the covariance operator $\mathcal{C}_{\text{prior}} = \mathcal{A}^{-\alpha}$ with $\mathcal{A} = -\gamma \Delta + \delta I$. Let M and A denote the finite element mass matrix and stiffness matrix given by

$$M_{ij} = \int_D \phi_i(x) \phi_j(x) dx, \quad i, j = 1, \dots, d_m, \quad (18)$$

and

$$A_{ij} = \int_D (-\gamma \nabla \phi_i(x) \cdot \nabla \phi_j(x) + \delta \phi_i(x) \phi_j(x)) dx, \quad i, j = 1, \dots, d_m,$$

then the inverse of the covariance matrix Γ_{prior} for $\alpha = 2$ is given by [45]

$$\Gamma_{\text{prior}}^{-1} = AM^{-1}A.$$

3.2. Laplace and low-rank approximations

By the high-dimensional discretization, we can write the Laplace approximation of the posterior distribution of \mathbf{m} as $\mathcal{N}(\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi}, \Gamma_{\text{post}}^{\mathbf{y}, \xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi}))$, where the MAP point $\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi}$ is obtained from solving the following finite/high-dimensional optimization problem corresponding to (8) in the function space

$$\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi} = \arg \min_{\mathbf{m} \in \mathbb{R}^{d_m}} \frac{1}{2} \|\mathbf{y} - F_\xi(\mathbf{m})\|_{\Gamma_{\text{noise}}^{-1}}^2 + \frac{1}{2} \|\mathbf{m} - \mathbf{m}_{\text{prior}}\|_{\Gamma_{\text{prior}}^{-1}}^2, \quad (19)$$

with F_ξ denoting a map from \mathbf{m} to the observables corresponding to a discrete version of the PtO map \mathcal{F}_ξ for a given sensor selection ξ . To solve this optimization problem, we can employ, e.g., an inexact Newton-CG algorithm, which requires the computation of the derivative of $F_\xi(\mathbf{m})$ with respect to \mathbf{m} .

The posterior covariance matrix $\Gamma_{\text{post}}^{\mathbf{y},\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi})$ evaluated at $\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}$ is given by

$$\Gamma_{\text{post}}^{\mathbf{y},\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}) = (H_{\text{misfit}}^{\mathbf{y},\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}) + \Gamma_{\text{prior}}^{-1})^{-1}, \quad (20)$$

where $H_{\text{misfit}}^{\mathbf{y},\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi})$ is a discrete approximation of $\mathcal{H}_{\text{misfit}}^{\mathbf{y},\xi}(m_{\text{MAP}})$ in (10) given by

$$H_{\text{misfit}}^{\mathbf{y},\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}) \approx H_{\text{misfit}}^{\text{GN}}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}) = (\nabla_{\mathbf{m}} F_{\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}))^T \Gamma_{\text{noise}}^{-1} \nabla_{\mathbf{m}} F_{\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}). \quad (21)$$

The posterior covariance matrix Γ_{post} is high-dimensional and involves the Jacobian $\nabla_{\mathbf{m}} F_{\xi}$ and its transpose which requires solving the PDE and its adjoint. To efficiently compute the posterior covariance, we employ a low-rank decomposition by first solving a generalized eigenvalue problem

$$H_{\text{misfit}}^{\mathbf{y},\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}) \mathbf{w}_i = \lambda_i \Gamma_{\text{prior}}^{-1} \mathbf{w}_i, \quad i = 1, \dots, r, \quad (22)$$

with eigenpairs $(\lambda_i, \mathbf{w}_i)$, $i = 1, \dots, r$, corresponding to the largest r eigenvalues $\lambda_1 \geq \dots \geq \lambda_r$, and $\mathbf{w}_i^T \Gamma_{\text{prior}}^{-1} \mathbf{w}_j = \delta_{ij}$, $i, j = 1, \dots, r$. It requires $O(r)$ linearized PDE solves using a randomized SVD algorithm [45], see [5] for more details. Let $\Lambda_r = \text{diag}(\lambda_1, \dots, \lambda_r)$, $W_r = (\mathbf{w}_1, \dots, \mathbf{w}_r)$, and $V_r = \Gamma_{\text{prior}}^{-1} W_r$, we have $W_r^T V_r = I$ and obtain a low-rank approximation from (22) as in [48]

$$\Gamma_{\text{prior}} H_{\text{misfit}}^{\mathbf{y},\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}) \approx W_r \Lambda_r V_r^T, \quad (23)$$

which is accurate for a relatively small r as long as the eigenvalues decay fast with $\lambda_{r+1} \ll 1$. To this end, the posterior covariance can be approximated by [48]

$$\begin{aligned} \Gamma_{\text{post}}^{\mathbf{y},\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}) &= (\Gamma_{\text{prior}} H_{\text{misfit}}^{\mathbf{y},\xi}(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}) + I)^{-1} \Gamma_{\text{prior}} \\ &\approx (W_r \Lambda_r V_r^T + I)^{-1} \Gamma_{\text{prior}} \\ &= (I - W_r D_r V_r^T) \Gamma_{\text{prior}} \\ &= \Gamma_{\text{prior}} - W_r D_r W_r^T, \end{aligned} \quad (24)$$

where we factorized Γ_{prior} in the first equality, used low-rank approximation (23) in the inequality, employed Sherman–Morrison–Woodbury formula for the second equality with $D_r = \text{diag}(\lambda_1/(1+\lambda_1), \dots, \lambda_r/(1+\lambda_r))$, and used the definition of V_r such that $V_r^T \Gamma_{\text{prior}} = W_r^T$ in the last equality.

3.3. High-fidelity Bayesian OED

By applying the Laplace and low-rank approximations of the posterior, we can compute the approximate A-optimality optimal design corresponding to (12) as

$$\xi_A^* \approx \arg \min_{\xi} \mathbb{E}_{\pi(\mathbf{y}|\xi)} [\text{trace}(\Gamma_{\text{prior}} - W_r D_r W_r^T)] = \arg \max_{\xi} \mathbb{E}_{\pi(\mathbf{y}|\xi)} [\text{trace}(W_r D_r W_r^T)], \quad (25)$$

where the equality is due to that Γ_{prior} does not depend on \mathbf{y} and ξ .

Similarly, we can compute the approximate D-optimality optimal design corresponding to (13) as

$$\xi_D^* \approx \arg \min_{\xi} \mathbb{E}_{\pi(\mathbf{y}|\xi)} [\det((W_r \Lambda_r V_r^T + I)^{-1})] = \arg \max_{\xi} \mathbb{E}_{\pi(\mathbf{y}|\xi)} \left[\prod_{i=1}^r (1 + \lambda_i) \right]. \quad (26)$$

For the EIG criteria defined in (14) with the Laplace approximation and (16), we can compute the approximation with the following form [5]

$$\xi_{\text{EIG}}^* \approx \arg \max_{\xi} \mathbb{E}_{\pi(\mathbf{y}|\xi)} \left[\sum_{i=1}^r \log(1 + \lambda_i) - \sum_{i=1}^r \frac{\lambda_i}{1 + \lambda_i} + \|\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi} - \mathbf{m}_{\text{prior}}\|_{\Gamma_{\text{prior}}^{-1}}^2 \right]. \quad (27)$$

3.4. Computational challenges

We remark that both the eigenpairs (λ_i, w_i) , $i = 1, \dots, r$, and the MAP point $\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi}$ depend on the data sample \mathbf{y} and the design ξ . The conditional expectation $\mathbb{E}_{\pi(\mathbf{y}|\xi)}$ in the above three optimal design problems can be computed using sample average approximation (SAA) by drawing samples $\mathbf{y}^{(n)} = F_\xi(\mathbf{m}^{(n)}) + \boldsymbol{\epsilon}^{(n)}$, $n = 1, \dots, N_s$, with $\mathbf{m}^{(n)}$ drawn from the prior distribution of \mathbf{m} and $\boldsymbol{\epsilon}^{(n)}$ drawn from the observation noise distribution. Then we need to solve the optimization problem (19) for the MAP point and the generalized eigenvalue problem (22) for the eigenpairs at each of the N_s samples, which becomes computationally prohibitive for large N_s and large-scale PDEs to solve. We point out that the authors of [5] replace the MAP point $\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi}$ by the prior sample $\mathbf{m}^{(n)}$, which avoids solving the optimization problem (19). Moreover, they assume uncorrelated observation noise $\boldsymbol{\epsilon}^{(n)}$ in each observation dimension, by which they only need to solve a generalized eigenvalue problem in high dimensions once and then solve N_s eigenvalue problems of size $r \times r$ for each design ξ in the design optimization. However, the MAP point may be rather different from the prior sample, especially for a small number of observables, which can make their method potentially less effective. The computational method in solving the generalized eigenvalue problem also becomes not applicable for correlated observation noise. In this work, we propose to efficiently compute both the MAP point and the eigenpairs by using derivative-enhanced surrogate models, which completely avoids solving the PDE models during the optimization for the optimal design once the surrogate models are constructed. Moreover, we do not need the observation noise (as required in [5]) to be uncorrelated to use the surrogate models.

4. Bayesian OED with DINO

In this section, we present our approach for scalable and efficient computation of the optimality criteria based on DINO [29], and introduce a swapping greedy algorithm modified from [5] to optimize the experimental design. To achieve the scalability of the DINO approximation with accurate derivative evaluation, we employ derivative-informed dimension reduction methods [40, 49] to reduce the dimensions of both the input parameters and output observables. For the efficient computation of the optimality criteria, we formulate both the optimization of the MAP point problem (19) and the generalized eigenvalue problem (22) in the derivative-informed subspaces. Additionally, we present a complexity analysis of the high-fidelity and DINO approximations in terms of the number of PDE solves for the full Bayesian OED optimization.

4.1. Derivative-informed dimension reduction

The random field parameter m defined in the function space \mathcal{M} is infinite-dimensional. By the high-fidelity discretization of m in Section 3.1, the dimension $d_{\mathbf{m}}$ of discrete parameters \mathbf{m} is often very high. Meanwhile, the dimension d_F of candidate sensors or observables F is also set high to select the most informative sensors. The high dimensionality of input parameters and output observables makes it challenging to build an accurate neural network surrogate of the PtO map without a sufficiently large number of training data and a big training cost. To address this challenge, we present derivative-informed dimension reduction and a commonly used principal component analysis (PCA) for comparison.

In both the computation of the MAP point by a Newton-CG optimizer to solve (19) and the evaluation of the Gauss-Newton Hessian in (21), we need to compute the derivative $\nabla_{\mathbf{m}} F(\mathbf{m})$. This motivates a derivative-informed dimension reduction for the input parameter \mathbf{m} . Specifically, let $\mathbf{m}(\boldsymbol{\eta}) = \mathbf{m}_{\text{prior}} + \Gamma_{\text{prior}}^{1/2} \boldsymbol{\eta}$ with the whitened noise $\boldsymbol{\eta} \sim \mathcal{N}(0, I)$. We can formally write the eigenvalue decomposition for input dimension reduction as

$$\mathbb{E}_{\boldsymbol{\eta}} [\nabla_{\boldsymbol{\eta}} F^T(\mathbf{m}(\boldsymbol{\eta})) \nabla_{\boldsymbol{\eta}} F(\mathbf{m}(\boldsymbol{\eta}))] \boldsymbol{\varphi}^{(i)} = \lambda_i \boldsymbol{\varphi}^{(i)}, \quad i = 1, \dots, r, \quad (28)$$

with $\lambda_1 \geq \dots \geq \lambda_r$ and $(\boldsymbol{\varphi}^{(i)})^T \boldsymbol{\varphi}^{(j)} = \delta_{ij}$, which are equivalent to compute [40, 49]

$$\mathbb{E}_{\mathbf{m}} [\nabla_{\mathbf{m}} F^T(\mathbf{m}) \nabla_{\mathbf{m}} F(\mathbf{m})] \boldsymbol{\psi}^{(i)} = \lambda_i \Gamma_{\text{prior}}^{-1} \boldsymbol{\psi}^{(i)}, \quad i = 1, \dots, r, \quad (29)$$

with $\boldsymbol{\psi}^{(i)} = \Gamma_{\text{prior}}^{1/2} \boldsymbol{\varphi}^{(i)}$ and $(\boldsymbol{\psi}^{(i)})^T \Gamma_{\text{prior}}^{-1} \boldsymbol{\psi}^{(j)} = \delta_{ij}$. The expectation can be computed by SAA. To this end, the input parameter dimension reduction can be computed by the linear projection

$$\mathbf{m} \approx \mathbf{m}_r := \mathbf{m}_{\text{prior}} + \sum_{i=1}^r \beta_i \boldsymbol{\psi}^{(i)} \quad \text{where} \quad \beta_i = (\mathbf{m} - \mathbf{m}_{\text{prior}})^T \Gamma_{\text{prior}}^{-1} \boldsymbol{\psi}^{(i)}. \quad (30)$$

We also consider a derivative-informed output dimension reduction by writing

$$\mathbb{E}_{\boldsymbol{\eta}} [\nabla_{\boldsymbol{\eta}} F(\mathbf{m}(\boldsymbol{\eta})) \nabla_{\boldsymbol{\eta}} F^T(\mathbf{m}(\boldsymbol{\eta}))] \boldsymbol{\psi}^{(i)} = \lambda_i \boldsymbol{\psi}^{(i)}, \quad i = 1, \dots, r, \quad (31)$$

with $\lambda_1 \geq \dots \geq \lambda_r$ and $(\boldsymbol{\psi}^{(i)})^T \boldsymbol{\psi}^{(j)} = \delta_{ij}$, which are equivalent to compute

$$\mathbb{E}_{\mathbf{m}} [\nabla_{\mathbf{m}} F(\mathbf{m}) \Gamma_{\text{prior}} \nabla_{\mathbf{m}} F^T(\mathbf{m})] \boldsymbol{\psi}^{(i)} = \lambda_i \boldsymbol{\psi}^{(i)}, \quad i = 1, \dots, r, \quad (32)$$

where the expectation is computed by SAA. Then the projected output observables can be computed by

$$F \approx F_r := \bar{F} + \sum_{i=1}^r \beta_i \boldsymbol{\psi}^{(i)} \quad \text{where} \quad \beta_i = (F - \bar{F})^T \boldsymbol{\psi}^{(i)}, \quad (33)$$

where \bar{F} is a the sample mean of F . To distinguish the eigenvectors for the input and output dimension reduction, we use $\boldsymbol{\psi}_{\mathbf{m}}^{(i)}$ for \mathbf{m} and $\boldsymbol{\psi}_F^{(i)}$ for F . We also name $\Psi_{\mathbf{m}} = (\boldsymbol{\psi}_{\mathbf{m}}^{(1)}, \dots, \boldsymbol{\psi}_{\mathbf{m}}^{(r)})$ and $\Psi_F = (\boldsymbol{\psi}_F^{(1)}, \dots, \boldsymbol{\psi}_F^{(r)})$ as the derivative-informed input subspace (DIS) basis and derivative-informed output subspace (DOS) basis.

For a comparison of the derivative-informed dimension reduction, we briefly present a commonly used PCA dimension reduction. Given N samples of the input-output pairs $(\mathbf{m}^{(n)}, F^{(n)})$, $n = 1, \dots, N$, where $F^{(n)}$ is computed by solving the PDE model at $\mathbf{m}^{(n)}$, we can perform dimension reduction of them by PCA or equivalently a truncated singular value decomposition (tSVD). Let matrix $B = (F^{(1)} - \bar{F}, \dots, F^{(N)} - \bar{F})$ with sample mean \bar{F} . The tSVD of B , which can be computed by a randomized SVD algorithm, is given by

$$B \approx B_r := \Psi_r \Sigma_r \Phi_r^T, \quad (34)$$

where $\Psi_r = (\boldsymbol{\psi}^{(1)}, \dots, \boldsymbol{\psi}^{(r)})$ and $\Phi_r = (\boldsymbol{\phi}^{(1)}, \dots, \boldsymbol{\phi}^{(r)})$ are the matrices with r columns of the left and right singular vectors respectively corresponding to the r largest singular values $\sigma_1 \geq \dots \geq \sigma_r$ with $\Sigma_r = \text{diag}(\sigma_1, \dots, \sigma_r)$. For any new F , we can perform the dimension reduction by the linear projection

$$F \approx F_r := \bar{F} + \sum_{i=1}^r \beta_i \boldsymbol{\psi}^{(i)} \quad \text{where} \quad \beta_i = (F - \bar{F})^T \boldsymbol{\psi}^{(i)}. \quad (35)$$

Note that instead of using the sample-based PCA dimension reduction for the input parameter, we can also compute a discrete version of a truncated Karhunen–Loève expansion (KLE) for $\mathbf{m} \sim \mathcal{N}(\mathbf{m}_{\text{prior}}, \Gamma_{\text{prior}})$ as

$$\mathbf{m} \approx \mathbf{m}_r := \mathbf{m}_{\text{prior}} + \sum_{i=1}^r \beta_i \boldsymbol{\psi}^{(i)} \quad \text{with} \quad \beta_i = \sqrt{\lambda_i} \eta_i, \quad (36)$$

where $(\lambda_i, \boldsymbol{\psi}^{(i)})$, $i = 1, \dots, r$, are the eigenpairs of the covariance matrix Γ_{prior} , and $\eta_i \sim \mathcal{N}(0, 1)$. The sample-based PCA projection converges to the KLE projection with an increasing number of samples.

4.2. Derivative-informed neural operators

Based on the dimension reduction for the input and output, we construct a neural network surrogate $F_{\text{NN}}(\mathbf{m})$ of the PtO map $F(\mathbf{m})$, or the discrete observation operator (thus neural operator), as

$$F(\mathbf{m}) \approx F_{\text{NN}}(\mathbf{m}) := \mathcal{D}_{\Psi_F} \circ \Phi_{\boldsymbol{\theta}} \circ \mathcal{E}_{\Psi_{\mathbf{m}}}(\mathbf{m}), \quad (37)$$

where $\mathcal{E}_{\Psi_{\mathbf{m}}} : \mathbb{R}^{d_{\mathbf{m}}} \rightarrow \mathbb{R}^{r_{\mathbf{m}}}$ represents an encoder defined by the linear projection with basis $\Psi_{\mathbf{m}} \in \mathbb{R}^{d_{\mathbf{m}} \times r_{\mathbf{m}}}$ as in (30) by DIS or in (36) by KLE. The output of the encoder is the $r_{\mathbf{m}}$ -dimensional coefficient vector $\mathcal{E}_{\Psi_{\mathbf{m}}}(\mathbf{m}) = \boldsymbol{\beta}_{\mathbf{m}} = (\beta_1, \dots, \beta_{r_{\mathbf{m}}})^T$ defined in the linear projections. $\mathcal{D}_{\Psi_F} : \mathbb{R}^{r_F} \rightarrow \mathbb{R}^{d_F}$ represents a decoder by the linear projection with basis $\Psi_F \in \mathbb{R}^{d_F \times r_F}$ as in (33) by DOS or in (35) by PCA. Note that the full dimension of the observable vector F is d_F , the same as the number of candidate sensors d_s . The map $\Phi_{\boldsymbol{\theta}} : \mathbb{R}^{r_{\mathbf{m}}} \rightarrow \mathbb{R}^{r_F}$ is given by a neural network parametrized by parameters $\boldsymbol{\theta}$, which takes the $r_{\mathbf{m}}$ -dimensional input $\boldsymbol{\beta}_{\mathbf{m}}$ and map it to a r_F -dimensional output to approximate the coefficient vector $\boldsymbol{\beta}_F$ of the linear projection of F as in (33) by DOS or in (35) by PCA. One example is given by

$$\Phi_{\boldsymbol{\theta}}(\mathbf{z}) := Z_L \mathbf{z}_L + \mathbf{b}_L, \quad (38)$$

where

$$\mathbf{z}_{l+1} = \mathbf{z}_l + W_l h_l(Z_l \mathbf{z}_l + \mathbf{b}_l), \quad l = 0, \dots, L-1, \quad (39)$$

is a sequence of ResNets [50] with input $\mathbf{z}_0 = \mathbf{z}$, h_l represents elementwise nonlinear activation functions, $\boldsymbol{\theta} = (W_0, Z_0, \mathbf{b}_0, \dots, W_{L-1}, Z_{L-1}, \mathbf{b}_{L-1}, Z_L, \mathbf{b}_L)$ collects all the neural network parameters. Note that the neural network size depends only on r_F and $r_{\mathbf{m}}$, which is small so a relatively small number of data would be sufficient for the training of the neural network.

To generate the training data, we draw N_t random samples $\mathbf{m}^{(n)}$ and solve the PDE to obtain $F^{(n)}$, $n = 1, \dots, N_t$. Then we project them to their corresponding reduced basis and obtain $\boldsymbol{\beta}_{\mathbf{m}}^{(n)}$ and $\boldsymbol{\beta}_F^{(n)}$. Note that the samples in computing the projection basis in the last section do not need to be the same samples drawn here. In practice, we take a subset of samples drawn here to compute the projection basis first. To construct the neural network surrogate that is accurate not only for the PtO map $F(\mathbf{m})$ but also for its Jacobian $J(\mathbf{m}) = \nabla_{\mathbf{m}} F(\mathbf{m})$, we also compute a reduced Jacobian $J_r^{(n)} = \Psi_F^T J^{(n)} \Psi_{\mathbf{m}} \in \mathbb{R}^{r_F \times r_{\mathbf{m}}}$ with the full Jacobian $J^{(n)} = \nabla_{\mathbf{m}} F^{(n)}$ at each of the N samples. This computation requires the solve of $\min(r_F, r_{\mathbf{m}})$ linearized PDEs corresponding to the PDE model (1). These linearized PDEs have the same linear operator $\partial_u \mathcal{R}$ or its adjoint at each sample $\mathbf{m}^{(n)}$. The computational cost can be amortized by, e.g., one LU factorization of the linear operator and repeated use of this factorization as a direct solver.

To this end, we define the loss function for the training of the neural network with two terms as

$$\ell(\boldsymbol{\theta}) = \frac{1}{N_t} \sum_{n=1}^{N_t} \|\boldsymbol{\beta}_F^{(n)} - \Phi_{\boldsymbol{\theta}}(\boldsymbol{\beta}_{\mathbf{m}}^{(n)})\|_2^2 + \lambda \|J_r^{(n)} - \nabla_{\boldsymbol{\beta}} \Phi_{\boldsymbol{\theta}}(\boldsymbol{\beta}_{\mathbf{m}}^{(n)})\|_2^2, \quad (40)$$

where the first term measures the difference between the projected PtO map and its neural network approximation in Euclidean norm, and the second term is informed by the derivative (thus DINO [29]) and measures the difference of their Jacobians in Frobenius norm. Note that $\nabla_{\boldsymbol{\beta}} \Phi_{\boldsymbol{\theta}} \in \mathbb{R}^{r_F \times r_{\mathbf{m}}}$ represents the Jacobian of the neural network output $\Phi_{\boldsymbol{\theta}}$ with respect to its input, which can be computed by automatic differentiation. The evaluation of the loss function and its gradient with respect to the parameters $\boldsymbol{\theta}$ is very efficient since all quantities depend only on the small reduced dimensions r_F and $r_{\mathbf{m}}$. We remark that the scaling parameter λ in the loss function (40) can be properly tuned to balance the two terms. We find that in practice $\lambda = 1$ is sufficient to achieve improved accuracy of the PtO map and its reduced Jacobian.

4.3. Efficient computation of the MAP point and the eigenpairs with DINO

Once the neural network $\Phi_{\boldsymbol{\theta}}$ defined in (38) is well trained with the derivative-informed loss function (40), we obtain a DINO surrogate F_{NN} of the PtO map F as in (37). As the surrogate is trained to approximate both the PtO map and its derivative, we can use it in a derivative-based optimization algorithm to solve problem (19) for the MAP point and compute the eigenpairs in (22) that involve the Jacobian as in (21). Thanks to the dimension reduction of the input parameters and output observables by the encoder and decoder as in (37), respectively, we can efficiently compute the MAP point and eigenpairs in the reduced spaces, which can significantly reduce the computational cost.

More specifically, to compute the MAP point, we solve the following optimization problem

$$\boldsymbol{\beta}_{\text{MAP}}^{y, \xi} = \arg \min_{\boldsymbol{\beta}} \frac{1}{2} \|\mathbf{y} - \xi \mathcal{D}_{\Psi_F} \circ \Phi_{\boldsymbol{\theta}}(\boldsymbol{\beta})\|_{\Gamma_{\text{noise}}^{-1}}^2 + \frac{1}{2} \|\boldsymbol{\beta}\|_{\Gamma_{\mathbf{m}}^{-1}}^2, \quad (41)$$

where $\Gamma_{\mathbf{m}}^{-1} = \Psi_{\mathbf{m}}^T \Gamma_{\text{prior}}^{-1} \Psi_{\mathbf{m}} \in \mathbb{R}^{r_{\mathbf{m}} \times r_{\mathbf{m}}}$ can be computed once and used repeatedly in solving the optimization problem. When $\Psi_{\mathbf{m}}$ is taken as the DIS basis computed from (29), we have $\Gamma_{\mathbf{m}}^{-1} = \mathbb{I}$. Note that this optimization problem has a reduced dimension $r_{\mathbf{m}} \ll d_{\mathbf{m}}$, compared to the high $d_{\mathbf{m}}$ -dimensional optimization problem (19). We propose to solve it using a gradient-based optimization algorithm, e.g., the quasi-Newton LBFGS algorithm, where the derivative of the neural network $\Phi_{\theta}(\beta)$ with respect to the input β can be efficiently computed by automatic differentiation. Once $\beta_{\text{MAP}}^{\mathbf{y}, \xi}$ is obtained, we can compute the approximate high-dimensional MAP point $\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi}$ by the linear projection, e.g., DIS in (30).

To compute the eigenpairs of (22) at the MAP point using the DINO surrogate F_{NN} in (37), we first observe that its Jacobian is given by

$$\nabla_{\mathbf{m}} F_{\text{NN}}(\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi}) = \Psi_F \nabla_{\beta} \Phi_{\theta}(\beta_{\text{MAP}}^{\mathbf{y}, \xi}) \Psi_{\mathbf{m}}^T \Gamma_{\text{prior}}^{-1}, \quad (42)$$

where we use the DIS basis $\Psi_{\mathbf{m}}$ for the input encoder, and Ψ_F , either PCA basis or DOS basis for the output decoder. Second, by observing the similarity of the generalized eigenvalue problem (22) and the generalized eigenvalue problem (29) in computing the DIS basis, we propose to approximate the eigenvector \mathbf{w}_i of (22) by the linear projection with DIS basis as $\mathbf{w}_i = \Psi_{\mathbf{m}} \mathbf{u}_i \in \mathbb{R}^{d_{\mathbf{m}}}$ with the coefficient vector $\mathbf{u}_i \in \mathbb{R}^{r_{\mathbf{m}}}$, then the generalized eigenvalue problem (22) can be simplified (by multiplying $\Psi_{\mathbf{m}}$ on both sides) as

$$\hat{H}_{\text{misfit}}^{\mathbf{y}, \xi}(\beta_{\text{MAP}}^{\mathbf{y}, \xi}) \mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad i = 1, \dots, r_{\mathbf{m}}, \quad (43)$$

where the reduced matrix $\hat{H}_{\text{misfit}}^{\mathbf{y}, \xi}(\beta_{\text{MAP}}^{\mathbf{y}, \xi}) \in \mathbb{R}^{r_{\mathbf{m}} \times r_{\mathbf{m}}}$ is given by

$$\hat{H}_{\text{misfit}}^{\mathbf{y}, \xi}(\beta_{\text{MAP}}^{\mathbf{y}, \xi}) = (\nabla_{\beta} \Phi_{\theta}(\beta_{\text{MAP}}^{\mathbf{y}, \xi}))^T \Psi_F^T \xi^T \Gamma_{\text{noise}}^{-1} \xi \Psi_F \nabla_{\beta} \Phi_{\theta}(\beta_{\text{MAP}}^{\mathbf{y}, \xi}), \quad (44)$$

which can be efficiently evaluated by automatic differentiation. Moreover, the eigenvalue problem (44) can also be efficiently solved in the reduced dimension $r_{\mathbf{m}}$. Once the MAP point and the eigenpairs are computed, we can evaluate all the optimality criteria as in Section 3.3.

4.4. Swapping greedy algorithm

In this section, we present a swapping greedy algorithm as a modification of that developed in [5] to optimize the experimental design for the optimal sensor placement problem. Recall that we need to choose r_s sensors out of d_s candidate sensor locations. We first apply a standard greedy algorithm to place the sensor one by one from 1 to r_s sensors according to one of the optimality criteria, and then employ a swapping greedy algorithm to swap the selected r_s sensors one by one with the rest of the sensors to improve the optimality criterion until a stopping condition is satisfied, see Algorithm 1. More specifically, we initialize an empty sensor set in line 3, select and add the sensors s_t^* , $t = 1, \dots, r_s$, one by one by a greedy algorithm in line 4 to 10 to maximize one of the optimality criteria, where the optimality criteria are computed either by high-fidelity approximation as in Section 3.3, or by neural network acceleration as in Section 4.3. Then, we start from the selected sensor set by the greedy algorithm and swap the sensors from it one by one with the rest of the candidate sensors in line 12 to 22. We stop the swapping loop if the improved design optimality is smaller than tolerance or if the number of the swapping loops is larger than a maximum number k_{max} . We remark that an initial sensor set with r_s sensors is selected by a leverage score in [5] before starting the swapping loop since the optimality criteria can only be evaluated in an online optimization stage for a fixed number (r_s) of sensors in its algorithm. In contrast, we can efficiently evaluate the optimality criteria for an arbitrary number of sensors, which allows us to initialize the sensor set by a greedy algorithm before swapping. We observe fewer swapping loops to converge in our example (1 to 2 loops are sufficient).

4.5. Computational complexity

The total computation for the swapping greedy algorithm requires $O(k_s r_s d_s)$ evaluations of the optimality criterion to select r_s sensors from d_s candidates with $k_s \leq k_{\text{max}}$ loops of the swapping greedy algorithm. Note that the greedy algorithm for initialization takes $O(r_s d_s)$ evaluations of the optimality criterion. Each of the optimality criteria takes N_s times of solving the optimization problem (41) to compute the MAP point

Algorithm 1 Swapping greedy algorithm

1: **Input:** A set $S = \{s_1, \dots, s_{d_s}\}$ of d_s candidate sensors, a budget of r_s sensors, a maximum number of swapping loops k_{\max} , and an optimality tolerance ε_{\min} .

2: **Output:** A set S^* of r_s optimal sensors.

3: Set an empty sensor set $S^0 = \emptyset$.

4: **for** $t = 1, \dots, r_s$ **do**

5: Choose the optimal sensor s_t^* at step t as

6:
$$s_t^* = \arg \max_{s \in S \setminus S^{t-1}} \text{optimality}(\xi(\{s\} \cup S^{t-1})),$$

7: where $\xi(\{s\} \cup S^{t-1})$ is the design matrix for the selected sensors $\{s\} \cup S^{t-1}$.

8: Update the set of selected sensors S^t as

9:
$$S^t = \{s_t^*\} \cup S^{t-1}.$$

10: **end for**

11: Set an initial loop number $k = 0$ and design improvement $\varepsilon = 2\varepsilon_{\min}$.

12: **while** $k \leq k_{\max}$ and $\varepsilon > \varepsilon_{\min}$ **do**

13: Denote a set of r_s selected sensors as $S_k^{r_s} = S^{r_s} = \{s_1, \dots, s_{r_s}\}$, and compute $\text{optimality}(\xi(S_k^{r_s}))$ for the corresponding experimental design $\xi(S_k^{r_s})$.

14: **for** $t = 1, \dots, r_s$ **do**

15: Select the candidate sensor s_t^* to swap with the t -th sensor s_t in S^{r_s} as

16:
$$s_t^* = \arg \max_{s \in S \setminus S^{r_s}} \text{optimality}(\xi(S^{r_s} \setminus \{s_t\} \cup \{s\})),$$

17: Update S^{r_s} by swapping its t -th sensor s_t with s_t^* .

18: **end for**

19: Compute the $\text{optimality}(\xi(S^{r_s}))$ for the updated sensor set S^{r_s} , compute the incremental improvement of the optimality criterion

20:
$$\varepsilon = \text{optimality}(\xi(S^{r_s})) - \text{optimality}(\xi(S_k^{r_s})),$$

21: and update $k \leftarrow k + 1$.

22: **end while**

23: **return** $S^* = S^{r_s}$.

and N_s times of solving the eigenvalue problem (43) to compute the eigenpairs, where N_s is the number of samples in the SAA of the expectation in the optimality criteria.

If we only use the high-fidelity approximation without neural network acceleration, by the same swapping greedy algorithm we would have to solve $O(N_s k_s r_s d_s)$ times the optimization problem (19) to compute the high-fidelity MAP point and solve $O(N_s k_s r_s d_s)$ times the high-fidelity generalized eigenvalue problem (22) to compute the eigenpairs. By using an inexact Newton-CG algorithm to solve the optimization problem (19), we need to solve N_{nt} state PDEs and $N_{\text{nt}} N_{\text{cg}}$ linearized PDEs (for Hessian-vector product), with N_{nt} Newton iterations and N_{cg} (in average) CG iterations per Newton iteration. For the solution of the generalized eigenvalue problem (22) by a randomized SVD algorithm, we need to solve one state PDE and $O(r)$ linearized PDEs to compute r eigenpairs. We remark that if a direct solver is used, e.g., by first LU factorization and then its repeated use to solve the linearized PDEs, then the cost C_2 of each linearized PDE solve becomes much smaller than the cost C_1 to solve the state PDE, so that the additional computational cost for the Jacobian data generation is comparable or smaller (depending on the nonlinearity of the state

PDEs) than that for the PtO map data generation, as reported in our examples. The computational complexity in terms of the number of PDE solves is summarized in Table 1.

For the neural network construction, we first need to build the basis for the input and output dimension reduction, with $O(N_{\mathbf{m}})$ state PDE solves (using $O(N_{\mathbf{m}})$ samples for SAA in (29)) and $O(N_{\mathbf{m}}r_{\mathbf{m}})$ linearized PDE solves to get $r_{\mathbf{m}}$ bases for input dimension reduction, and $O(N_F)$ state and $O(N_F r_F)$ linearized PDE solves to get r_F bases for output dimension reduction. To generate N_t samples of the PtO map and its Jacobian in the reduced dimension for training the neural network surrogate, we need to solve N_t state PDEs (for PtO map), and $O(N_t r_t)$ linearized PDEs (for Jacobian) with $r_t = \min(r_{\mathbf{m}}, r_F)$. As the neural network is constructed in the reduced dimensions with a relatively small $O(r_{\mathbf{m}}r_F)$ degrees of freedom, its training cost is negligible compared to the data generation. Once trained in an offline stage, the evaluation of the neural network and its Jacobian is many orders of magnitude faster than the large-scale PDE solves and can be efficiently used to solve the Bayesian OED problem without the PDE solves in the online stage. The computational complexity in terms of the number of PDE solves is also summarized in Table 1.

# PDE solves	High-fidelity
MAP point	$O(N_s k_s r_s d_s N_{nt})C_1 + O(N_s k_s r_s d_s N_{nt} N_{cg})C_2$
Eigenvalue problem	$O(N_s k_s r_s d_s)C_1 + O(N_s k_s r_s d_s r)C_2$
# PDE solves	DINO
Dimension reduction	$O(N_{\mathbf{m}} + N_F)C_1 + O(N_{\mathbf{m}}r_{\mathbf{m}} + N_F r_F)C_2$
Training data	$O(N_t)C_1 + O(N_t r_t)C_2$

Table 1: Computational complexity in terms of the number of PDE solves, each state PDE with cost C_1 and each linearized PDE with cost C_2 . N_s : # samples to compute optimality criterion, k_s : # swapping loops, r_s : # sensors to select, d_s : # candidate sensors, N_{nt} : # Newton iterations, N_{cg} : # CG iterations per Newton iteration, r : # eigenpairs, N_t : # training samples, $r_t = \min(r_{\mathbf{m}}, r_F)$: the smaller number of the dimensions of the input projection $r_{\mathbf{m}}$ and the output projection r_F , $N_{\mathbf{m}}$ and N_F : # samples used to construct the input and output dimension reduction.

5. Numerical result

In this section, we demonstrate the accuracy, scalability, and efficiency of our proposed computational framework for two Bayesian OED problems. For the accuracy of neural network surrogates, we consider the approximation of the PtO map, its Jacobian, the MAP point, the eigenvalues, and the three different optimality criteria for Bayesian OED. For scalability, we show the preserved accuracy and cost in terms of the number of PDE solves with increasing parameter dimensions. For efficiency, we compare the computational cost of the neural network surrogates with that of the high-fidelity approximation, including the cost of both offline training and online evaluation of the surrogate in selecting optimal sensors.

5.1. Test problems

We consider two test problems, one of a linear diffusion problem and the other of a nonlinear convection-diffusion-reaction problem with a nonlinear reaction term, both with infinite-dimensional parameter fields and leading to nonlinear Bayesian inverse problems. We present the two examples in this subsection and the numerical results side by side in the following subsections for ease of comparison.

5.1.1. Linear diffusion problem

For the first test problem, we consider Bayesian OED for optimal sensor placement to infer a permeability field in pressure-driven Darcy flow in a physical domain $D = (0, 1)^2$, which is governed by the following diffusion equation prescribed with suitable boundary conditions

$$\begin{aligned}
-\nabla \cdot (e^m \nabla u) &= 0 \text{ in } D, \\
u &= 1 \text{ on } \partial D_{\text{top}}, \\
u &= 0 \text{ on } \partial D_{\text{bottom}}, \\
e^m \nabla u \cdot \mathbf{n} &= 0 \text{ on } \partial D_{\text{sides}},
\end{aligned} \tag{45}$$

where the state variable u represents the pressure field of the Darcy flow driven by the pressure difference with Dirichlet boundary conditions $u = 1$ on the top boundary ∂D_{top} and $u = 0$ on the bottom boundary $\partial D_{\text{bottom}}$. A homogeneous Neumann boundary condition is assumed on the two sides $\partial D_{\text{sides}}$, where \mathbf{n} denotes the unit-length outward normal for the side boundaries. e^m represents a positive permeability field with lognormal distribution, i.e., the model parameter $m \sim \mathcal{N}(0, \mathcal{C})$ is assumed to be Gaussian random field with Matérn covariance $\mathcal{C} = \mathcal{A}^{-2} = (-\gamma\Delta + \delta I)^{-2}$, where we set $\gamma = 0.1$, and $\delta = 0.5$. We use a finite element method (FEM) with piecewise linear polynomials to approximate the parameter and pressure fields discretized using a uniform mesh of size 64×64 . A random sample of the parameter, the corresponding state as a solution of the diffusion equation, and its pointwise observation with Gaussian additive noise following $\mathcal{N}(0, 0.1)$ at the 50 candidate sensor locations are shown in the top of Figure 1. We consider 50 candidate sensors in the lower part of the physical domain and select 5 sensors from the 50 candidates.

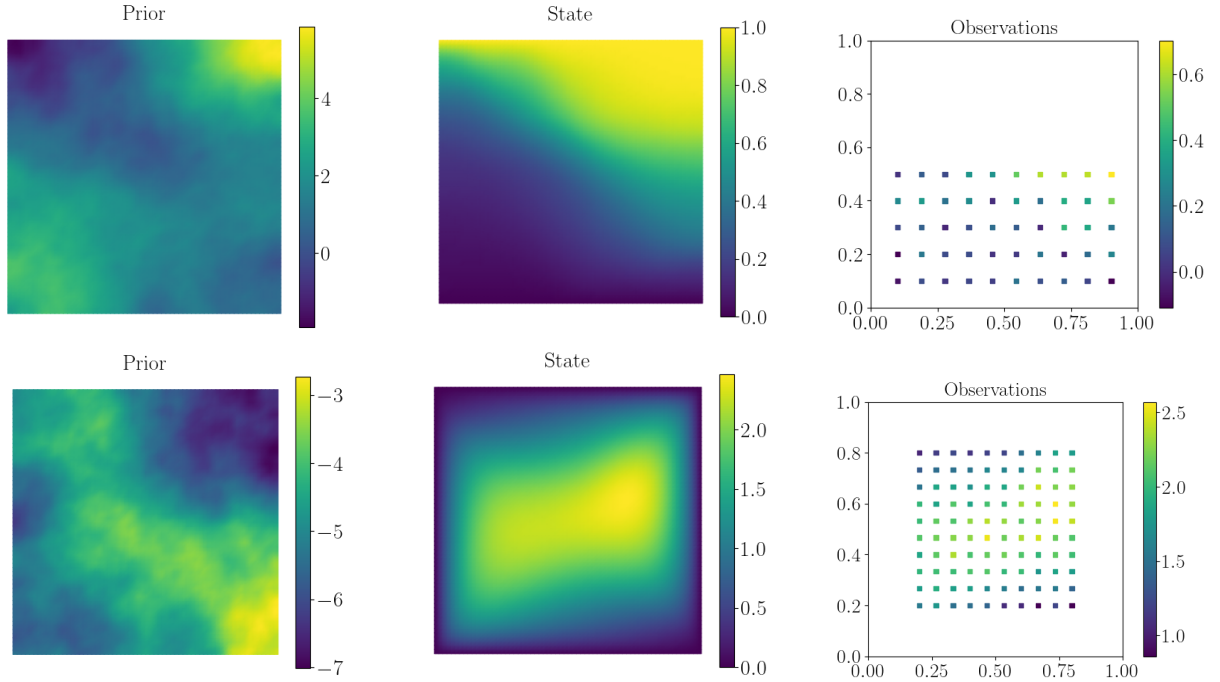


Figure 1: A random prior sample $m \sim \mathcal{N}(0, \mathcal{C})$ (left), the high-fidelity solution $u(m)$ by finite element approximation (middle), and the observation data \mathbf{y} at all the candidate observation points (right). Top: diffusion problem. Bottom: CDR problem.

5.1.2. Nonlinear convection-diffusion-reaction problem

For the second test problem, we consider Bayesian OED to infer a reaction coefficient field in mass transfer problem in the domain $D = (0, 1)^d$, $d = 2, 3$, which is governed by the nonlinear (semilinear) convection-diffusion-reaction (CDR) equation

$$\begin{aligned} -\nu\Delta u + \mathbf{v} \cdot \nabla u + e^m u^3 &= f \text{ in } D, \\ u &= 0 \text{ on } \partial D, \end{aligned} \tag{46}$$

where a homogeneous Dirichlet boundary condition is prescribed on the whole boundary ∂D . The source term is given as a Gaussian bump $f(\mathbf{x}) = \max(0.5, \exp(-25\|\mathbf{x} - \mathbf{x}_s\|_2^2))$ at $\mathbf{x}_s = (0.7, 0.7)$ (2D) and $\mathbf{x}_s = (0.7, 0.7, 0.7)$ (3D). The velocity field \mathbf{v} in the convection term is obtained as the solution of the following

steady-state Navier–Stokes equations

$$\begin{aligned} -\nu \Delta \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v} + \nabla p &= 0 \text{ in } D, \\ \nabla \cdot \mathbf{v} &= 0 \text{ in } D, \\ \mathbf{v} &= \mathbf{g} \text{ on } \partial D, \end{aligned} \tag{47}$$

where $\mathbf{g} = (0, 1)$ (2D) and $\mathbf{g} = (0, 1, 0)$ (3D) on left wall, $\mathbf{g} = (0, -1)$ (2D) and $\mathbf{g} = (0, -1, 0)$ (3D) on the right wall, and zero elsewhere. We set the viscosity $\nu = 0.01$. We use a finite element method to solve the equations with piecewise quadratic polynomials for the approximation of the velocity field \mathbf{v} and piecewise linear polynomials for the approximation of the pressure field p , the parameter field m , and the concentration field u . The finite element solution u at a random parameter sample m with the same Gaussian prior distribution as in the last example, and its corresponding observation data \mathbf{y} at 100 candidate observation locations are shown at the bottom plots of Figure 1. We seek to choose 10 sensors out of the 100 candidates for the Bayesian OED problem.

5.2. Dimension reduction

We perform dimension reduction for the high-dimensional input parameters in both application problems and for the output observables in the second problem since the output dimension $d_s = 50$ is relatively small in the first problem. We first use a uniform mesh of size 64×64 , which leads to a discretization of the infinite-dimensional input parameter m to a 4,225-dimensional parameter vector \mathbf{m} by finite element approximation as in (17). We compute the eigenpairs for the KLE projection (36) and the DIS projection (30) by solving the eigenvalue problem for Γ_{prior} and the generalized eigenvalue problem (29), respectively. We use 1,024 random samples to approximate the expectation in (29). The eigenvalues and eigenvectors of KLE and DIS are shown in Figure 2 and 3. From Figure 2 we can observe that the DIS eigenvalues are smaller and also decay faster than the KLE eigenvalues for both problems. From the second and third rows of Figure 3 we can see that the DIS basis functions are different for the two problems and the first DIS basis functions expose the features of the solutions of the two PDE problems as shown in Figure 1, while the KLE basis functions are the same for the two problems as shown in the first row.

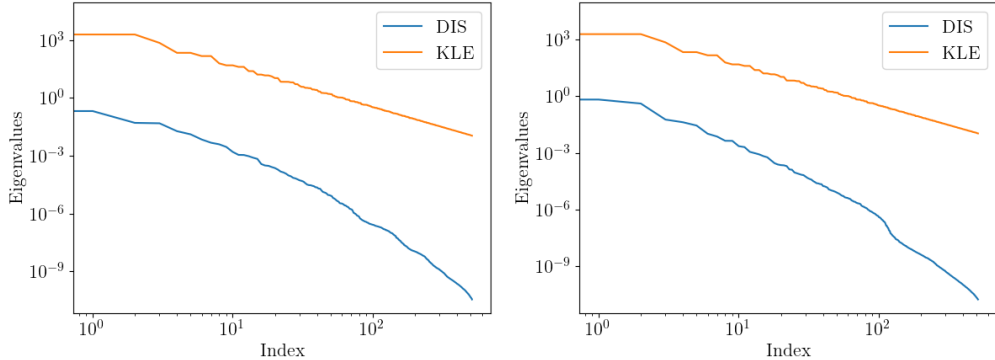


Figure 2: Decay of the eigenvalues of the prior covariance Γ_{prior} used in the KLE approximation of the parameter (36) and the eigenvalues of the generalized eigenvalue problem (29) used in DIS approximation of the parameter (30). Left: for the diffusion problem. Right: for the CDR problem.

The relative errors of different input and output projection methods are shown in Figure 4. Specifically, we compute the mean of the relative input projection errors for the observables and the Jacobian as

$$\mathbb{E}_{\mathbf{m}} \left[\frac{\|F(\mathbf{m}) - F(P_r(\mathbf{m}))\|_2}{\|F(\mathbf{m})\|_2} \right] \text{ and } \mathbb{E}_{\mathbf{m}} \left[\frac{\|J(\mathbf{m}) - J(P_r(\mathbf{m}))\|_F}{\|J(\mathbf{m})\|_F} \right] \tag{48}$$

where P_r denotes a linear projector of the parameter vector \mathbf{m} by either KLE in (36) or DIS in (30), $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the Euclidean norm for the observable vector and the Frobenius norm for the Jacobian

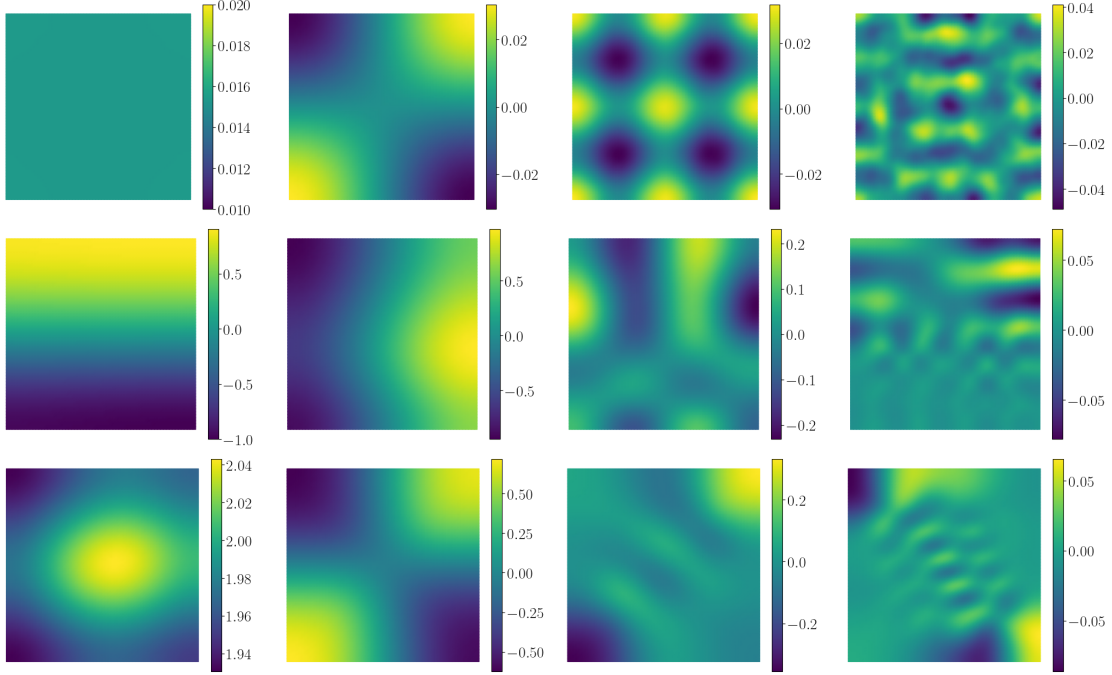


Figure 3: KLE bases 1, 4, 16, 64 (top). DIS bases 1, 4, 16, 64 for the diffusion (middle) and CDR (bottom) problems.

matrix, respectively. More explicitly, we evaluate the projected Jacobian matrix as

$$J(P_r(\mathbf{m})) = J(\mathbf{m})\Psi_{\mathbf{m}}^{\text{KLE}}(\Psi_{\mathbf{m}}^{\text{KLE}})^T \text{ or } J(P_r(\mathbf{m})) = J(\mathbf{m})\Psi_{\mathbf{m}}^{\text{DIS}}(\Psi_{\mathbf{m}}^{\text{DIS}})^T \Gamma_{\text{prior}}^{-1} \quad (49)$$

for the KLE basis $\Psi_{\mathbf{m}}^{\text{KLE}}$ or the DIS basis $\Psi_{\mathbf{m}}^{\text{DIS}}$. The expectation in (48) is evaluated by SAA using 100 samples. We also compute the relative input projection error for the MAP point in (19) as

$$\mathbb{E}_{\mathbf{y}} \left[\frac{\|\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi} - P_r(\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi})\|_M}{\|\mathbf{m}_{\text{MAP}}^{\mathbf{y},\xi}\|_M} \right] \quad (50)$$

where $\|\mathbf{m}\|_M = \sqrt{\mathbf{m}^T M \mathbf{m}}$ with the mass matrix M defined in (18). The expectation is evaluated by SAA using 100 samples of $\mathbf{y} = F(\mathbf{m}) + \epsilon$ by drawing random samples \mathbf{m} and ϵ . From the first and second rows of Figure 4, we can observe that using the DIS basis yields smaller projection errors for all three quantities. Moreover, the DIS basis is particularly advantageous for the approximation of Jacobian, which plays a key role in evaluating the optimality criteria. On the other hand, as shown in the third row of Figure 4, the output projection used in the CDR problem is less sensitive to the projection basis of PCA in (35) and DOS in (33), as measured by the average (using 100 samples) of the relative projection error for the observables and the Jacobian as

$$\mathbb{E}_{\mathbf{m}} \left[\frac{\|F(\mathbf{m}) - F_r(\mathbf{m})\|_2}{\|F(\mathbf{m})\|_2} \right] \text{ and } \mathbb{E}_{\mathbf{m}} \left[\frac{\|J(\mathbf{m}) - \Psi_F \Psi_F^T J(\mathbf{m})\|_F}{\|J(\mathbf{m})\|_F} \right], \quad (51)$$

where the projected observables F_r are defined in (35) or (33) with corresponding PCA basis $\Psi_F = \Psi_F^{\text{PCA}}$ or DOS basis $\Psi_F = \Psi_F^{\text{DOS}}$. For the following computations, we use 128 DIS basis functions for input projection in both problems and 30 PCA basis functions for output projection in the second problem, which leads to less than 1% error in both the input and output dimension reduction for the PtO map.

5.3. Accuracy and scalability of DINO approximations

We construct the DINO surrogates as presented in Section 4.2 using PyTorch [51]. Specifically, we construct the encoder by the DIS projection with the reduced dimension $r_{\mathbf{m}} = 128$ as in (30). For the

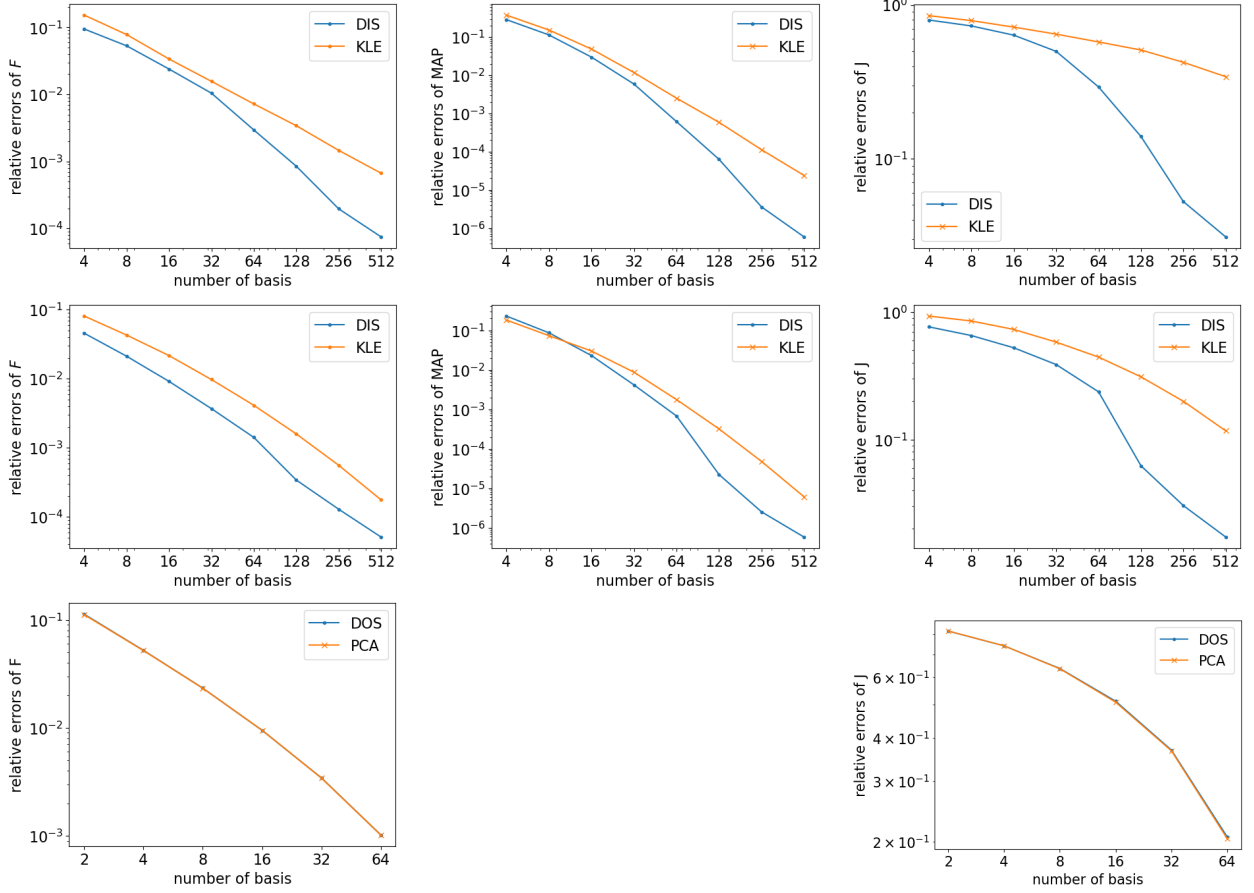


Figure 4: Relative errors of the input projection (top and middle rows for the two 2D examples, respectively) and output projection (bottom row for the CDR problem) for the observables (left), the MAP point (middle), and the Jacobian (right). We use DIS (30) and KLE (36) basis for input projection, and DOS (33) and PCA (35) basis for output projection.

CDR problem, we also construct the decoder by the PCA projection with the reduced dimension $r_F = 30$ as in (35). We connect the encoder layer with one linear layer, three ResNet layers, and another linear layer before the decoder layer. The two linear layers map the input and output dimensions to the same dimension as in the ResNet layers, for which we choose 100. We use sigmoid as the activation function inside the ResNet layers and tanh for all other layers. This specific architecture is among the best (with the smallest generalization errors) of several ones we tested with different numbers of layers, dimensions, and activation functions. We use an Adam optimizer [52] with a learning rate of 0.001. For the diffusion model, we train the neural network with Jacobian for 100 epochs and without Jacobian for 200 epochs. For the CDR model, we train the neural network with Jacobian for 200 epochs and without Jacobian for 300 epochs. We select these numbers of epochs based on when the validation error stops decreasing. We use the libraries FEniCS [53], hIPPYlib [54], and hIPPYflow [55] to generate the high-fidelity data, using a mesh of size 64×64 unless otherwise stated in scalability test, and project the data to get (β_m, β_F, J_r) used in the loss function (40) for training and testing.

5.3.1. The accuracy of the neural network approximations

We train the neural networks ten times starting from different initializations with different training sizes. Then we compute for each case the neural network approximations of the PtO map F and the reduced Jacobian J_r , as well as the MAP points by solving the optimization problem (41). We report the mean of the relative errors averaged over the ten trials of the neural network training in Figure 5. The relative errors

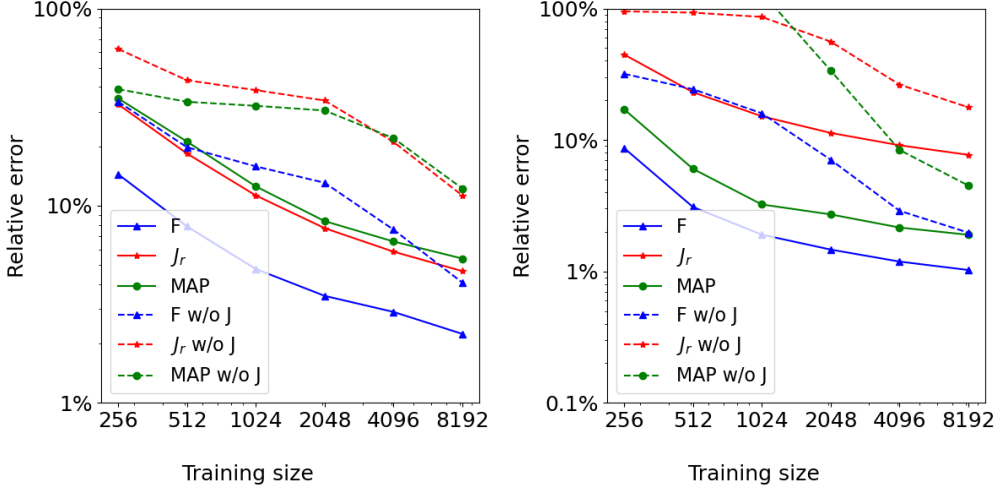


Figure 5: Mean of relative errors with increasing training size for the neural network approximations of the PtO map F , the reduced Jacobian J_r , and the MAP point \mathbf{m}_{MAP} . The results in solid and dashed lines are for the neural networks trained with and without the Jacobian, respectively. Diffusion (left) and CDR (right) problems.

of the PtO map and the reduced Jacobian are defined as

$$\mathbb{E}_{\mathbf{m}} \left[\frac{\|F(\mathbf{m}) - F_{\text{NN}}(\mathbf{m})\|_2}{\|F(\mathbf{m})\|_2} \right] \text{ and } \mathbb{E}_{\mathbf{m}} \left[\frac{\|J_r(\mathbf{m}) - \nabla_{\beta} \Phi_{\theta}(\beta_{\mathbf{m}})\|_F}{\|J_r(\mathbf{m})\|_F} \right], \quad (52)$$

averaged over 1024 test samples, and the relative error of the MAP point is defined as

$$\mathbb{E}_{\mathbf{y}} \left[\frac{\|\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi} - \mathbf{m}_r(\beta_{\text{MAP}}^{\mathbf{y}, \xi})\|_M}{\|\mathbf{m}_{\text{MAP}}^{\mathbf{y}, \xi}\|_M} \right], \quad (53)$$

averaged over 200 samples, with $\mathbf{m}_r = \mathbf{m}_{\text{prior}} + \Psi_{\mathbf{m}}^{\text{DIS}} \beta_{\text{MAP}}^{\mathbf{y}, \xi}$. We use PyTorch’s LBFGS optimizer with 100 maximum iterations to compute the MAP points using the neural network surrogate (DINO) trained with the Jacobian data. For the neural network surrogate trained without Jacobian data, we have to use an additional 300 iterations of Pytorch’s Adam optimizer with a 0.01 learning rate before LBFGS to obtain a convergent computation of the MAP points. Increasing training size makes the neural network approximations more accurate for all three quantities. Moreover, the neural networks trained with the Jacobian information lead to much more accurate approximations than those trained without Jacobian, especially when the training size is relatively small, see Table 2 for the relative approximation errors (mean and standard deviation) at the training size of a relatively small size 1,024 and larger size 8,192.

To illustrate the accuracy of the MAP point, we take a random prior sample from the test data set, see left of Figure 6, solve the PDE, and generate synthetic observation data at all candidate sensors to compute the MAP point using FEM and DINO trained with 8,192 samples. The MAP points computed by DINO and FEM, as shown in the middle of Figure 6, are very close to each other with pointwise errors about two orders of magnitude smaller than the MAP point, see right of Figure 6. We can also observe that MAP points look similar but not very close to the prior samples because of the intrinsic ill-posedness of the inverse problems. Note that in solving the Bayesian OED problems, we only need to consider random samples drawn from the prior distribution so it is sufficient for DINO to be accurate for the prior samples. We also test DINO with an out-of-(prior)distribution sample (piecewise constant binned from a prior sample) in the CDR problem, see the bottom of Figure 6. The MAP point computed by DINO is still very close to the MAP point computed by FEM, which demonstrates the robustness of DINO for this problem. However, we do not expect this to hold for more general problems or more general out-of-distribution samples, especially those far from the prior distribution.

Linear diffusion problem				
Training size	1,024		8,192	
Loss function	with J	w/o J	with J	w/o J
F	4.77% (0.4%)	17.2% (0.42%)	2.23% (0.34%)	5.82% (0.26%)
J_τ	11.26% (0.23%)	40.32% (0.42%)	4.65% (0.1%)	15.4% (0.28%)
MAP	12.52% (0.65%)	34.48% (1.32%)	5.39% (1.39%)	17.23% (0.7%)
Nonlinear convection-diffusion-reaction problem				
Training size	1,024		8,192	
Loss function	with J	w/o J	with J	w/o J
F	1.9% (0.1%)	15.94% (0.93%)	1.02% (0.1%)	1.90% (0.06%)
J_τ	15.13% (0.21%)	86.24% (0.1%)	7.7% (0.21%)	17.69% (0.21%)
MAP	3.22% (0.14%)	132.1% (84.73%)	1.89% (0.36%)	4.38% (0.22%)

Table 2: Mean (and standard deviation) of the relative approximation errors of the neural network surrogates trained with 1,024 and 8,192 training samples for the loss function with Jacobian (DINO) and without (w/o) Jacobian for the two problems.

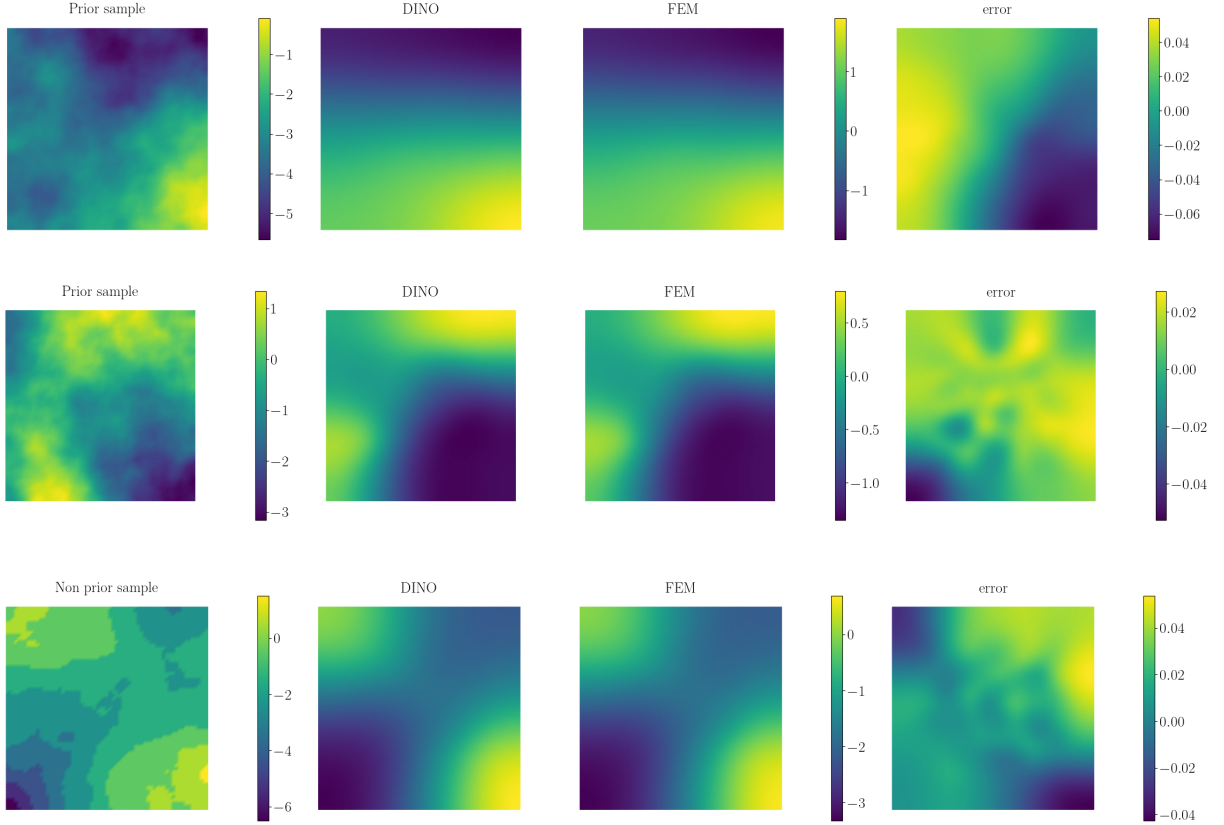


Figure 6: Prior samples (left), MAP points computed by the DINO surrogate and FEM (middle), and the errors of DINO approximation compared to FEM (right) for the diffusion problem (top) and the CDR problem (middle). Bottom: The comparison as above rows for an out-of-distribution sample (piecewise constant) for the CDR problem.

To demonstrate the high accuracy of the reduced Jacobian by DINO in solving the generalized eigenvalue problems, we plot the generalized eigenvalues of (22) by FEM and of (43) by DINO at the MAP points computed from three random test samples, as shown in Figure 7. We can observe that the dominant eigenvalues, the first ten in the leading three orders of magnitude, are almost indistinguishable for the two methods in both test problems, especially in the nonlinear CDR problem.

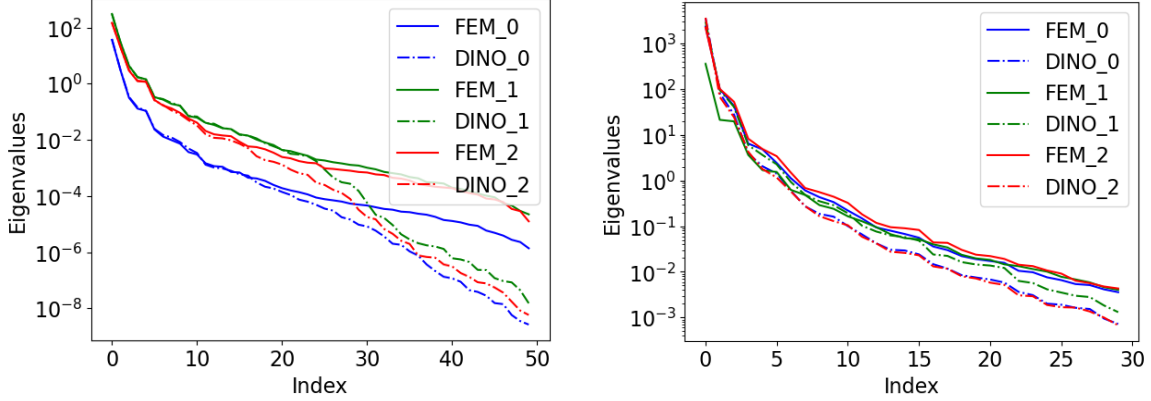


Figure 7: The generalized eigenvalues of (22) computed using FEM and the eigenvalues of (43) computed using DINO at three MAP points (computed from three random prior samples) for the diffusion problem (left) and the CDR problem (right).

5.3.2. The scalability of the neural network approximations

To check the scalability of the neural network approximations by DINO, we increase the parameter dimensions by refining the mesh. Thanks to the dimension reduction using the same reduced dimensions, the neural network size does not change which leads to the same training cost for different mesh sizes. Figure 8 indicates that the accuracy of the DINO approximations for increasing mesh sizes of 32×32 , 64×64 , and 128×128 remains the same for the PtO map F , the reduced Jacobian J_r , and the MAP point, which implies the scalability (dimension-independence) of the DINO approximations with respect to increasing parameter dimensions. This scalability also holds for the approximations of the generalized eigenvalues with the use of the DINO approximations of the reduced Jacobian as shown in Figure 9.

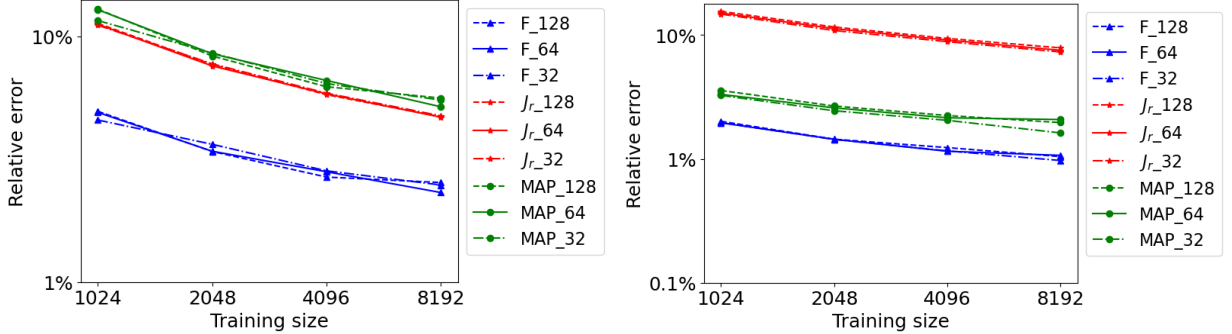


Figure 8: Mean of relative errors of the PtO map F , the reduced Jacobian J_r , and the MAP point with increasing training sizes and mesh sizes 32×32 , 64×64 , and 128×128 for the diffusion problem (left) and the CDR problem (right).

5.4. Accuracy verification and solution of Bayesian OED

For the Bayesian OED problem, we select 5 sensor locations from 50 candidates for the diffusion problem and 10 sensor locations from 100 candidates for the CDR problem as shown in Figure 1. To check that the DINO surrogates constructed for the full PtO map F at all candidate observations preserve the approximation accuracy for the selected number of observations, we plot the relative errors of the DINO approximation for the PtO map F_ξ at a random design ξ , its reduced Jacobian, the MAP point, and the generalized eigenvalues in Figure 10, which confirms the preserved accuracy for all quantities. Note that there are only 5 and 10 generalized eigenvalues for the 5 and 10 selected observations.

To this end, we have constructed and verified the accuracy and scalability of the DINO surrogates for approximating the quantities involved in computing the optimality criteria of the Bayesian OED. We

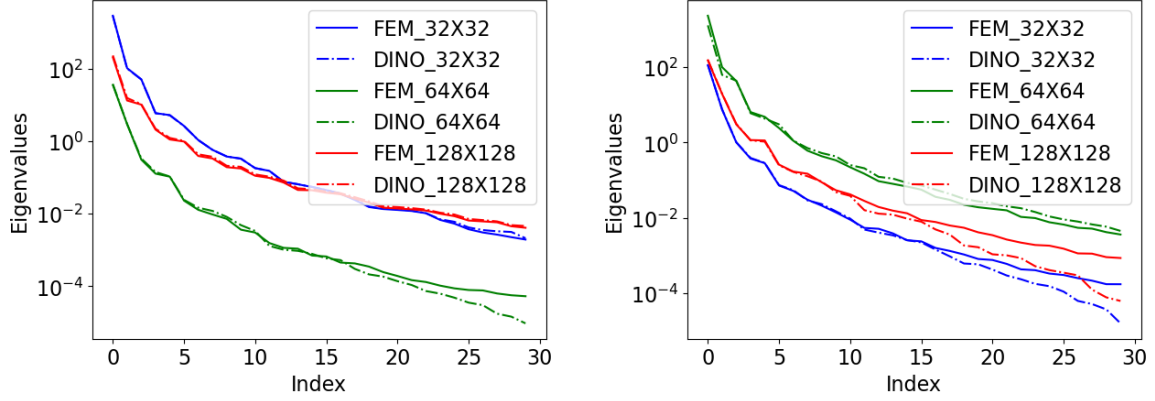


Figure 9: The generalized eigenvalues of (22) by FEM and the eigenvalues of (43) by DINO with increasing parameter dimensions with mesh sizes (32×32 , 64×64 , and 128×128) for the diffusion problem (left) and the CDR problem (right).

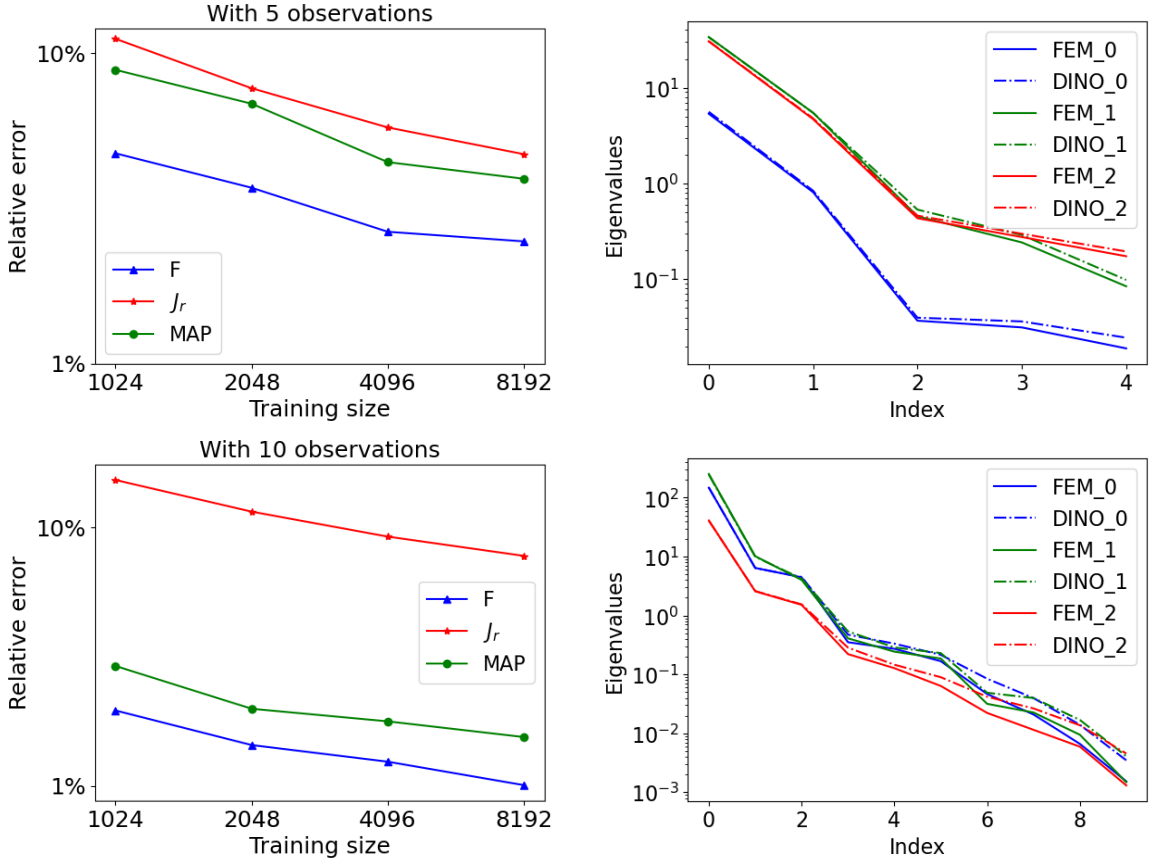


Figure 10: Left: mean of relative errors for the PtO map F_ξ , the reduced Jacobian J_r , and the MAP point with 5 and 10 observations for the diffusion problem (top) and CDR problem (bottom). Right: the eigenvalues by DINO and FEM.

compute these criteria, namely the A-optimality in (25), the D-optimality in (26), and the EIG-optimality in (27) using the eigenpairs from reduced eigenvalue decomposition. We plot the optimality criteria computed using DINO and FEM approximations at 200 random samples (\mathbf{m}, \mathbf{y}) and report both the R^2 scores and the mean and standard deviation of the relative errors in Figure 11. The results reveal a very high correlation

between the DINO and FEM computation of the optimality criteria with R^2 score mostly larger than 0.99 and with small relative errors. On the other hand, the relative errors for the three optimality criteria obtained by the neural network trained without the Jacobian (NN without J) are much larger than those obtained by DINO. This demonstrates the enhanced accuracy of the proposed method by incorporating derivative information in DINO. We use 2048 training samples for both DINO and NN without J for the plots in Figure 11. We also include the relative errors of approximate optimality criteria with the training sizes of 1024, 4096, and 8192 in Table 3. We observe that as the neural network approximations become more accurate with increasing training sizes, the approximation of the optimality criteria also become more accurate. Moreover, DINO leads to much more accurate approximations (particularly for the D-optimality and EIG-optimality) than the neural network trained without Jacobian, especially when the training size is relatively small (e.g., at 1024).

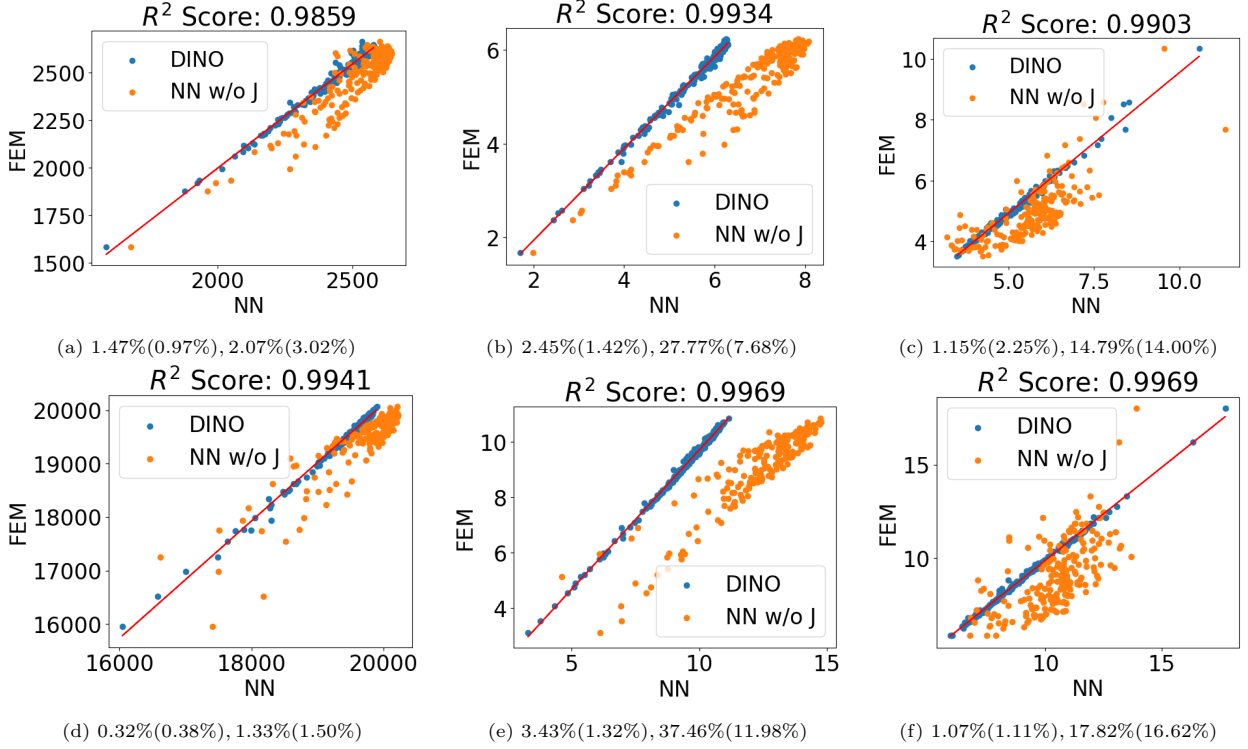


Figure 11: R^2 score (for correlation) of DINO vs FEM evaluations of the optimality criteria, mean and standard deviation of the relative errors obtained by DINO and the neural network trained without Jacobian information (NN w/o J) compared to FEM in the evaluation of the trace (left), determinant (middle), and information gain (right) at 200 random samples. We choose 5 sensors in the diffusion problem (top) and 10 sensors in the CDR problem (bottom).

We run the swapping greedy algorithm 1 by setting the stopping criteria $k_{\max} = 3$ and $\varepsilon_{\min} = 0.01$, and use DINO to find the optimal sensor locations according to the three optimality criteria of A-optimality, D-optimality, and EIG. To check the optimality of the sensors selected by the algorithm using DINO surrogates, we compute the three optimality criteria using the high-fidelity FEM at the selected sensors compared to 200 randomly selected sensors. The results are shown in Figure 12, from which we can see that optimality criteria achieve their largest values at the sensors selected using DINO surrogates according to the corresponding optimality criteria, which demonstrates the effectivity of the DINO surrogates. Moreover, note that larger optimality criteria correspond to smaller uncertainty in the model parameter estimation. The uncertainty indicated by the optimality criteria is effectively reduced by the optimal design from random design.

The corresponding sensors selected according to different optimality criteria are shown in Figure 13 for the two test problems with 5 and 10 sensors, respectively. We can see that different optimality criteria can

Linear diffusion problem						
Training size	1,024		4,096		8,192	
Optimality	with J	w/o J	with J	w/o J	with J	w/o J
A	1.41% (1.12%)	3.08% (3.26%)	1.49% (0.98%)	1.43% (1.21%)	1.46% (1.14%)	1.14% (1.03%)
D	3.95% (1.58%)	27.73% (9.48%)	2.07% (1.25%)	19.64% (5.14%)	1.59% (1.23%)	12.36% (2.12%)
EIG	2.26% (2.04%)	16.97% (9.92%)	1.33% (1.08%)	13.18% (7.57%)	0.98% (0.78%)	6.76% (3.83%)
Nonlinear convection-diffusion-reaction problem						
Training size	1,024		4,096		8,192	
Optimality	with J	w/o J	with J	w/o J	with J	w/o J
A	0.55% (0.47%)	1.76% (1.95%)	0.41% (0.21%)	0.45% (0.65%)	0.47% (0.31%)	0.42% (0.26%)
D	4.73% (2.96%)	81.7% (39.94%)	3.33% (1.17%)	10.48% (3.4%)	2.65% (0.91%)	4.98% (1.54%)
EIG	1.81% (1.18%)	48.61% (28.84%)	1.31% (0.73%)	4.93% (3.1%)	1.0% (0.54%)	2.14% (1.29%)

Table 3: Mean (and standard deviation) of the relative errors of optimality criteria using the neural network surrogates trained with 1,024, 4,096, and 8,192 samples with Jacobian (DINO) and without (w/o) Jacobian for the two problems.

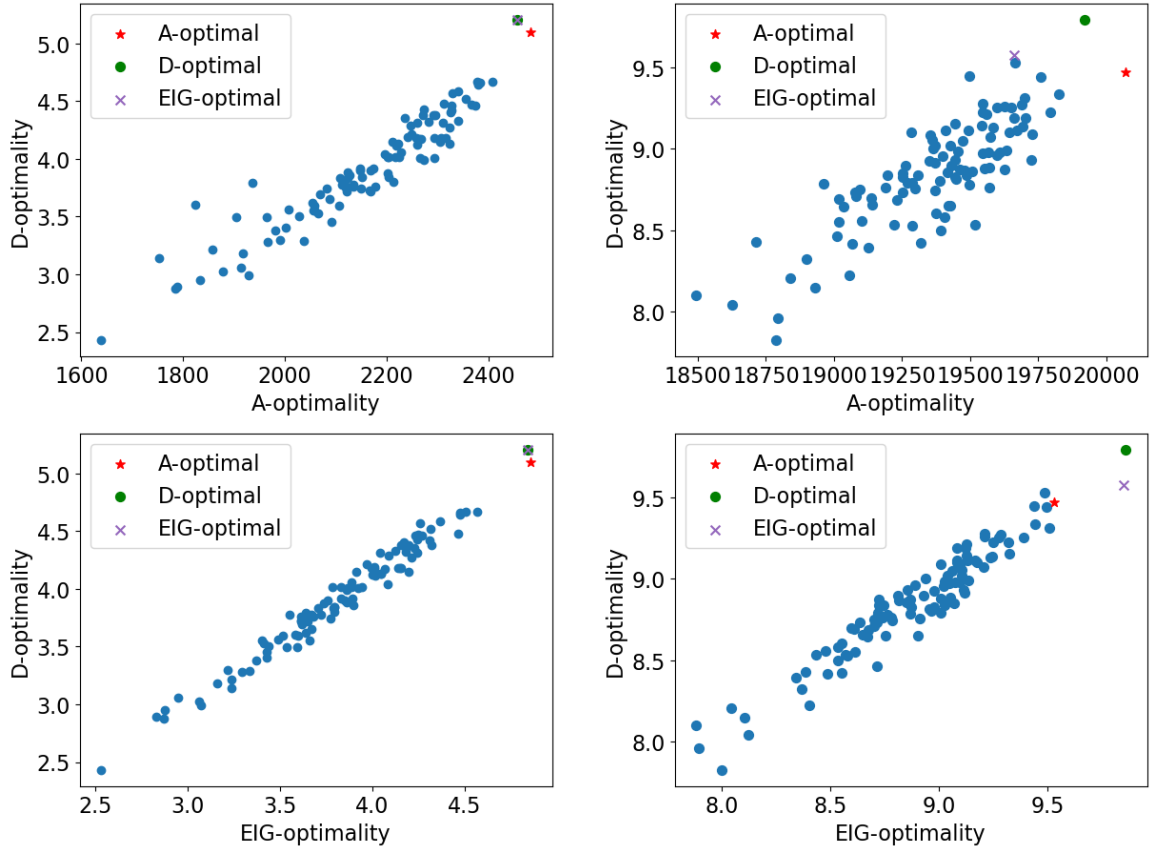


Figure 12: FEM optimality values at the optimal sensors selected using DINO by the swapping greedy algorithm 1 with different optimality criteria compared to those at 200 random designs. Diffusion (left) and CDR (right) problems.

lead to slightly different sensors. The uncertainty is effectively reduced from the prior to the posterior as observed in the pointwise prior and posterior variance in Figure 14. Moreover, we can also see that the reduction of the uncertainty by the (A-)optimal sensor locations is more than that by a random choice of the sensor locations, consistent with the smaller trace values.

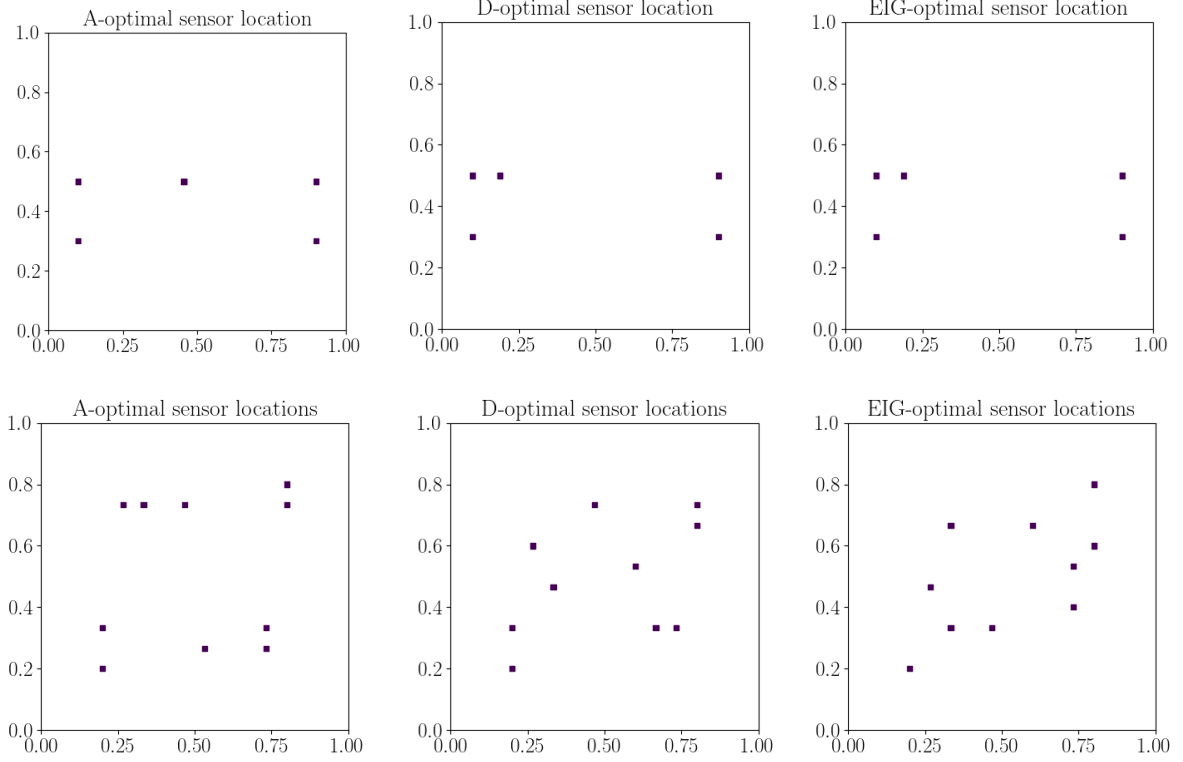


Figure 13: Optimal sensor locations selected by the swapping greedy algorithm 1 using the A-/D-/EIG-optimality criteria. 5 sensors selected for the diffusion problem (top) and 10 sensors selected for the CDR problem (bottom).

5.5. Application to a 3D test problem

We further apply the proposed method to solve a Bayesian OED problem governed by a 3D CDR equation (46) in the physical domain $D = (0, 1)^3$. We consider a selection of 8 sensor locations out of 512 candidates equidistantly distributed in the domain to infer the model parameter following Gaussian prior $m \sim \mathcal{N}(0, \mathcal{C})$ with Matérn covariance operator $\mathcal{C} = \mathcal{A}^{-2}$, where $\mathcal{A} = -\gamma\Delta + \delta I$, we set $\gamma = 0.4$ and $\delta = 1.0$.

For the FEM approximation of both the state and the parameter fields, we use piecewise linear polynomials with a uniform mesh of size $32 \times 32 \times 32$, leading to 35,937 dimensional discrete parameters. The FEM solution of the 3D forward problem is computationally much more expensive than that of the 2D forward problem. For the DINO surrogate, we use the same ResNet architecture as that for the 2D diffusion problem with a larger reduced input dimension of 256 and reduced output dimension of 128 for the 512 outputs.

We train the DINO with increasing numbers of training samples, which leads to increasing accuracy of the DINO approximation of the PtO map F , the reduced Jacobian J_r , and MAP point \mathbf{m}_{MAP} , as shown in Figure 15 (left). From this figure, we can also see much higher accuracy for all three quantities predicted by the neural networks trained with the Jacobian information compared to those trained without it. A comparison of the MAP point computed by DINO vs FEM is shown in Figure 16, from which we can observe an accurate approximation by DINO. The eigenvalues at three different MAP points computed by DINO are very accurate compared to FEM, as shown in Figure 15 (right).

We use the trained DINO surrogate to compute the three optimality criteria, i.e., the expected A/D-optimality and information gain, with 128 random samples in evaluating the expectation of the optimality criteria. We apply the swapping greedy algorithm to optimize the sensor locations, with the eight sensors selected using different optimality criteria shown in Figure 17, and the optimality values at the selected sensors shown in Figure 18. We observe that the optimality values at the selected sensors are optimal with

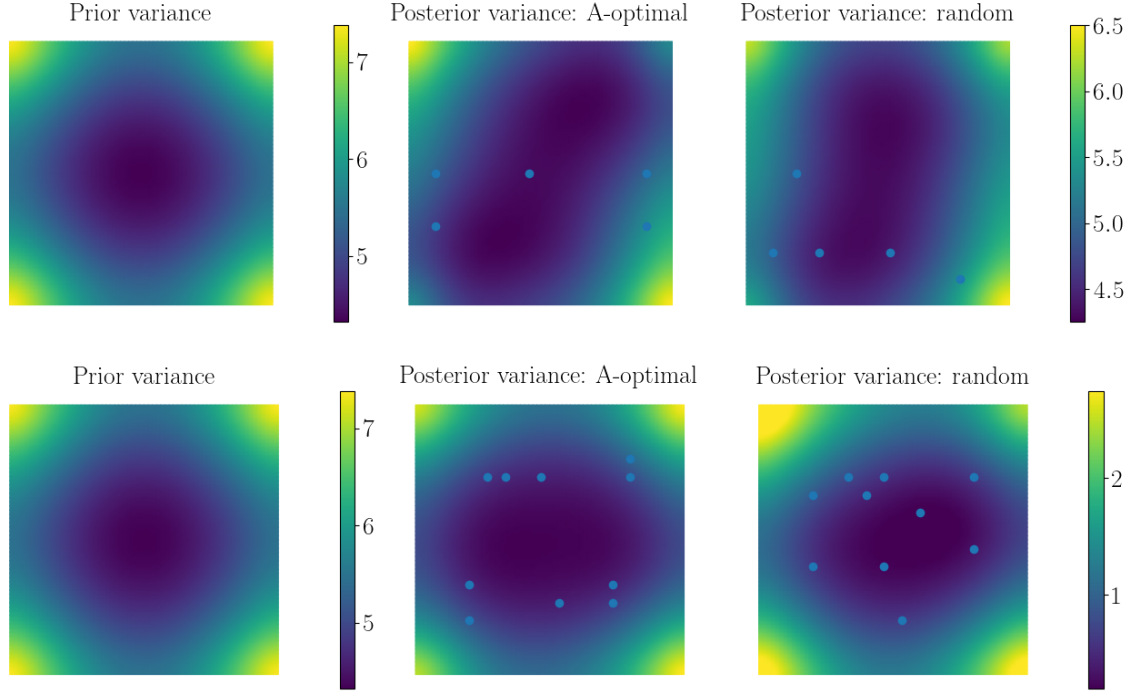


Figure 14: Pointwise prior variance (left), posterior variance with A-optimal (middle) and random sensor locations (right) for the diffusion (top) and the CDR problem (bottom). The trace of the posterior variance for A-optimal and random sensor placements are 20,176 and 20,599 for the diffusion problem, and 4,076 and 4,458 for the CDR problem, respectively.

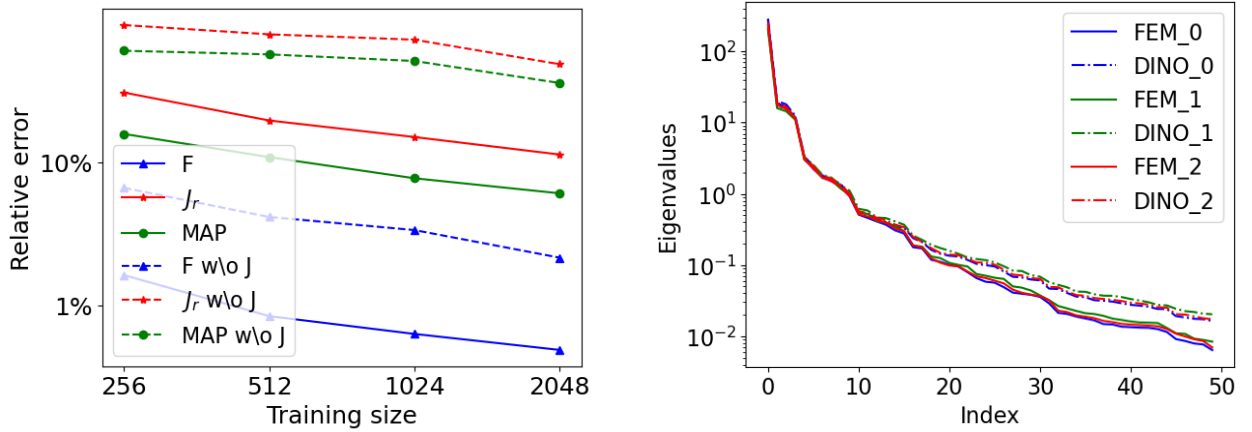


Figure 15: Mean of relative errors with increasing training size for the DINO approximations of the PtO map F , the reduced Jacobian J_r , and the MAP point \mathbf{m}_{MAP} (left). Comparison of the generalized eigenvalues by DINO and FEM (right).

respect to the corresponding optimality criteria and much larger than those at the randomly selected sensors, which demonstrates the effectiveness of the swapping greedy algorithm for the 3D Bayesian OED problem.

5.6. Efficiency of the computational framework

In this section, we demonstrate the efficiency of our proposed approach by reporting the computational cost of generating the training data, training the DINO surrogates, and their use in solving Bayesian OED

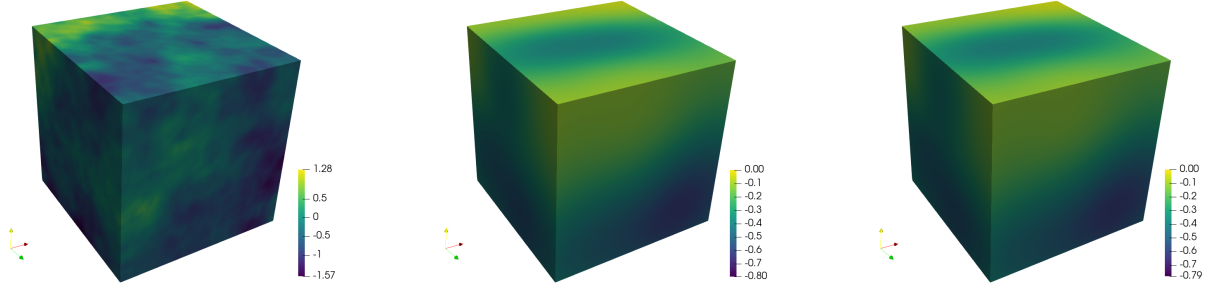


Figure 16: Prior sample (left), MAP points computed by DINO (middle) and FEM (right).

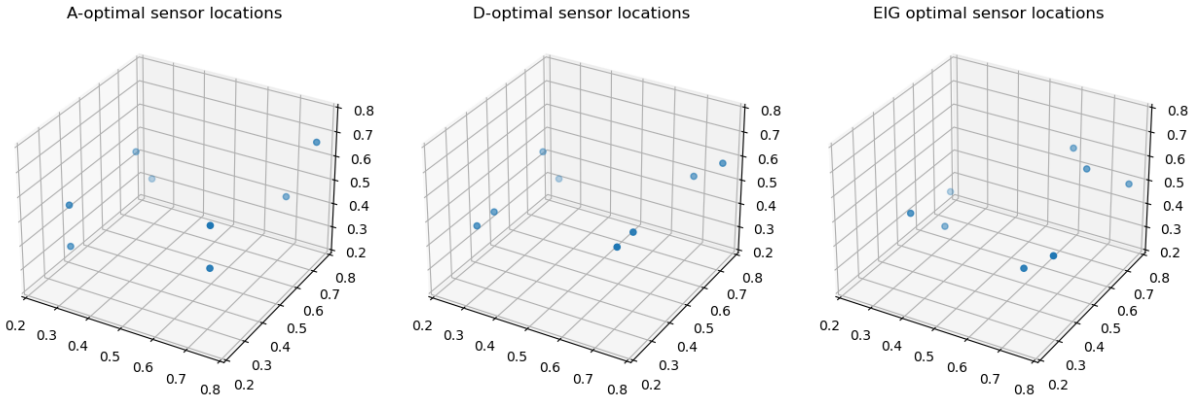


Figure 17: Eight optimal sensor locations selected using DINO by the swapping greedy algorithm 1 with the optimality criteria of A-optimality (left), D-optimality (middle), and EIG-optimality (right).

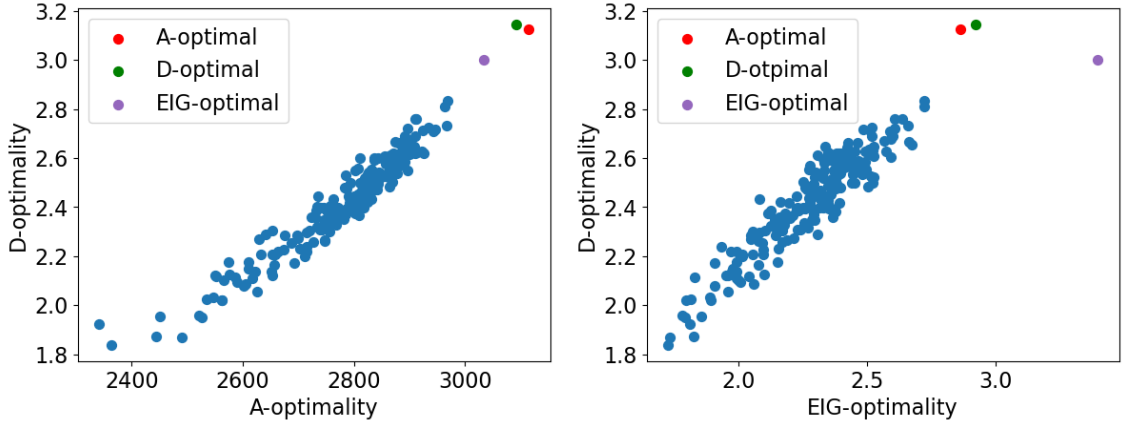


Figure 18: Optimality values at the optimal sensors selected by the swapping greedy algorithm 1 using different optimality criteria compared to those at 200 random designs. The optimality values are efficiently computed using DINO.

problems. We compare these costs to those of the high-fidelity computations. We use the CPU of AMD EPYC 7543 with 1 TB memory for high-fidelity computation and neural network evaluation. For the training of the neural network surrogate, we use the GPU of NVIDIA RTX A6000 with 48 GB memory.

In Table 4, we report the numbers for the two (2D and 3D) CDR problems corresponding to the computational complexity analysis in Section 4.5. In particular, for the computation of the MAP point, we use

an inexact Newton-CG optimizer from the HIPPLYlib library for the high-fidelity approximation (averaged 5 times) and an LBFGS optimizer from PyTorch for the neural network approximation.

Problem	N_s	k_s	r_s	d_s	N_{nt}	N_{cg}	$d_{\mathbf{m}}$	$r_{\mathbf{m}}$	r_F	N_t	N_b
2D CDR	128	2	10	100	5.6	4.6	16,641	128	30	8,192	100
3D CDR	128	2	8	512	4.2	2.2	35,937	256	128	2,048	100

Table 4: N_s : # SAA samples, k_s : # swapping loops, r_s : # sensors to select, d_s : # candidate sensors, N_{nt} : averaged # Newton iterations, N_{cg} : averaged # CG iterations per Newton iteration, $d_{\mathbf{m}}$: # discretized parameters, $r_{\mathbf{m}}$: # input project bases, r_F : # output projection bases, N_t : # training samples, N_b : # BFGS iterations.

In Table 5, we report the computational time (averaged over 5 times) by FEM using FEniCS v.s. by DINO using PyTorch for the computation of the PtO map, the MAP point by solving the optimization problem (19) v.s. (41), and the eigenvalue decomposition (22) v.s. (43). A speedup from hundreds for the 2D problem to thousands for the 3D problem can be observed. As the evaluation of the optimality criteria is dominated by one MAP optimization and one eigenvalue decomposition for each sample, the combined speedup of the DINO compared to FEM is about $(5.1 + 0.7)/(0.04 + 0.007) = 123$ and $(128.7 + 10.0)/(0.04 + 0.02) = 2,312$ for the 2D and 3D problems, respectively.

2D-CDR	PtO	MAP	Eigenpairs	3D-CDR	PtO	MAP	Eigenpairs
FEM	0.8	5.1	0.7	FEM	25.5	128.7	10.0
DINO	0.002	0.04	0.007	DINO	0.003	0.04	0.02
Speedup	400	127	100	Speedup	8,500	3,218	500

Table 5: Time (in seconds) and speedup by DINO v.s. FEM for the computation of the PtO map, the MAP point, and the eigenvalue decomposition, averaged over 5 times.

In Table 6, we report the offline computational time in (1) computing the (DIS and PCA) projection bases for input and output dimension reduction, (2) computing the training data of the PtO maps and (3) their Jacobians, and (4) training the neural networks. We note that the cost of computing the additional reduced Jacobian data is smaller than that of the PtO map data, as the reduced Jacobian computation is amortized by a direct solver using LU factorization (0.04 seconds for 2D CDR and 2.38 seconds for 3D CDR) and its repeated use in solving $r_t = \min(r_{\mathbf{m}}, r_F)$ linearized PDEs (each takes $0.0008 \ll 0.04$ seconds for 2D CDR and $0.02 \ll 2.38$ seconds for 3D CDR). The training of the neural networks takes much less time (145 seconds for 2D CDR and 413 seconds for 3D CDR in GPU) than that of generating the training data.

Problem	Bases	PtO	Jacobian	Total	Train (GPU)
2D-CDR	437	6,554	492	7,482	145
3D-CDR	31,241	52,224	10,076	93,541	413

Table 6: Offline time (in seconds) in computing the input and output projection bases, PtO maps, Jacobians, and training (GPU) the neural networks, averaged over 5 times.

In solving the Bayesian OED problem constrained by the 2D CDR equation, it takes 2.1 CPU hours to generate the projection bases and the training data of the PtO map and its Jacobian, and an additional 0.04 GPU hours for training the neural networks. Then it takes 3.8 CPU hours using DINO to run the swapping optimization algorithm with 293,120 evaluations of the optimality criteria. In contrast, it would take 472.2 CPU hours using the high-fidelity model to solve the Bayesian OED problem. The speedup in CPU hours by the DINO surrogate (including both offline and online time) over the high-fidelity evaluation is 80.0 $(472.2/(3.8+2.1))$. In the case of the 3D CDR, the speedup in CPU hours by DINO surrogate over the high-fidelity model with 1,552,896 evaluations of the optimality criteria is 1148.3 $(59,829.6/(26.1 + 26.0))$. We remark that to further speed up the computation, we have implemented both the data generation and the optimality criteria evaluation in parallel, using 128 CPU processors.

6. Conclusion

In this work, we developed an accurate, scalable, and efficient computational framework based on DINO to solve Bayesian OED problems constrained by PDEs with infinite-dimensional parameter input. We considered the three different optimality criteria of A-/D-optimality and EIG, which all require the computation of the Jacobian with Laplacian and low-rank approximations of the posterior. We employed DINO with derivative-informed dimension reduction to achieve high accuracy and scalability of the approximations for both the PtO map and its Jacobian. We derived efficient formulations for computing the MAP point and eigenvalue decomposition in the reduced dimension. To solve the optimization problem of the Bayesian OED, we proposed a modified swapping greedy algorithm initialized by a greedy algorithm.

With two numerical examples of linear diffusion and nonlinear CDR problems in both two and three dimensional physical domains, we demonstrated the accuracy, scalability, and efficiency of the proposed method. Specifically, we demonstrated that DINO provides more accurate approximations of the PtO map, the reduced Jacobian, and the MAP point than other neural networks trained without Jacobian. This results in significantly higher approximation accuracy for all optimality criteria in Bayesian OED. We demonstrated that the accuracy of DINO approximations with proper dimension reduction is scalable with respect to increasing parameter dimensions for the same number of training samples, or the same number of PDE solves. We also demonstrated that the proposed method achieved high efficiency with a speedup of $80\times$ in 2D and $1148\times$ in 3D, including both the offline data generation and online evaluation time, compared to the high-fidelity approximations for the CDR problems.

The proposed computational framework is applied to using Laplace and low-rank approximations of the posterior distribution for Bayesian OED in this work. It is limited to that Laplace approximation is a good approximation of the posterior and the decay of the eigenvalues of the Hessian is fast. In future work, we plan to address these limitations by developing the DINO surrogates in the context of variational approximation of the posterior distribution that can be highly non-Gaussian, and by using nonlinear dimension reduction and hierarchical approximation of the Hessian. We also plan to extend the computational framework to solve sequential Bayesian OED problems with efficient construction of time-dependent DINO surrogates.

Acknowledgments

This work is partially funded by National Science Foundation under grants #2245674, #2325631, #2245111, and #2233032. The first author acknowledges travel support from the Society for Industrial and Applied Mathematics. We thank helpful discussions with Thomas O’Leary-Roseberry, Dingcheng Luo, and Grant Bruer.

References

- [1] A. C. Atkinson, *Optimum Experimental Design*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 1037–1039. doi:10.1007/978-3-642-04898-2_434. URL https://doi.org/10.1007/978-3-642-04898-2_434
- [2] F. Pukelsheim, *Optimal design of experiments*, SIAM, 2006.
- [3] A. Alexanderian, N. Petra, G. Stadler, O. Ghattas, A-optimal design of experiments for infinite-dimensional Bayesian linear inverse problems with regularized ℓ_0 -sparsification, *SIAM Journal on Scientific Computing* 36 (5) (2014) A2122–A2148. doi:10.1137/130933381.
- [4] X. Huan, Y. M. Marzouk, Simulation-based optimal Bayesian experimental design for nonlinear systems, *Journal of Computational Physics* 232 (1) (2013) 288–317. doi:http://dx.doi.org/10.1016/j.jcp.2012.08.013. URL <http://www.sciencedirect.com/science/article/pii/S0021999112004597>
- [5] K. Wu, P. Chen, O. Ghattas, A fast and scalable computational framework for large-scale and high-dimensional Bayesian optimal experimental design, *SIAM/ASA Journal on Uncertainty Quantification* 11 (1) (2023) 235–261.
- [6] A. Alexanderian, Optimal experimental design for infinite-dimensional Bayesian inverse problems governed by PDEs: A review, *Inverse Problems* 37 (4) (2021) 043001.
- [7] A. Alexanderian, P. J. Gloor, O. Ghattas, On Bayesian A- and D-optimal experimental designs in infinite dimensions, *Bayesian Analysis* 11 (3) (2016) 671–695. doi:10.1214/15-BA969.
- [8] A. Alexanderian, A. K. Saibaba, Efficient D-optimal design of experiments for infinite-dimensional Bayesian linear inverse problems, *SIAM Journal on Scientific Computing* 40 (5) (2018) A2956–A2985. doi:10.1137/17M115712X.

- [9] J. Beck, B. M. Dia, L. F. Espath, Q. Long, R. Tempone, Fast bayesian experimental design: Laplace-based importance sampling for the expected information gain, *Computer Methods in Applied Mechanics and Engineering* 334 (2018) 523 – 553. doi:<https://doi.org/10.1016/j.cma.2018.01.053>.
URL <http://www.sciencedirect.com/science/article/pii/S0045782518300616>
- [10] K. Wu, P. Chen, O. Ghattas, An offline-online decomposition method for efficient linear Bayesian goal-oriented optimal experimental design: Application to optimal sensor placement, *SIAM Journal on Scientific Computing* 45 (1) (2023) B57–B77.
- [11] K. Wu, T. O’Leary-Roseberry, P. Chen, O. Ghattas, Large-scale Bayesian optimal experimental design with derivative-informed projected neural network, *Journal of Scientific Computing* 95 (1) (2023) 30.
- [12] X. Huan, Y. M. Marzouk, Gradient-based stochastic optimization methods in Bayesian experimental design, *International Journal for Uncertainty Quantification* 4 (6) (2014) 479–510.
- [13] A. Alexanderian, N. Petra, G. Stadler, O. Ghattas, A fast and scalable method for A-optimal design of experiments for infinite-dimensional Bayesian nonlinear inverse problems, *SIAM Journal on Scientific Computing* 38 (1) (2016) A243–A272. doi:[10.1137/140992564](https://doi.org/10.1137/140992564).
- [14] Q. Long, M. Scavino, R. Tempone, S. Wang, Fast estimation of expected information gains for Bayesian experimental designs based on Laplace approximations, *Computer Methods in Applied Mechanics and Engineering* 259 (2013) 24–39.
- [15] Q. Long, M. Motamed, R. Tempone, Fast Bayesian optimal experimental design for seismic source inversion, *Computer Methods in Applied Mechanics and Engineering* 291 (2015) 123 – 145. doi:<https://doi.org/10.1016/j.cma.2015.03.021>.
URL <http://www.sciencedirect.com/science/article/pii/S0045782515001310>
- [16] J. Beck, B. Mansour Dia, L. Espath, R. Tempone, Multilevel double loop Monte Carlo and stochastic collocation methods with importance sampling for Bayesian optimal experimental design, *International Journal for Numerical Methods in Engineering* 121 (15) (2020) 3482–3503.
- [17] A. K. Saibaba, A. Alexanderian, I. C. Ipsen, Randomized matrix-free trace and log-determinant estimators, *Numerische Mathematik* 137 (2) (2017) 353–395.
- [18] B. Crestel, A. Alexanderian, G. Stadler, O. Ghattas, A-optimal encoding weights for nonlinear inverse problems, with application to the Helmholtz inverse problem, *Inverse Problems* 33 (7) (2017) 074008.
URL <http://iopscience.iop.org/10.1088/1361-6420/aa6d8e>
- [19] A. Attia, A. Alexanderian, A. K. Saibaba, Goal-oriented optimal design of experiments for large-scale Bayesian linear inverse problems, *Inverse Problems* 34 (9) (2018) 095009.
- [20] N. Aretz-Nellesen, P. Chen, M. A. Grepl, K. Veroy, A-optimal experimental design for hyper-parameterized linear Bayesian inverse problems, *Numerical Mathematics and Advanced Applications ENUMATH 2020* (2020).
- [21] N. Aretz, P. Chen, K. Veroy, Sensor selection for hyper-parameterized linear Bayesian inverse problems, *PAMM* 20 (S1) (2021) e202000357.
- [22] N. Aretz, P. Chen, D. Degen, K. Veroy, A greedy sensor selection algorithm for hyperparameterized linear bayesian inverse problems with correlated noise models, *Journal of Computational Physics* 498 (2024) 112599.
- [23] A. Foster, M. Jankowiak, E. Bingham, P. Horsfall, Y. W. Teh, T. Rainforth, N. Goodman, Variational Bayesian optimal experimental design, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Vol. 32, Curran Associates, Inc., 2019, pp. 14036–14047.
URL <https://proceedings.neurips.cc/paper/2019/file/d55cbf210f175f4a37916eafe6c04f0d-Paper.pdf>
- [24] S. Kleinegesse, M. U. Gutmann, Bayesian experimental design for implicit models by mutual information neural estimation, in: *International Conference on Machine Learning*, PMLR, 2020, pp. 5316–5326.
- [25] J. Go, T. Isaac, Robust expected information gain for optimal bayesian experimental design using ambiguity sets, in: *Uncertainty in Artificial Intelligence*, PMLR, 2022, pp. 728–737.
- [26] W. Shen, X. Huan, Bayesian sequential optimal experimental design for nonlinear models using policy gradient reinforcement learning, *Computer Methods in Applied Mechanics and Engineering* 416 (2023) 116304.
- [27] J. Jagalur-Mohan, Y. Marzouk, Batch greedy maximization of non-submodular functions: Guarantees and applications to experimental design, *Journal of Machine Learning Research* 22 (252) (2021) 1–62.
- [28] T. Helin, N. Hyvonen, J.-P. Puska, Edge-promoting adaptive Bayesian experimental design for X-ray imaging, *SIAM Journal on Scientific Computing* 44 (3) (2022) B506–B530.
- [29] T. O’Leary-Roseberry, P. Chen, U. Villa, O. Ghattas, Derivative-informed neural operator: an efficient framework for high-dimensional parametric derivative learning, *Journal of Computational Physics* 496 (2024) 112555.
- [30] T. Bui-Thanh, C. Burstedde, O. Ghattas, J. Martin, G. Stadler, L. C. Wilcox, *Extreme-scale UQ for Bayesian inverse problems governed by PDEs*, in: *SC12: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2012.
- [31] T. Bui-Thanh, O. Ghattas, J. Martin, G. Stadler, A computational framework for infinite-dimensional Bayesian inverse problems Part I: The linearized case, with application to global seismic inversion, *SIAM Journal on Scientific Computing* 35 (6) (2013) A2494–A2523. doi:[10.1137/12089586X](https://doi.org/10.1137/12089586X).
- [32] P. Chen, U. Villa, O. Ghattas, Hessian-based adaptive sparse quadrature for infinite-dimensional Bayesian inverse problems, *Computer Methods in Applied Mechanics and Engineering* 327 (2017) 147–172.
URL <https://doi.org/10.1016/j.cma.2017.08.016>
- [33] P. Chen, K. Wu, J. Chen, T. O’Leary-Roseberry, O. Ghattas, Projected Stein variational Newton: A fast and scalable Bayesian inference method in high dimensions, *Advances in Neural Information Processing Systems* (2019).
- [34] P. Chen, K. Wu, O. Ghattas, Bayesian inference of heterogeneous epidemic models: Application to COVID-19 spread accounting for long-term care facilities, *Computer Methods in Applied Mechanics and Engineering* 385 (2021) 114020.
- [35] P. Chen, O. Ghattas, Projected Stein variational gradient descent, in: *Advances in Neural Information Processing Systems*,

2020.

- [36] Y. Wang, P. Chen, W. Li, Projected Wasserstein gradient descent for high-dimensional Bayesian inference, *SIAM/ASA Journal on Uncertainty Quantification* 10 (4) (2022) 1513–1532.
- [37] O. Bashir, K. Willcox, O. Ghattas, B. van Bloemen Waanders, J. Hill, Hessian-based model reduction for large-scale systems with initial condition inputs, *International Journal for Numerical Methods in Engineering* 73 (2008) 844–868.
- [38] P. Chen, O. Ghattas, Hessian-based sampling for high-dimensional model reduction, *International Journal for Uncertainty Quantification* 9 (2) (2019).
- [39] N. Alger, P. Chen, O. Ghattas, Tensor train construction from tensor actions, with application to compression of large high order derivative tensors, *SIAM Journal on Scientific Computing* 42 (5) (2020) A3516–A3539.
- [40] T. O’Leary-Roseberry, U. Villa, P. Chen, O. Ghattas, Derivative-informed projected neural networks for high-dimensional parametric maps governed by PDEs, *Computer Methods in Applied Mechanics and Engineering* 388 (2022) 114199.
- [41] A. Alexanderian, N. Petra, G. Stadler, O. Ghattas, Mean-variance risk-averse optimal control of systems governed by PDEs with random parameter fields using quadratic approximations, *SIAM/ASA Journal on Uncertainty Quantification* 5 (1) (2017) 1166–1192. doi:10.1137/16M106306X.
- [42] P. Chen, U. Villa, O. Ghattas, Taylor approximation and variance reduction for PDE-constrained optimal control under uncertainty, *Journal of Computational Physics* 385 (2019) 163–186.
URL <https://arxiv.org/abs/1804.04301>
- [43] P. Chen, M. Haberman, O. Ghattas, Optimal design of acoustic metamaterial cloaks under uncertainty, *Journal of Computational Physics* 431 (2021) 110114.
- [44] D. Luo, T. O’Leary-Roseberry, P. Chen, O. Ghattas, Efficient PDE-constrained optimization under high-dimensional uncertainty using derivative-informed neural operators, *arXiv preprint arXiv:2305.20053* (2023).
- [45] U. Villa, N. Petra, O. Ghattas, HIPPLYlib: An Extensible Software Framework for Large-Scale Inverse Problems Governed by PDEs: Part I: Deterministic Inversion and Linearized Bayesian Inference, *ACM Transactions on Mathematical Software* 47 (2) (Apr. 2021). doi:10.1145/3428447.
- [46] A. M. Stuart, Inverse problems: a Bayesian perspective, *Acta numerica* 19 (2010) 451–559.
- [47] N. N. Schraudolph, Fast curvature matrix-vector products for second-order gradient descent, *Neural computation* 14 (7) (2002) 1723–1738.
- [48] T. Isaac, N. Petra, G. Stadler, O. Ghattas, Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the Antarctic ice sheet, *Journal of Computational Physics* 296 (2015) 348–368. doi:10.1016/j.jcp.2015.04.047.
- [49] O. Zahm, P. G. Constantine, C. Prieur, Y. M. Marzouk, Gradient-based dimension reduction of multivariate vector-valued functions, *SIAM Journal on Scientific Computing* 42 (1) (2020) A534–A558.
- [50] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [51] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, in: *NIPS-W*, 2017.
- [52] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- [53] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, G. N. Wells, The fenics project version 1.5, *Archive of numerical software* 3 (100) (2015).
- [54] U. Villa, N. Petra, O. Ghattas, hIPPYlib: An extensible software framework for large-scale inverse problems governed by PDEs: Part I: Deterministic inversion and linearized Bayesian inference, *ACM Transactions on Mathematical Software (TOMS)* 47 (2) (2021) 1–34.
- [55] T. O’Leary-Roseberry, U. Villa, hIPPYflow: Dimension reduced surrogate construction for parametric PDE maps in Python (2021).