PIED: Physics-Informed Experimental Design for Inverse Problems

Apivich Hemachandra [*1] Gregory Kang Ruey Lau [*1 2] See-Kiong Ng [1] Bryan Low Kian Hsiang [1]

Abstract

In many inverse problems (IPs) in science and engineering, optimization of design parameters (e.g., sensor placement) with experimental design (ED) methods is performed due to high data acquisition costs when conducting physical experiments, and often has to be done up front due to practical constraints on sensor deployments. However, existing ED methods are often challenging to use in practical PDE-based inverse problems due to significant computational bottlenecks during forward simulation and inverse parameter estimation. This paper presents Physics-Informed Experimental Design (PIED), the first ED framework that makes use of PINNs in a fully differentiable architecture to perform continuous optimization of design parameters for IPs. PIED utilizes techniques such as learning of a shared NN parameter initialization, and approximation of PINN training dynamics during the ED process, for better estimation of the inverse parameters. PIED selects the optimal design parameters for one-shot deployment, allows exploitation of parallel computation unlike existing methods, and is empirically shown to significantly outperform existing ED benchmarks in IPs for both finite-dimensional and function-valued inverse parameters given limited budget for observations.

1. Introduction

Inverse problems (IP), where the goal is to estimate unknown latent parameters given the model dynamics and some observation data (Vogel, 2002; Ghattas and Willcox, 2021), are key challenges in many science and engineering disciplines such as classical mechanics (Tanaka and Bui, 1993; Gazzola et al., 2018), quantum mechanics (Chadan et al., 1989) or geophysics (Smith et al., 2021; Waheed et al., 2021). In these settings, we typically have rich domain knowledge on the model dynamics, such as the form of governing partial differential equations (PDEs). Such knowledge or equations typically allows us to develop *forward simulations* to predict observation measurements, but it is still challenging to directly solve the IP of inferring key parameters of these equations from observation data. This is especially so given that data acquisition (e.g., measurements from experiments or field trials) is often very costly in these settings, making the choice of *what data to collect given a limited budget* critical to solving these problems in practice.

Experimental design (ED) methods aim to tackle the data scarcity problem by optimizing design parameters, such as sensor placement locations, to yield the most informative measurements for estimating the unknown inverse parameters (Razavy, 2020; Alexanderian, 2021). These methods typically make use of several forward simulations based on guesses of the true inverse parameter to find the best design parameter for the measurements. However, these methods have been challenging to use in practical PDE-informed IPs due to significant computational bottlenecks in both forward simulations and parameter optimization, especially for systems with complex forward models and PDEs. In particular, simulators using conventional PDE solvers often are costly to compute, and returns discretized (mesh-based) approximate solutions whose derivatives are difficult to compute, preventing the use of continuous optimization methods for ED.

Physics-Informed Neural Networks (PINNs) are neural networks that incorporate PDEs and their initial/boundary conditions (IC/BCs) into the NN loss function (Raissi et al., 2019), and have been successfully applied to various problems. PINNs are especially well-suited to tackle experimental design for IPs, as they (1) allow easy incorporation of observation data into the inverse problem solver through the training loss function, (2) can be used for both running forward simulations and directly solving IPs with the same model architecture, (3) are continuous and differentiable w.r.t. the function inputs, and (4) can amortize training cost, e.g., through transfer learning. However, to our knowledge, there has not been an ED framework for IPs that fully utilizes these advantages from PINNs for ED problems.

[*]Equal contribution [1]Department of Computer Science, National University of Singapore, Singapore 117417 [2]CNRS@CREATE, 1 Create Way, 08-01 Create Tower, Singapore 138602. Correspondence to: Bryan Low Kian Hsiang <lowkh@comp.nus.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

In this paper, we present Physics-Informed Experimental Design (PIED), the first ED framework that makes use of PINNs in a fully differentiable architecture to perform continuous optimization of design parameters for IPs for one-shot deployment. Our contributions are summarized as follows.

- We propose a novel ED framework comprising of different components, such as the use of PINNs as both forward simulators and inverse solvers, the parameterization of sensor placements for constrained configurations, and an ED criteria for evaluating the performance of the inverse solver (Sec. 3). These components enable efficient parallel computation and gradient-based optimization methods in the continuous optimization of observation inputs.
- We introduce the use of a learned initial NN parameter which are used for all PINNs in PIED (Sec. 4.1), based on first-order meta-learning methods, allowing for more efficient PINN training over multiple PDE inverse parameters (Sec. 5.1).
- We present effective ED criteria based on novel methods such as empirical Neural Tangent Kernel regression for inverse PINNs and analysis based on the training dynamics of PINN-based inverse solvers (Sec. 4.2).
- We empirically demonstrate that PIED is able to outperform other ED methods on inverse problems (Sec. 5.2), both in the case where the inverse parameters of interest are finite-dimensional and when they are unknown functions.

2. Background and Related Works

Problem setup. Consider a system described by a set of PDEs of the form

$$\mathcal{D}[u,\beta](x) = f(x) \qquad \forall x \in \mathcal{X}, \text{ and} \tag{1}$$

$$\mathcal{B}[u,\beta](x') = g(x') \qquad \forall x' \in \partial\mathcal{X}. \tag{2}$$

where $u : \mathcal{X} \to \mathbb{R}^{d_{\text{out}}}$ describes the observable function (solution of the PDE) over a coordinate variable $x \in \mathcal{X} \subset \mathbb{R}^{d_{\text{in}}}$ (where time could be a subcomponent), and $\beta \in \mathbb{R}^{d_{\text{inv}}}$ is the PDE inverse parameter[1]. $\mathcal{D}$ is a PDE operator and $\mathcal{B}$ is an operator for the initial/boundary conditions (IC/BCs) at boundary $\partial\mathcal{X} \subset \mathcal{X}$[2]. Different inverse parameters β results in different observable function u which satisfies (1), which for convenience will be denoted by u_β.

[1]For simplicity we assume β is finite-dimensional. In our experiments, we demonstrate how our method can also be extended to cases where β is a function.

[2]Examples of PDEs, specifically those in our experiments, can be found in App. C.1.

In the inverse problem (IP), we assume the operators $\mathcal{D}$ and $\mathcal{B}$, and functions f and g are known. However, the true *inverse parameter* of interest, β_0, is unknown and cannot be observed directly. We can make measurements at M observation points[3] $X = \{x_j\}_{j=1}^M \subset \mathcal{X}$ to get observation values $Y = \{u_{\beta_0}(x_j) + \varepsilon_j\}_{j=1}^M$, which corresponds to noisy observations of u_{β_0} evaluated at points X where we assume Gaussian noise $\varepsilon_j \sim \mathcal{N}(0,\sigma^2)$. In general, X could possibly be constrained to a set of feasible configurations $\mathcal{S} \subseteq \mathcal{X}^M$. To solve the IP, we could use an *inverse solver* that finds the inverse parameter $\hat{\beta}$ that fits best with the observed data (X, Y) where

$$\hat{\beta}(X,Y) \approx \arg\min_{\beta} \|u_\beta(X) - Y\|^2. \tag{3}$$

Unfortunately, the observations Y are typically expensive to obtain due to costly sensors or operations. In many settings, the observation points also have to be chosen in a one-shot rather than in a sequential, adaptive manner due to the costs of reinstalling sensors and inability to readjust the design parameters on-the-fly. Hence, the observation points X should be carefully chosen before actual measurements are made. As we *do not* know the true inverse parameter β_0, good observation points should maximize the average performance of the inverse solver over its distribution $p(\beta)$. In the experimental design (ED) problem, the goal is therefore to select the observation input $X \in \mathcal{S}$ which minimizes

$$L(X) = \mathbb{E}_{\beta, Y \sim p(\beta)p(Y|\beta)}\left[\left\|\hat{\beta}(X,Y) - \beta\right\|^2\right], \tag{4}$$

or the *expected* error of the estimated inverse parameter w.r.t. the possible true inverse parameters.

Related works. Existing ED methods in the literature include those that adopts a Bayesian framework (Chaloner and Verdinelli, 1995; Long et al., 2013; Belghazi et al., 2018; Foster et al., 2019) and those that aims to construct a policy for choosing the optimal experimental design (Ivanova et al., 2021; Lim et al., 2022). These methods are typically designed for the *adaptive* setting, where multiple rounds of observations with different design parameters are allowed. In each round, the design parameter can be adjusted based on the data that has already been observed.

However, there are many practical scenarios where *non-adaptive* ED are desirable. For instance, field scientists who may incur significant cost in planning, deployment, and collection of results for each iteration may strongly prefer to perform a single round of sensor placement and measurements as opposed to sequentially making few measurements per round and frequently adjusting sensor placement locations. Another example is when the phenomena being

[3]We abuse notations by allowing the set of X to be an argument for functions which takes in $x \in \mathcal{X}$ as well.

observed occurs in a single time period, and all observations have to be decided beforehand. In these cases, it is more practical to optimize for a single design parameter upfront for collecting all observation data needed for the IP in one-shot. Unlike past works, our work optimizes on one-shot deployment.

Additionally, many ED methods assume that the observation generation model exists and can be efficiently queried from. This is not the case in our problem setting where the observation is governed by a PDE (1) which may not be easy nor efficient to solve. While there are proposed ED methods which are specific for solving PDE-based IPs (Ghattas and Willcox, 2021; Alexanderian, 2021; Alexanderian et al., 2022), they typically rely on numerical solvers, which can be computationally expensive for forward simulations and inefficient for solving IPs. Furthermore, numerical solvers often require input discretization and cannot be easily differentiated, which makes optimization problems involving numerical solvers more complex. Further material on related works are in App. B.

Physics-informed neural networks. In our work, we will use physics-informed neural networks (PINNs) (Raissi et al., 2019) to simulate PDEs and solve IPs. PINNs are neural networks (NNs) $\hat{u}_\theta$ that approximates the solution u to (1) by minimizing the composite loss[4]

$$\mathcal{L}(\theta, \beta; X, Y) = \mathcal{L}_{\text{obs}}(\theta; X, Y) + \mathcal{L}_{\text{PDE}}(\theta, \beta), \tag{5}$$

where

$$\mathcal{L}_{\text{obs}}(\theta; X, Y) = (2|X|)^{-1} \left\| \hat{u}_\theta(X) - Y \right\|_2^2 \tag{6}$$

$$\begin{aligned} \mathcal{L}_{\text{PDE}}(\theta, \beta) &= (2|X_p|)^{-1} \left\| \mathcal{D}[\hat{u}_\theta, \beta](X_p) - f(X_p) \right\|_2^2 \\ &\quad + (2|X_b|)^{-1} \left\| \mathcal{B}[\hat{u}_\theta, \beta](X_b) - g(X_b) \right\|_2^2, \end{aligned} \tag{7}$$

(X, Y) are observation data if available, and $X_p \subset \mathcal{X}$ and $X_b \subset \partial\mathcal{X}$ are collocation points to enforce the PDE and IC/BC constraints respectively. PINNs can be used both as forward simulators when β is known but no observation data is available ($\mathcal{L}_{\text{obs}}(\theta; X, Y) = 0$), or as inverse solvers when observation data is available but β is unknown. For the latter, $\hat{\beta}$ is learned jointly with NN parameters θ during training.

3. Experimental Design Loop

In this section, we provide an overview of our framework, PIED, and its key components. Our framework aims to fully utilize the advantages provided by PINNs for ED problems, and should allow efficient parallel computation and differentiability for gradient-based optimization.

3.1. Components Within the Experimental Design Loop

Intuitively, to solve our ED problem given the loss in (4), we could make use of PINNs, which could perform as both forward simulator and inverse solver using the same architecture, i.e., mapping from β to u_β and then reconstructing an estimate $\hat{\beta}$. The choice of observation points could then be viewed as an information bottleneck, where we are querying a relatively low-dimensional representation of the underlying function u_β to pass to the inverse solver. Our goal would then be to choose the best points such that reconstruction error is minimized across possible β values.

To achieve this, we propose an ED framework which is visualized in Fig. 1. We consider N *parallel threads*, each representing sampled inverse parameter values (reference β) from a prior distribution based on domain knowledge. Within each thread, we would have (1) a *forward simulator* to return an observable function $\tilde{u}_{\beta_i}$ which approximates the PDE solution u_{β_i} with inverse parameter β_i, (2) *observation selector* which queries the approximated function $\tilde{u}_{\beta_i}(X)$ at observation points X (consistent across all threads), and (3) *inverse solver* an algorithm that takes in the queried points to produce estimates of the inverse parameter $\hat{\beta}_i$. Finally, we require (4) a criterion that captures how good the estimates are across all threads, and an *input optimization* method to select the single best observation input according to the criterion. We will first describe these components, before discussing how to efficiently compute them in Sec. 4.

PINN-based forward simulator (F). The first component involves training parallelized PINNs (forward PINNs) to estimate u_β for each of the N reference β threads i as mentioned above. For each β_i, a forward PINN is trained with loss $\mathcal{L}_{\text{PDE}}(\theta, \beta_i)$ from (5) to generate $\tilde{u}_{\beta_i}$, an estimate of u_{β_i} over the input space $\mathcal{X}$, i.e., for each $i = 1, \ldots, N$,

$$\beta_i \xrightarrow{\text{forward simulator } \mathbf{F}} \tilde{u}_{\beta_i} = \hat{u}_{\theta_i} \approx u_{\beta_i}. \tag{8}$$

Note that the trained PINNs $\tilde{u}_{\beta_i}$ represent learned *functions* that are meshless, and therefore can be queried at and even differentiated w.r.t. any input $x \in \mathcal{X}$. This is unlike traditional simulators based on numerical integrators which would require discretization of the input space $\mathcal{X}$.

While it may seem like the parallelized PINNs would be computationally expensive, in practice this would not be the case. First, the parameters of PINNs can be initialized from some pre-trained base model or even from parameters of PINNs trained on other values of β, which can reduce the amount of time required for convergence of the PINN during training. Second, modern software and hardware allows

[4]For simplicity we consider one IC/BC, however the loss can be generalized to include multiple IC/BCs by adding similar loss terms for each constraint.

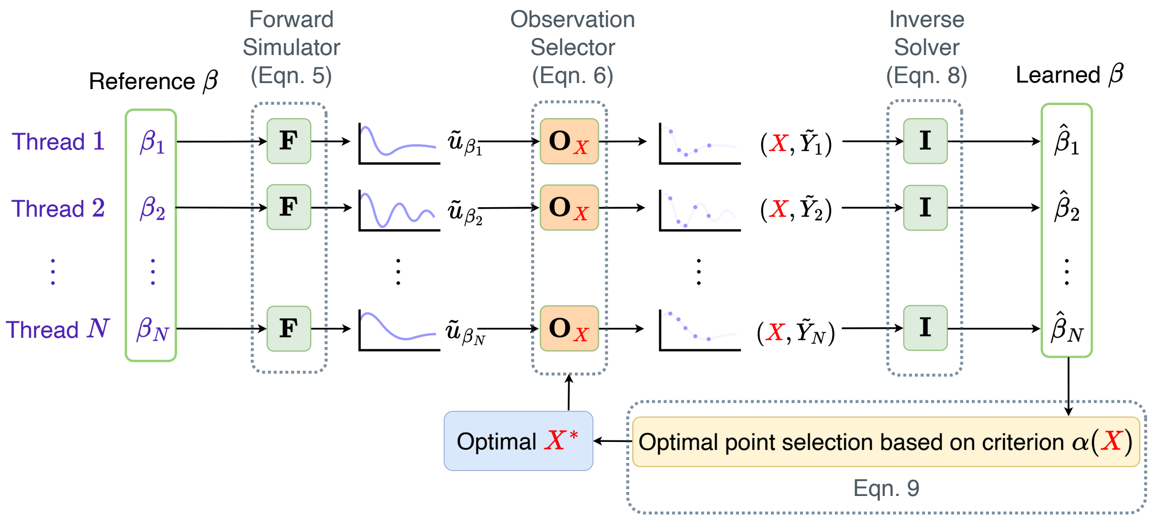

Figure 1. Overview of the various components composed in PIED for solving the ED problem of observation input optimization for IPs. This includes the PINN-based forward simulator, the observation selector, the PINN-based inverse solver, and the ED criterion $\hat{\alpha}_i$ and its optimization.

for very efficient training of multiple NNs in parallel (for example, via the `vmap` function on JAX). Third, we have additional training and approximation techniques (described in Sec. 4) that makes use of the trained forward simulators to significantly reduce the inverse solver costs.

Observation selector ($\mathbf{O}_X$). Given the trained forward PINNs $\{\tilde{u}_{\beta_i}\}_{i=1}^N$, the second component applies a "sieve" that queries all forward PINNs with the *same* M observation points $X = \{x_j\}_{j=1}^M \in \mathcal{S}$ to produce the respective sets of predicted observations, i.e., for each $i = 1, \dots, N$,

$$\tilde{u}_{\beta_i} \xrightarrow{\text{obs. selector } \mathbf{O}_X} (X, \tilde{Y}_i) = \Big(\{x_j\}_{j=1}^M, \{\tilde{u}_{\beta_i}(x_j)\}_{j=1}^M\Big). \tag{9}$$

In real ED problems, the observation points X often cannot be set arbitrarily, but instead is restricted due to operational constraints to some feasible set $\mathcal{S} \subset \mathcal{X}^M$. For example, the observations may be possible at any spatial location but must be made at specific time intervals, or must be placed in a regular grid. Hence, unlike many of the past ED works, we allow for both freely-chosen observation points and *constrained configurations* that can be parameterized by some design parameter $\gamma \in \mathcal{S}_\gamma \subset \mathbb{R}^d$ from a closed set where $d \leq M d_{\text{in}}$. Specifically, we consider $\mathcal{S}$ of the form

$$\mathcal{S} = \{X_\gamma : \gamma \in \mathcal{S}_\gamma\} \quad \text{where} \quad X_\gamma = \{x_{\gamma,j}\}_{j=1}^M. \tag{10}$$

Finding the optimal X for a criterion (elaborated below) can then be done by optimizing for γ, which is a continuous optimization problem in a convex domain. Examples of how various observation point configurations can be parameterized this way are in App. D.

PINN-based inverse solver (I). For each β_i, the predicted observation data $(X, \tilde{Y}_i)$ from (9) will then be used to train an inverse solver PINN (inverse PINN) with loss function $\mathcal{L}(\theta, \beta; X, \tilde{Y}_i)$ from (5) to return an estimated inverse parameter $\hat{\beta}_i$, i.e., for each $i = 1, \dots, N$,

$$(X, \tilde{Y}_i) \xrightarrow{\text{inverse solver } \mathbf{I}} \hat{\beta}_i. \tag{11}$$

Interestingly, in our ED framework, the forward and inverse PINNs have the same architecture. This enables us to develop effective approximation techniques based on predicting the dynamics of the inverse PINNs using its corresponding forward PINN, significantly reducing computational cost for the ED process. We elaborate further on these techniques in Sec. 4.

Criterion and optimal point selection. The components above allows us to directly consider the ED criterion in (4) that evaluates how well a chosen set X impacts the average performance of the inverse solver. Specifically, for a given X, our criterion evaluates how well the estimates $\hat{\beta}_i$ from the inverse solvers matches the corresponding reference parameter values β_i across all parallel threads i,

$$\alpha(X) = \frac{1}{N}\sum_{i=1}^N \alpha_i(X) = \frac{1}{N}\sum_{i=1}^N \Big[-\big\|\hat{\beta}_i(X, \tilde{Y}_i) - \beta_i\big\|^2 \Big]. \tag{12}$$

We can then construct an optimization loop where we search for the best observation points X that maximizes $\alpha(X)$. Due to the parameterization of X_γ and the differentiability of our inverse solver, we could back-propagate through α directly to compute $\nabla_\gamma \alpha(X_\gamma)$. This allows α to be optimized using gradient-based iterative optimization methods such as gradient descent, instead of methods such as Bayesian optimization which do not typically perform well on high-dimensional problems, or requires combinatorial optimization over discretized observation points.

3.2. Inference for Inverse Problem

After obtaining the optimal observation input configuration X^* from the ED loop, we can conduct actual experiments to obtain the true observations Y. We can proceed to train the inverse PINNs like before, but on the true observations Y, as opposed to the simulated observations, to obtain an estimate $\hat{\beta}^*$ of the true inverse parameter β, i.e.,

$$\left(X^*, Y\right) \xrightarrow{\text{inverse solver } \mathbf{I}} \hat{\beta}^*. \tag{13}$$

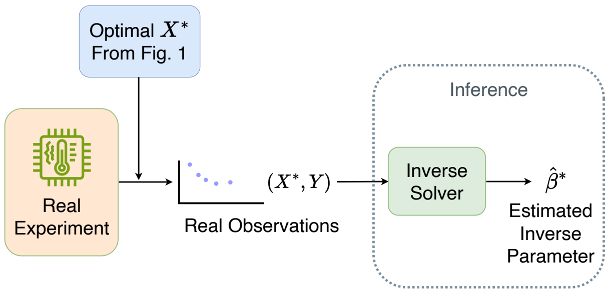

Figure 2. Inference stage for the IP using the inverse solver.

4. Practical Implementation of PIED

In this section, we introduce training and approximation methods which makes PIED more practical. Specifically, we adopt a meta-learning approach to learn a NN initialization for efficient fine-tuning of our forward and inverse PINNs across different threads (Sec. 4.1), and propose approximations of our criterion in (12) to effectively optimize for the observation points (Sec. 4.2).

4.1. Learning a Fixed Initialization for PINNs

While it may seem challenging to train multiple forward and inverse PINNs, each for different reference β, in practice we can efficiently achieve this with REPTILE (Nichol et al., 2018; Liu et al., 2021), a first-order meta-learning algorithm. Each β_i could be interpreted as a task, and our goal of using REPTILE is to learn a fixed PINN initialization, $\theta_{\text{L-Init}}$, that allows quick fine-tuning to many different values of β_i using fewer training steps. This improves the efficiency of (1) forward PINNs as we first train for $\theta_{\text{L-Init}}$ before fine-tuning $\theta_{\text{L-Init}}$ across different reference β, (2) inverse PINNs as we can re-use $\theta_{\text{L-Init}}$ for the approximation of inverse PINNs performance to optimize our ED criterion (elaborated in Sec. 4.2), and (3) inference as we could also re-use $\theta_{\text{L-Init}}$ for the final inverse PINN applied to actual experimental data to find the true $\hat{\beta}^*$. Furthermore, $\theta_{\text{L-Init}}$ can also be learned once and re-used many times on different IP instances with the same PDE setting, further reducing computational costs for practical applications. Using a fixed shared NN initialization $\theta_{\text{L-Init}}$ also results in more predictable PINN training dynamics and more stable forward simulator and inverse solver behaviors. We demonstrate these benefits empirically in Sec. 5.1, including indications that the PINN $\hat{u}_{\theta_{\text{L-Init}}}$ at initialization has some common structure shared among various different u_{β_i}.

4.2. Approximate Criteria for Performance of the Inverse Solver

Directly optimizing the criterion in (12) while fully training inverse solvers in all threads requires high computational resources as it involves differentiating over many inverse PINN training steps, causing memory issues when performing back-propagation. Hence, we consider several approaches to approximate the inverse solver performance and criterion in (12).

Few-step Inverse Solver Training (FIST) Criterion. First, we consider a relatively straightforward approach of reducing computational cost by partially training the inverse PINN for fewer training steps to get an intermediate estimate of $\hat{\beta}_i$ achievable at convergence – this will require fewer back-propagation steps when optimizing for the set of observation points X. Ideally, a set X that will eventually result in better $\hat{\beta}_i$ should also result in better intermediate estimates even with fewer training steps, and provide us with reliable signals for gradient descent optimization. In practice, we find that this is more likely to be true when the inverse PINN is closer to convergence, compared to the early stages of training when most X choices would also give good performance gains.

Hence, our method FIST initialize the inverse PINNs for each thread i with NN parameters that are perturbed from that of the corresponding converged forward PINN for β_i, making the inverse PINNs closer to convergence. With this initialization, we then perform partial training for r training steps (where $r \sim 10^2$ in our experiments) to obtain the estimate $\hat{\beta}_i$ which is then compared against the reference β_i to compute the criterion. We present the pseudocode for FIST in Alg. 1 of the Appendix. While FIST may outperform benchmarks in some settings as we show in Sec. 5, we found that its performance is sensitive to hyperparameters, making it less practical in more complex settings.

Model Training Estimate (MoTE) Criterion. A better approach may be to approximate the inverse solver output $\hat{\beta}_i$ at convergence instead of training it directly. To do so, we can perform kernel regression with the empirical Neural Tangent Kernel (eNTK) of the PINN (Jacot et al., 2018; Wang et al., 2020; Lau et al., 2024), given a set of observation data X. Specifically, under assumptions that the PINN is in the linearized regime and trained via gradient descent with (5), the inverse parameter $\hat{\beta}_i$ at convergence can be estimated by (IC/BC terms omitted for notational simplicity)

$$\hat{\beta}_i(X_\gamma, \tilde{Y}_\gamma) \approx \beta^{(0)} - J_\beta^\top \Theta_\gamma^{-1} R_\gamma \tag{14}$$

where

$$J_\beta = \begin{bmatrix} 0 & J_{p,\beta} \end{bmatrix}, \tag{15}$$

$$\Theta_\gamma = \begin{bmatrix} J_{\gamma,\theta} J_{\gamma,\theta}^\top & J_{\gamma,\theta} J_{p,\theta}^\top \\ J_{p,\theta} J_{\gamma,\theta}^\top & J_{p,\theta} J_{p,\theta}^\top + J_{p,\beta} J_{p,\beta}^\top \end{bmatrix}, \tag{16}$$

$$R_\gamma = \begin{bmatrix} \hat{u}_{\theta^{(0)}}(X_\gamma) - \tilde{Y}_\gamma \\ \mathcal{D}[\hat{u}_{\theta^{(0)}}, \beta^{(0)}](X_p) - f(X_p) \end{bmatrix}, \tag{17}$$

$(\theta^{(0)}, \hat{\beta}^{(0)})$ are the initialization parameters, $J_{\gamma,\theta} = \nabla_\theta \hat{u}_{\theta^{(0)}}(X_\gamma)$, $J_{p,\theta} = \nabla_\theta \mathcal{D}[\hat{u}_{\theta^{(0)}}, \beta](X_p)$, and $J_{p,\beta} = \nabla_\beta \mathcal{D}[\hat{u}_{\theta^{(0)}}, \beta^{(0)}](X_p)$ (note that $\nabla_\beta \hat{u}_{\theta^{(0)}}(X_\gamma) = 0$). This estimate for $\hat{\beta}$ in (14) is adapted from Lee et al. (2018), and is verified in App. E.2.1, which includes further details on the assumptions and derivation. Note that while the use of NTK in PINNs is not new (Wang et al., 2020; Lau et al., 2024), previous works have not utilized NTK to directly analyze the performance of PINNs in solving inverse problems as we have done.

In practice, as noted by Lau et al. (2024), finite-width PINNs have eNTKs that evolves over training before they can better reconstruct the true PDE solution. Hence, for the MoTE criterion, we first do r steps of training on the $\theta_{\text{L-Init}}$ initialized inverse PINN before computing the eNTK and performing kernel regression. We present the pseudocode for MoTE in Alg. 2 in the Appendix.

Tolerable Inverse Parameter (TIP) Criterion. Another method to reduce computational cost is to directly approximate (12) without explicitly approximating $\hat{\beta}_i$. Intuitively, we consider the situation where the inverse solver has been initialized with NN parameters of the corresponding forward PINN θ_i and with the correct inverse parameter $\hat{\beta}_i = \beta_i$. When trained with a bad choice of X, the parameters of the inverse solver may drift to other incorrect $\hat{\beta}_i$ which may still be consistent with the observed data. Meanwhile, a good choice of X should retain the correct $\hat{\beta}_i$ throughout training. We formalize this analytically to develop an approximate criterion to optimize for X.

To formalize this, first note that the forward PINN parameters θ_i and its corresponding β_i should minimize $\mathcal{L}(\theta, \beta_i; X_\gamma, \tilde{Y}_{\gamma,i})$ given by (5), and would roughly simultaneously minimize $\mathcal{L}_{\text{obs}}(\theta; X_\gamma, \tilde{Y}_{\gamma,i})$ and $\mathcal{L}_{\text{PDE}}(\theta, \beta)$. Consider an inverse parameter β' close to β_i. We can derive, as done in App. E.3 or in other past works (van der Vaart, 2000; Koh and Liang, 2017), that the NN parameter $\tilde{\theta}_i(\beta')$ which minimizes $\mathcal{L}_{\text{PDE}}(\tilde{\theta}_i(\beta'), \beta')$ would be given by

$$\tilde{\theta}_i(\beta') \approx \theta_i - \left(\nabla_\theta^2 \mathcal{L}_{\text{PDE}}(\theta_i, \beta_i)\right)^{-1} \nabla_\theta \left(\Delta \mathcal{L}_{\text{PDE}}(\theta_i, \beta')\right), \tag{18}$$

where $\Delta\mathcal{L}_{\text{PDE}}(\theta', \beta') = \mathcal{L}_{\text{PDE}}(\theta', \beta') - \mathcal{L}_{\text{PDE}}(\theta', \beta_i)$. Consequently, the error of the observations at a specific β' can be written as

$$\ell_{X_\gamma,i}(\beta') \triangleq \mathcal{L}_{\text{obs}}\left(\tilde{\theta}_i(\beta'); X_\gamma, \tilde{Y}_{\gamma,i}\right) \tag{19}$$

$$= (2M)^{-1} \| \hat{u}_{\tilde{\theta}_i(\beta')}(X_\gamma) - \tilde{Y}_{\gamma,i} \|^2. \tag{20}$$

Given a choice of X_γ and its observation values $\tilde{Y}_{\gamma,i}$, we could characterize how likely the inverse solver would remain at β_i or drift to a neighbouring β' after training by considering the Hessian of $\ell_{X_\gamma,i}(\beta)$ – a "larger" Hessian means a lower chance that $\hat{\beta}_i$ will change as the inverse solver undergoes training to minimize loss. Hence, to optimize for X_γ we could minimize the criterion

$$\hat{\alpha}_{\text{TIP},i}(X_\gamma) = \log\det \nabla_{\beta'}^2 \, \ell_{X_\gamma,i}(\beta_i). \tag{21}$$

Interestingly, we are able to also interpret $\hat{\alpha}_{\text{TIP},i}$ as the information gain of $\hat{\beta}_i$ given observation data $(X_\gamma, \tilde{Y}_{\gamma,i})$, which we discuss in App. E.3.3. This provides an explanation of how TIP could also be viewed with a Bayesian interpretation as well. This also links TIP to some existing ED methods for IPs based on Laplace's approximation (Beck et al., 2018; Alexanderian et al., 2022). Despite these links, our method remains novel in that through the use of PINNs, we can consider the Hessian of the PDE solution directly, allowing the resulting criterion to be differentiable w.r.t. γ. This is unlike past works which often require more careful analysis of the specific PDE involved, and relies on discretized simulations (and therefore are not differentiable w.r.t. the input points).

For completeness, in App. F, we present the full loop for PIED, which incorporates all the approximation methods and the full ED loop from Sec. 3.

5. Experiments

In this section, we present experimental results to demonstrate the performance of PIED. We use multiple physics-based IP in the setup described in Sec. 2. Details on the problem setup and PDE definitions are in App. G. For comparison, we test our criteria against an approximation of the expected mutual information criterion (Foster et al., 2019), and traditional optimal sensor placement criteria (Krause et al., 2008). We discuss these benchmarks along with others tested in App. G.5.

5.1. Shared Initial NN Parameters

We first demonstrate the benefits of learning a shared NN initialization for inverse PINNs. In Fig. 3a, we present the

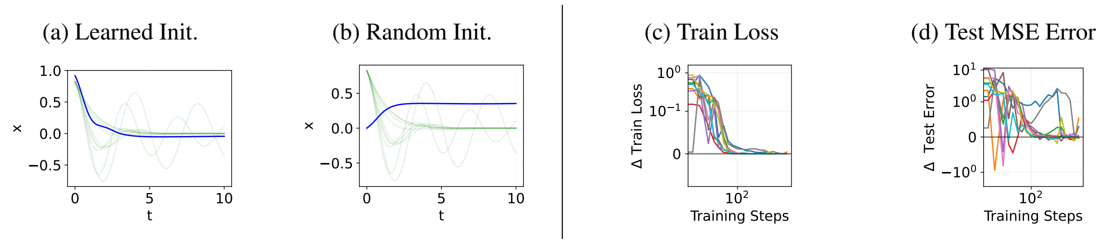

Figure 3. Results for learning a NN initialization for PINNs trained on 1D damped oscillator case. Figs. 3a and 3b consists of the NN initialization (blue line) compared to the PDE solutions u_β for different values of β (faint green lines). Figs. 3c and 3d shows the improvement of the train and test loss of forward PINNs when using a learned NN initialization as compared to using a random NN initialization on different cases of β.

plot of the output of the learned NN initialization for the 1D damped oscillator case. We can see that the learned initialization results in a NN which shares similar structure to the PDE solutions u_β for different values of β, e.g., replicating the damping effect observed in the various solutions. In Figs. 3c and 3d, we plot the *improvement* in train and test loss for forward PINNs when initialized from $\theta_{\text{L-Init}}$ versus when initialized randomly. We can see that in both examples, using the learned NN initialization consistently leads to lower training loss and faster convergence to the minima than when random initialization is used. While given enough training steps we will eventually reach comparable losses regardless of initialization, initializing with $\theta_{\text{L-Init}}$ allows this convergence to happen in fewer steps. We provide further empirical results on $\theta_{\text{L-Init}}$ in App. H.1, which demonstrates that this phenomena also applies for inverse PINNs and for PINNs trained for other PDEs.

5.2. Experimental Design and Inverse Problem Experiments

We now consider the ED problem on different IP scenarios (i.e., different PDEs). Each method computes their optimal observation points, which are then evaluated on multiple IP instances (i.e., different ground truth inverse parameters β). We report the distribution of the error of inverse parameter estimation across all of these instances. This is because some inverse parameters can be more easily estimated than others, and ED methods should optimize the observation points for the best IP performance across all inverse parameters β.

Finite-dimensional inverse parameters. We first present ED problems on IPs with scalar-valued outcomes. In Fig. 4a, we present the ED results for the 1D damped oscillator example. We see that our proposed criteria all consistently outperform the other benchmarks. In App. H.2, we report further experimental results for the different variants of MoTE and TIP along with other ED benchmarks. In summary, we find that using a learned NN initialization leads to better performance in the IP. This is likely due to the quicker convergence of training as discussed in Sec. 5.1, and also the reduced harmful effects from bad random parameter initialization of the inverse PINN. We also find that MoTE criteria performs better as the number of initial training steps r increases. We also conduct experiments with the 1D wave equation, whose results are presented in Fig. 4b. We find that the results show similar trends compared to the damped oscillator example.

Function-valued inverse parameters. We also conducted experiments for more complex scenarios where the inverse parameter of interest is a function defined over $\mathcal{X}$. For our experiments, we parameterize the function-valued inverse parameters using a small NN. Note that in this case the true inverse function can be represented by many different NN parameters. Despite this, our criterion still shows strong performances over the benchmarks.

In Fig. 4c, we present the results for the 2D Eikonal equation. In this problem, the sensors can be placed freely in $\mathcal{X}$, yielding a high-dimensional design parameter. The results show that MoTE and TIP are able to better scale to these high-dimensional problems compared to benchmarks, and are able to select better observation inputs. This is further demonstrated in Fig. 5 where we plot the observation points chosen by TIP. We see that TIP selects points further away from the wave source to obtain more information from signal propagation, leading to better reconstruction of the inverse function compared to other methods. Meanwhile, FIST performs much worse than even random selection, which may be due to the criterion's heavy dependence on its hyperparameter for good performance, and therefore may not be ideal in complex settings.

In practice, we find that while MoTE can achieve error as low as TIP, it often requires a larger r (i.e., more initial training), making it less efficient than TIP. Based on our empirical results, TIP can achieve low error with less training or hyperparameter adjustment, making TIP preferrable over our other criteria. Additional results for the 2D Eikonal equation example can be found in App. H.3.

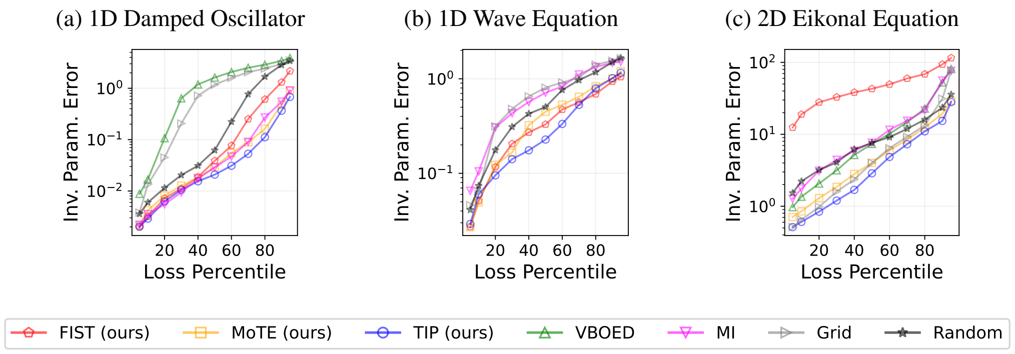

Figure 4. Results from the inverse problems. Figs. 4a and 4b shows examples for finite-dimensional inverse problem, while Fig. 4c shows an example for function-valued inverse problem. In each plots, we show the distribution (i.e., percentile scores) of the IP error across all of the experiment trials.

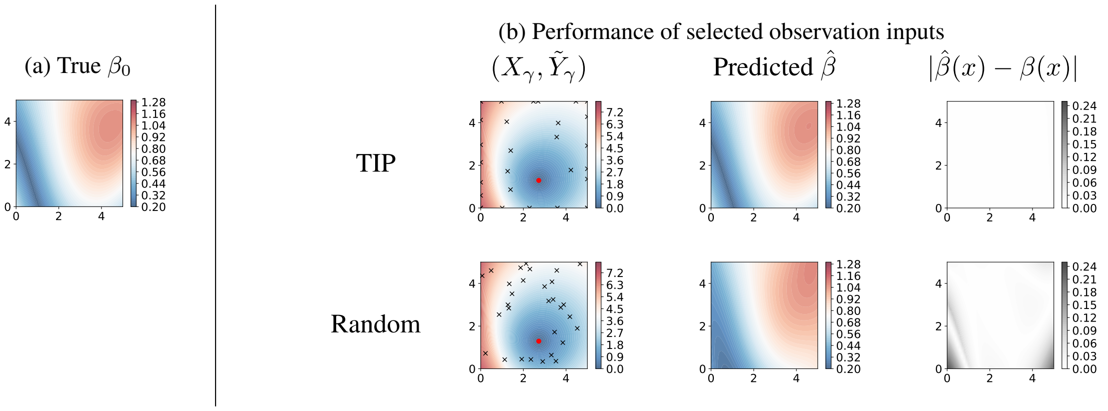

Figure 5. Results from one instance of the IP on Eikonal equation problem. Fig. 5a: true inverse function. Fig. 5b: demonstration of observation inputs selected by TIP and Random methods, and how the observation inputs affect the resulting inverse solver.

6. Conclusion

We have introduced PIED, the first ED framework that utilizes PINNs as both forward simulators and inverse solvers in a fully differentiable architecture to perform continuous optimization of design parameters for IPs. PIED selects optimal design parameters for one-shot deployment, and allows exploitation of parallel computation unlike existing methods. We have also designed effective criteria for the framework which are end-to-end differentiable and hence can be optimized through gradient-based methods.

A limitation of our work is that it is focused on one-shot selection of design parameters, and hence do not cater to adaptive settings that allow for iterative incorporation of new information from additional trials to choose subsequent design parameters. We chose to focus on this to address the many practical scientific settings where one-shot deployment is preferred. However, future work could extend PIED to the adaptive setting, possibly using adopting a Bayesian approach. Future work could also apply PIED to other differentiable physics-informed architectures, such as operator learning methods.

Acknowledgements

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-PhD/2023-01-039J). This research is part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-018).

References

Curtis R. Vogel. *Computational Methods for Inverse Problems*. Society for Industrial and Applied Mathematics, January 2002. ISBN 978-0-89871-550-7 978-0-89871-757-0. doi: 10.1137/1.9780898717570. URL http://epubs.siam.org/doi/book/10.1137/1.9780898717570.

Omar Ghattas and Karen Willcox. Learning physics-based models from data: perspectives from inverse problems and model reduction. *Acta Numerica*, 30:445–554, May 2021. ISSN 0962-4929, 1474-0508. doi: 10.1017/S0962492921000064. URL https://www.cambridge.org/core/product/identifier/S0962492921000064/type/journal_article.

Masataka Tanaka and Huy Duong Bui, editors. *Inverse Problems in Engineering Mechanics: IUTAM Symposium Tokyo, 1992*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993. ISBN 978-3-642-52441-7 978-3-642-52439-4. doi: 10.1007/978-3-642-52439-4. URL https://link.springer.com/10.1007/978-3-642-52439-4.

M. Gazzola, L. H. Dudte, A. G. McCormick, and L. Mahadevan. Forward and inverse problems in the mechanics of soft filaments. *Royal Society Open Science*, 5(6): 171628, June 2018. doi: 10.1098/rsos.171628. URL https://royalsocietypublishing.org/doi/10.1098/rsos.171628. Publisher: Royal Society.

K. Chadan, P. C. Sabatier, and R. G. Newton. *Inverse Problems in Quantum Scattering Theory*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989. ISBN 978-3-642-83319-9 978-3-642-83317-5. doi: 10.1007/978-3-642-83317-5. URL http://link.springer.com/10.1007/978-3-642-83317-5.

Jonathan D. Smith, Kamyar Azizzadenesheli, and Zachary E. Ross. EikoNet: Solving the Eikonal equation with Deep Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 59(12):10685–10696, December 2021. ISSN 0196-2892, 1558-0644. doi: 10.1109/TGRS.2020.3039165. URL http://arxiv.org/abs/2004.00361. arXiv:2004.00361 [physics, stat].

Umair bin Waheed, Ehsan Haghighat, Tariq Alkhalifah, Chao Song, and Qi Hao. PINNeik: Eikonal solution using physics-informed neural networks. *Computers & Geosciences*, 155:104833, October 2021. ISSN 0098-3004. doi: 10.1016/j.cageo.2021.104833. URL https://www.sciencedirect.com/science/article/pii/S009830042100131X.

M Razavy. *An Introduction to Inverse Problems in Physics*. WORLD SCIENTIFIC, July 2020. ISBN 9789811221668 9789811221675. doi: 10.1142/11860. URL https://www.worldscientific.com/worldscibooks/10.1142/11860.

Alen Alexanderian. Optimal experimental design for infinite-dimensional Bayesian inverse problems governed by PDEs: a review. *Inverse Problems*, 37(4):043001, March 2021. ISSN 0266-5611. doi: 10.1088/1361-6420/abe10c. URL https://dx.doi.org/10.1088/1361-6420/abe10c. Publisher: IOP Publishing.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Kathryn Chaloner and Isabella Verdinelli. Bayesian Experimental Design: A Review. *Statistical Science*, 10(3), August 1995. ISSN 0883-4237. doi: 10.1214/ss/1177009939. URL https://projecteuclid.org/journals/statistical-science/volume-10/issue-3/Bayesian-Experimental-Design-A-Review/10.1214/ss/1177009939.full.

Quan Long, Marco Scavino, Raúl Tempone, and Suojin Wang. Fast estimation of expected information gains for Bayesian experimental designs based on Laplace approximations. *Computer Methods in Applied Mechanics and Engineering*, 259:24–39, June 2013. ISSN 0045-7825. doi: 10.1016/j.cma.2013.02.017. URL https://www.sciencedirect.com/science/article/pii/S0045782513000492.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual Information Neural Estimation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 531–540. PMLR, July 2018. URL https://proceedings.mlr.press/v80/belghazi18a.html. ISSN: 2640-3498.

Adam Foster, Martin Jankowiak, Elias Bingham, Paul Horsfall, Yee Whye Teh, Thomas Rainforth, and Noah Goodman. Variational Bayesian Optimal Experimental Design. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/d55cbf210f175f4a37916eafe6c04f0d-Abstract.html.

Desi R Ivanova, Adam Foster, Steven Kleinegesse, Michael U. Gutmann, and Thomas Rainforth. Implicit Deep Adaptive Design: Policy-Based Experimental Design without Likelihoods. In *Advances in Neural Information Processing Systems*, volume 34, pages 25785–25798. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/d811406316b669ad3d370d78b51b1d2e-Abstract.html.

Vincent Lim, Ellen Novoseller, Jeffrey Ichnowski, Huang Huang, and Ken Goldberg. Policy-Based Bayesian Experimental Design for Non-Differentiable Implicit Models, March 2022. URL http://arxiv.org/abs/2203.04272. arXiv:2203.04272 [cs, stat].

Alen Alexanderian, Ruanui Nicholson, and Noemi Petra. Optimal design of large-scale nonlinear Bayesian inverse problems under model uncertainty, November 2022. URL http://arxiv.org/abs/2211.03952. arXiv:2211.03952 [cs, math, stat].

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms, October 2018. URL http://arxiv.org/abs/1803.02999. arXiv:1803.02999 [cs].

Xu Liu, Xiaoya Zhang, Wei Peng, Weien Zhou, and Wen Yao. A novel meta-learning initialization method for physics-informed neural networks. *Neural Computing and Applications*, 34:14511 – 14534, 2021. URL https://api.semanticscholar.org/CorpusID:236318461.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/5a4be1fa34e62bb8a6ec6b91d2462f5a-Abstract.html.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective, July 2020. URL http://arxiv.org/abs/2007.14527. arXiv:2007.14527 [cs, math, stat].

Gregory Kang Ruey Lau, Apivich Hemachandra, See-Kiong Ng, and Bryan Kian Hsiang Low. PINNACLE: PINN Adaptive ColLocation and Experimental points selection, April 2024. URL https://arxiv.org/abs/2404.07662v1.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep Neural Networks as Gaussian Processes. *arXiv:1711.00165 [cs, stat]*, March 2018. URL http://arxiv.org/abs/1711.00165. arXiv: 1711.00165.

A. W. van der Vaart. *Asymptotic Statistics*. Asymptotic Statistics. Cambridge University Press, 2000. ISBN 9780521784504.

Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1885–1894. PMLR, July 2017. URL https://proceedings.mlr.press/v70/koh17a.html. ISSN: 2640-3498.

Joakim Beck, Ben Mansour Dia, Luis FR Espath, Quan Long, and Raul Tempone. Fast Bayesian experimental design: Laplace-based importance sampling for the expected information gain. *Computer Methods in Applied Mechanics and Engineering*, 334:523–553, June 2018. ISSN 00457825. doi: 10.1016/j.cma.2018.01.053. URL http://arxiv.org/abs/1710.03500. arXiv:1710.03500 [math].

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *The Journal of Machine Learning Research*, 9:235–284, June 2008. ISSN 1532-4435.

Lorenz T. Biegler, Omar Ghattas, Matthias Heinkenschloss, and Bart van Bloemen Waanders. Large-Scale PDE-Constrained Optimization: An Introduction. In Lorenz T. Biegler, Matthias Heinkenschloss, Omar Ghattas, and Bart van Bloemen Waanders, editors, *Large-Scale PDE-Constrained Optimization*, pages 3–13, Berlin, Heidelberg, 2003. Springer. ISBN 978-3-642-55508-4. doi: 10.1007/978-3-642-55508-4_1.

Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, January 2023. ISSN 00457825. doi: 10.1016/j.cma.2022.115671. URL https://linkinghub.elsevier.com/retrieve/pii/S0045782522006260.

Tom Rainforth, Adam Foster, Desi R. Ivanova, and Freddie Bickford Smith. Modern Bayesian Experimental Design, February 2023. URL http://arxiv.org/abs/2302.14545. arXiv:2302.14545 [cs, stat].

Jay I. Myung, Daniel R. Cavagnaro, and Mark A. Pitt. A tutorial on adaptive design optimization. *Journal of Mathematical Psychology*, 57(3):53–67, June

2013. ISSN 0022-2496. doi: 10.1016/j.jmp.2013.05.005. URL https://www.sciencedirect.com/science/article/pii/S0022249613000503.

Quan Long, Mohammad Motamed, and Raúl Tempone. Fast Bayesian optimal experimental design for seismic source inversion. *Computer Methods in Applied Mechanics and Engineering*, 291:123–145, July 2015. ISSN 0045-7825. doi: 10.1016/j.cma.2015.03.021. URL https://www.sciencedirect.com/science/article/pii/S0045782515001310.

Adam Foster, Desi R. Ivanova, Ilyas Malik, and Tom Rainforth. Deep Adaptive Design: Amortizing Sequential Bayesian Experimental Design. In *Proceedings of the 38th International Conference on Machine Learning*, pages 3384–3395. PMLR, July 2021. URL https://proceedings.mlr.press/v139/foster21a.html. ISSN: 2640-3498.

J.R. Taylor. *Classical Mechanics*. Information and Interdisciplinary Subjects Series. University Science Books, 2005. ISBN 9781891389221.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12): 124002, December 2020. ISSN 1742-5468. doi: 10.1088/1742-5468/abc62b. URL http://arxiv.org/abs/1902.06720. arXiv: 1902.06720.

Sacha Binder. Wave equation simulations 1d/2d (équation de d'alembert). https://github.com/sachabinder/wave_equation_simulations, 2021.

Malcolm C. A. White, Hongjian Fang, Nori Nakata, and Yehuda Ben-Zion. PyKonal: A Python Package for Solving the Eikonal Equation in Spherical and Cartesian Coordinates Using the Fast Marching Method. *Seismological Research Letters*, 91(4):2378–2389, June 2020. ISSN 0895-0695. doi: 10.1785/0220190318. URL https://doi.org/10.1785/0220190318.

Peter I. Frazier. A tutorial on bayesian optimization, 2018.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and modular implicit differentiation. *arXiv preprint arXiv:2105.15183*, 2021.

A. Notations

Table 1. List of notations used throughout the paper

Symbol	Meaning	Example
$\mathcal{D}$	PDE operator	(1)
$\mathcal{B}$	Boundary condition operator	(1)
$\mathcal{X}$	Input domain	(1)
$\partial\mathcal{X}$	Boundary of input domain	(1)
$\mathcal{S}$	Set of feasible observation inputs	
β	Inverse parameter	(1)
u_β	Solution for (1) with inverse parameter β	(3)
$\hat{\beta}$	Estimate of inverse parameter from inverse solver	(3)
θ	NN parameter	(5)
$\hat{u}_\theta$	NN with parameters θ	(5)
$\mathcal{L}$	PINN training loss	(5)
$\mathcal{L}_{\text{obs}}$	Observation loss for PINN	(5)
$\mathcal{L}_{\text{PDE}}$	Collocation points loss for PINN	(5)
$\mathbf{F}$	Forward simulator	(8)
$\tilde{u}_{\beta_i}$	PINN, with NN parameter θ_i, which estimates u_{β_i}	(8)
$\mathbf{O}_X$	Observation selector with input X	(9)
$\tilde{Y}$	Mock observation output from $\tilde{u}_{\beta_i}(X)$	(9)
γ	Parameterization for observation input	(10)
X_γ	Observation input corresponding to parameter γ	(10)
$\mathcal{S}_\gamma$	Set of valid observation input parameterization	(10)
$\mathbf{I}$	Inverse solver	(11)
α	ED criterion without inverse ensemble approximation	(12)
α_i	ED criterion computed based on outputs of thread i of framework	(12)
∇_x	Derivative or Jacobian w.r.t. x	
∇_x^2	Hessian w.r.t. x	
$[a, b]$	Closed interval between a and b	

B. Related Works

Inverse problems. Inverse problem (IP) (Vogel, 2002; Ghattas and Willcox, 2021) is an commonly studied class of problem in many science and engineering disciplines such as classical mechanics (Tanaka and Bui, 1993; Gazzola et al., 2018), quantum mechanics (Chadan et al., 1989) or geophysics (Smith et al., 2021; Waheed et al., 2021).

Many methods of solving IPs have been proposed, often involving minimizing the objective as stated in (3), possibly with addition of some regularization terms. One such method is by using the Newton-conjugate gradient method (Biegler et al., 2003; Ghattas and Willcox, 2021) to optimize the objective as stated in (3). However, the method relies on finding the optimal β through gradient update steps, which requires computing the gradient and Hessian of the objective function with respect to β. The computation of the gradient and hessian is often done by reformulating the IP (which can be viewed as a constrained optimization problem) to instead be based on the Lagrangian, then using adjoint methods to compute the corresponding gradient or Hessian (Ghattas and Willcox, 2021). This typically results in gradient and Hessian computations requiring only some finite rounds of forward simulations instead. The computed Hessian can often also be used in Laplace's approximation in order to obtain a posterior distribution $p(\beta|X, Y)$ for the inverse parameter (Long et al., 2013; Beck et al., 2018; Ghattas and Willcox, 2021). However, this method is still restrictive since it may involve careful analysis of the PDE that is involved in the IP in order to form the correct Lagrangian and compute the adjoint.

Physics-informed neural networks. In recent years, physics-informed neural networks (PINNs) have been proposed as another method used to both perform forward simulations of PDE-based problems (Raissi et al., 2019) and for solving IPs (Raissi et al., 2019). PINNs solve PDEs by parameterizing the PDE solution using a neural network (NN), then finding the NN parameter such that the resulting NN obeys the specified PDE and the IC/BCs. This is done so via collocation points, which are pseudo-training points for enforcing the PDE and IC/BC soft constraints.

PINNs are difficult to train in many PDE instances, such as when the solution is known to have higher frequencies (Wang et al., 2020). As a result, many works have been proposed in improving the trianing of PINNs by rescaling the loss functions of PDEs (Wang et al., 2020), or through more careful selection of collocation points (Wu et al., 2023; Lau et al., 2024).

Experimental design. Typically, in solving IPs, the observation data will not be available right away, but instead has to be measured from some physical system. Due to the costs in making measurements of data, it is important that the observations made are carefully chosen to maximize the amount of information that can be obtained from the observations. Experimental design (ED) is a problem which attempts to find out what the best data to observe in order to gather the most information about the unknown quantity of interest (Rainforth et al., 2023).

Bayesian ED (BED) has been one of the frameworks used for solving ED (Chaloner and Verdinelli, 1995; Myung et al., 2013). BED considers the IP in a Bayesian framework, where Bayesian inference is used in the IP. In this case, the IP would result in a posterior distribution $p(\beta|X, Y)$. In BED, the criterion to be maximized would be some function of the posterior distribution, such as the A-optimality (based on the trace of the posterior covariance matrix) or the D-optimality (based on the expected information gain). Many works have proposed schemes to approximate the criteria in BED (Myung et al., 2013; Long et al., 2015; Beck et al., 2018; Foster et al., 2019). We elaborate on criteria used in BED in App. C.2. Other methods for solving ED aside from bayesian frameworks have also been proposed, such as policy-based methods (Foster et al., 2021).

C. Additional Background Information

C.1. Examples of PDEs

For more ideas behind the PDE problem setup, we provide multiple examples of PDEs and how the IP can be formulated based on the PDEs. These examples are also used in the experiments of the paper as well, as further elaborated in App. G.2.

Damped oscillator. The damped oscillator is one of the introductory second-order ordinary differential equation (ODE) in classical mechanics (Taylor, 2005). We consider the example due to the existence of a closed-form solution and its nice interpretation under our ED framework.

Imagine a mass-spring system which is laid horizontally. The spring has spring constant k and experiences a resistive force which is proportional to its current speed, where the constant of proportionality is μ. We let the attached mass have a mass of M. We also assume the case where there are no external driving forces on the system. By applying the relevant forces into Newton's law of motion, the displacement of the mass $x(t)$ can be expressed by the differential equation

$$M\frac{d^2x}{dt^2} + \mu\frac{dx}{dt} + kx = 0. \tag{22}$$

Given the IC of $x(0) = x_0$ and $\frac{dx}{dt}(0) = v_0$, we can write the solution as (Taylor, 2005)

$$x(t) = \begin{cases} Ae^{-\gamma t}\cos(\sqrt{\omega_0^2 - \gamma^2}t + \phi) & \text{if } \gamma < \omega_0, \\ (Bt + C)e^{-\omega_0 t} & \text{if } \gamma = \omega_0, \\ De^{-(\gamma + \sqrt{\gamma^2 - \omega_0^2})t} + Fe^{-(\gamma - \sqrt{\gamma^2 - \omega_0^2})t} & \text{if } \gamma > \omega_0, \end{cases} \tag{23}$$

where $\gamma = \mu/2M$, $\omega_0 = \sqrt{k/M}$, and A, ϕ, B, C, D, F are constants which depends on x_0 and v_0.

Wave equation. For simplicity, we consider the 1D wave equation, which is given by

$$\frac{\partial^2 u}{\partial t^2} = v^2\frac{\partial^2 u}{\partial x^2} \tag{24}$$

where v represents the speed of wave propagation, which may be a scalar or a function of x. In the inevrse problem setup, one may be required to recover the wave velocity v given measurements of $u(x, t)$.

Eikonal equation. Consider the Eikonal problem setup, which is often use to reconstruct material composition of some region based on how waves propagated through the medium reacts. Its equation relates the wave speed $v(x)$ at a point and the wave propagation time $T(x)$ at a point with PDE given by (Smith et al., 2021)

$$T(x) = \big(\nabla v(x)\big)^{-1} \quad \text{with} \quad T(x_0) = 0 \tag{25}$$

where x_0 is where the wave propagates from.

The goal of the experiments are to recover the true function v. However, this involves conducting seismic activities at different set values of x_0, and obtaining the corresponding reading for $T(x)$ at specified values of x.

C.2. Bayesian Experimental Design

A certain variant often considered in ED is Bayesian experimental design (BED). In the BED framework, we assume a prior $p(\beta)$ on the inverse parameter to compute. For a given design parameter d, the system observes some output y.

$$p(\beta|d, y) = \frac{p(y|\beta, d)\, p(\beta)}{\mathbb{E}_{\beta\sim p(\beta)}\big[p(y|\beta, d)\big]}. \tag{26}$$

Given the inference we can compute the expected information gain (EIG), which is sometimes also known as the Bayesian D-optimal criterion. EIG criterion is defined as the expected Kullback-Leibler divergence between the prior $p(\beta)$ and the posterior $p(\beta|d, y)$, averaged over the possible observations y. More formally, this can be written as

$$\mathrm{EIG}(d) = \mathbb{E}_{y\sim p(y|d)}\big[D_{\mathrm{KL}}\big(p(\beta|d, y)\|p(\beta)\big)\big] = \mathbb{H}[p(\beta)] - \mathbb{E}_{y'\sim p(y|d)}\big[\mathbb{H}[p(\beta|d, y = y')]\big] \tag{27}$$

where the posterior is defined in (26). A naive approximation technique is to perform a nested Monte Carlo (NMC) approximation (Myung et al., 2013).

$$\mathrm{EIG}(d) \approx \frac{1}{N}\sum_{i=1}^{N} \log \frac{p(y_i|\beta_{i,0}, d)}{\frac{1}{M}\sum_{j=1}^{M} p(y_i|\beta_{i,j}, d)} \tag{28}$$

where $\beta_{i,0}, \beta_{i,1}, \ldots, \beta_{i,j} \sim p(\beta)$ and $y_i \sim p(y|\beta_{i,0}, d)$. In [Appendix], we demonstrate that the estimate approaches the true EIG as $N, M \to \infty$. In practice, however, using the NMC estimator results in a biased estimtor for finite N and M, and results in slow convergence with N and M. To improve on the NMC estimator, various schemes have been proposed mainly to remove the need to perform two nested MC rounds, including variational methods (Foster et al., 2019) and Laplace approximation methods (Long et al., 2015; Beck et al., 2018).

D. Parameterization of Input Points

To demonstrate the flexibility of the input points parameterization, we provide some examples of possible methods to constrain the input points and how they be expressed in the appropriate forms. Fig. 6 graphically demonstrate what some of these observation input constraints may look like.

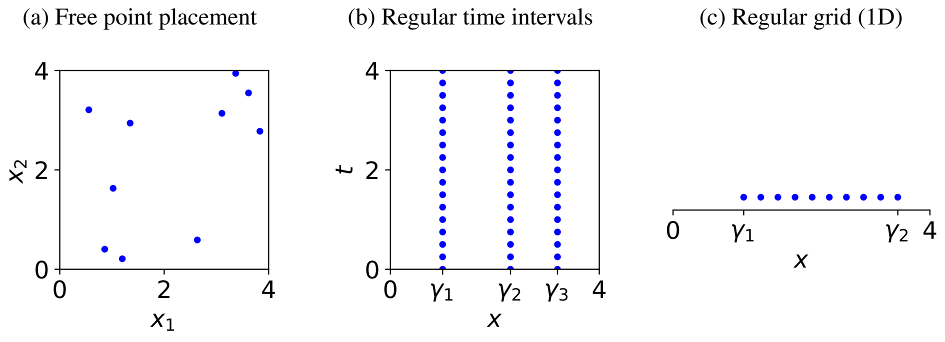

Figure 6. Examples of observation input placements.

Points placed freely in input space. In this case, the points can be placed anywhere in $\mathcal{X}$ without restriction. To parameterize this, we define $d = Md_{\text{in}}$, and define

$$X_\gamma = \Big\{ \big(\gamma_1 \; \gamma_2 \; \cdots \; \gamma_{d_{\text{in}}} \big), \; \big(\gamma_{d_{\text{in}}+1} \; \gamma_{d_{\text{in}}+2} \; \cdots \; \gamma_{2d_{\text{in}}} \big), \; \ldots, \; \big(\gamma_{Md_{\text{in}}-d_{\text{in}}} \; \gamma_{Md_{\text{in}}-d_{\text{in}}+1} \; \cdots \; \gamma_{Md_{\text{in}}} \big) \Big\} \tag{29}$$

Points placed at regular time intervals. In this case, we assume the points are placed at chosen spatial locations, and makes measurements at fixed time intervals $t_1, t_2, \ldots, t_f$. This is realistic in the case where the PDE solution evolves over time, and so it makes sense to fix the location of the sensor but allow it to make readings throughout the evolution of the system over time. In this case, γ only needs to encode the spatial location where the sensors should be placed. Specifically, if there is one spatial dimension, then the parameterization for the sensors can be chosen as

$$X_\gamma = \Big\{ (x, t) : x \in \big\{ \gamma_1, \gamma_2, \ldots, \gamma_d \big\} \text{ and } t \in \big\{ t_1, t_2, \ldots, t_f \big\} \Big\}. \tag{30}$$

Points placed in a regular grid. In this case, the points are placed in a regular grid at regular intervals. This provides one way to add extra constraints for sensor configurations to reduce the dimension of the problem. For demonstration, in 1D problems, if we want to allow placement of s sensors in total, we can let $d = 2$ and parameterize the sensor placements as

$$X_\gamma = \left\{ \gamma_1, \; \gamma_1 + \frac{\gamma_2 - \gamma_1}{s-2}, \; \gamma_1 + 2 \cdot \frac{\gamma_2 - \gamma_1}{s-2}, \; \ldots, \; \gamma_2 \right\}. \tag{31}$$

E. Further Details About The Experimental Design Criterion

E.1. Few-step Inverse Solver Training Criterion

E.1.1. Pseudocode for Criterion Computation

Algorithm 1 Criterion estimation by Few-step Inverse Solver Training

```
1:  function $\hat{\alpha}_{\text{FIST},i}(X_\gamma)$
2:      // Perturbation of NN and estimated inverse parameters
3:      $\bar{\theta} \leftarrow \theta_i + \varepsilon_\theta$ where $\varepsilon_\theta \sim \mathcal{N}(0, \sigma^2)$
4:      $\bar{\beta} \leftarrow \beta_i + \varepsilon_\beta$ where $\varepsilon_\beta \sim \mathcal{N}(0, \sigma^2)$
5:      $\tilde{Y}_\gamma \leftarrow \tilde{u}_{\beta_i}(X_\gamma)$
6:      // Partial training stage
7:      Initialize $(\theta^{(0)}, \beta^{(0)})$
8:      for $j = 1, \ldots, r$ do
9:          // The training may be replaced with other gradient-based methods as well in practice
10:         $\theta^{(j)} \leftarrow \theta^{(j-1)} - \eta \nabla_\theta \mathcal{L}(\theta^{(j-1)}, \hat{\beta}^{(j-1)}; X_\gamma, \tilde{Y}_\gamma)$
11:         $\hat{\beta}^{(j)} \leftarrow \hat{\beta}^{(j-1)} - \eta \nabla_\beta \mathcal{L}(\theta^{(j-1)}, \hat{\beta}^{(j-1)}; X_\gamma, \tilde{Y}_\gamma)$
12:     end for
13:     return $-\|\hat{\beta}^{(r)} - \beta_i\|^2$
14: end function
```

E.1.2. Limitations of Few-step Inverse Solver Training

A limitation of the criterion is that it contains two hyperparameters, the perturbation noise level σ^2 and the number of training steps r, which can vary the performance level of the criterion. This causes a problem since it may be difficult to find the correct set of hyperparameters which maximizes the ED method performance.

Furthermore, a large value of r can also make the criterion not differentiable in practice due to the need to perform back-propagation over the gradient descent update steps. Fortunately, we find that in our experiments, using r in the order of 10^2 is sufficient given the correct perturbation noise level is set.

E.2. Model Training Estimate Criterion

E.2.1. Assumptions and Proof of (14)

In this section, we will consider PINNs in the linearized regime to demonstrate the validity of the approximation given in (14). The results will be an extension from that given in (Lee et al., 2020). For convenience, we will drop the subscript and write the learnable inverse parameter as $\hat{\beta}$ and the NN parameters as θ.

We first recall the assumptions for the linearized regime of NNs. Following past works on the NTK for NNs (Jacot et al., 2018; Lee et al., 2020) and PINNs (Lau et al., 2024), we assume that

$$\hat{u}_\theta(x) \approx \hat{u}_{\theta^{(0)}}(x) + J^{(0)}_{\gamma,\theta}(\theta - \theta^{(0)}) \tag{32}$$

and

$$\mathcal{D}[\hat{u}_\theta, \hat{\beta}](x) \approx \mathcal{D}[\hat{u}_{\theta^{(0)}}, \hat{\beta}^{(0)}](x) + \begin{bmatrix} J^{(0)}_{p,\hat{\beta}} & J^{(0)}_{p,\theta} \end{bmatrix} \begin{bmatrix} \hat{\beta} - \hat{\beta}^{(0)} \\ \theta - \theta^{(0)} \end{bmatrix}. \tag{33}$$

We briefly discuss the consequences of this approximation. The linearized regime only holds when the learned parameters are similar to the initial parameters, which based on past works will hold when the NN is wide enough or when the NN is near convergence. This is unlikely to hold in the real training of PINNs, except in the case where the initialized parameters are already close to the true converged parameters anyway. Nonetheless, in our work, we do not use the assumptions to make actual predictions on the inverse parameters in the IP, however only use it to predict the direction of descent for γ, which would only use the local values anyway. Furthermore, we can also perform some pre-training in order to get closer to the converged parameters first as well to make the assumptions more valid.

Let $\hat{\beta}^{(t)}$ and $\theta^{(t)}$ be the learned inverse parameter and NN parameter respectively at step t of the GD training. We will write $J^{(t)}_{\gamma,\theta} = \nabla_\theta \hat{u}_{\theta^{(t)}}(X_\gamma)$, $J^{(t)}_{p,\theta} = \nabla_\theta \mathcal{D}[\hat{u}_{\theta^{(t)}}, \hat{\beta}^{(t)}](X_p)$, and $J^{(t)}_{p,\hat{\beta}} = \nabla_\beta \mathcal{D}[\hat{u}_{\theta^{(t)}}, \hat{\beta}^{(t)}](X_p)$ (note that $\nabla_\beta \hat{u}_{\theta^{(t)}}(X_\gamma) = 0$). We can then write the GD training step as

$$\frac{\partial}{\partial t}\begin{bmatrix} \hat{\beta}^{(t)} \\ \theta^{(t)} \end{bmatrix} = -\eta \begin{bmatrix} \nabla_\beta \mathcal{L}(\theta^{(t)}, \hat{\beta}^{(t)}; X_\gamma, Y) \\ \nabla_\theta \mathcal{L}(\theta^{(t)}, \hat{\beta}^{(t)}; X_\gamma, Y) \end{bmatrix} \tag{34}$$

$$= -\eta \underbrace{\begin{bmatrix} 0 & J^{(t)}_{p,\beta} \\ J^{(t)}_{\gamma,\theta} & J^{(t)}_{p,\theta} \end{bmatrix}}_{\mathcal{J}^{(t)}_\gamma} \begin{bmatrix} \hat{u}_{\theta^{(t)}}(X_\gamma) - \tilde{Y}_\gamma \\ \mathcal{D}[\hat{u}_{\theta^{(t)}}, \hat{\beta}^{(t)}](X_p) - f(X_p) \end{bmatrix}. \tag{35}$$

Under the linearized regime, we can see that, $J^{(t)}_{\gamma,\theta} \approx J^{(0)}_{\gamma,\theta}$, $J^{(t)}_{p,\hat{\beta}} \approx J^{(0)}_{p,\hat{\beta}}$ and $J^{(t)}_{p,\theta} \approx J^{(0)}_{p,\theta}$. We can use these approximations to obtain

$$\frac{\partial}{\partial t}\begin{bmatrix} \hat{\beta}^{(t)} - \hat{\beta}^{(0)} \\ \theta^{(t)} - \theta^{(0)} \end{bmatrix} = \frac{\partial}{\partial t}\begin{bmatrix} \hat{\beta}^{(t)} \\ \theta^{(t)} \end{bmatrix} \tag{36}$$

$$\approx -\eta \mathcal{J}^{(0)}_\gamma {\mathcal{J}^{(0)}_\gamma}^\top \begin{bmatrix} \hat{\beta}^{(t)} - \hat{\beta}^{(0)} \\ \theta^{(t)} - \theta^{(0)} \end{bmatrix} - \eta \mathcal{J}^{(0)}_\gamma \begin{bmatrix} \hat{u}_{\theta^{(0)}}(X_\gamma) - \tilde{Y}_\gamma \\ \mathcal{D}[\hat{u}_{\theta^{(0)}}, \hat{\beta}^{(0)}](X_p) - f(X_p) \end{bmatrix} \tag{37}$$

which can be solved to give

$$\begin{bmatrix} \hat{\beta}^{(t)} - \hat{\beta}^{(0)} \\ \theta^{(t)} - \theta^{(0)} \end{bmatrix} = -{\mathcal{J}^{(0)}_\gamma}^\top (\mathcal{J}^{(0)}_\gamma {\mathcal{J}^{(0)}_\gamma}^\top)^{-1} \big(I - e^{-\eta \mathcal{J}^{(0)}_\gamma {\mathcal{J}^{(0)}_\gamma}^\top t}\big) \begin{bmatrix} \hat{u}_{\theta^{(0)}}(X_\gamma) - \tilde{Y}_\gamma \\ \mathcal{D}[\hat{u}_{\theta^{(0)}}, \hat{\beta}^{(0)}](X_p) - f(X_p) \end{bmatrix}. \tag{38}$$

At convergence, i.e., when $t \to \infty$, we can reduce the results for $\hat{\beta}^{(\infty)} - \hat{\beta}^{(0)}$ as

$$\hat{\beta}^{(\infty)} - \hat{\beta}^{(0)} \approx \begin{bmatrix} 0 \\ {J^{(0)}_{p,\beta}}^\top \end{bmatrix} (\mathcal{J}^{(0)}_\gamma {\mathcal{J}^{(0)}_\gamma}^\top)^{-1} \begin{bmatrix} \hat{u}_{\theta^{(0)}}(X_\gamma) - \tilde{Y}_\gamma \\ \mathcal{D}[\hat{u}_{\theta^{(0)}}, \hat{\beta}^{(0)}](X_p) - f(X_p) \end{bmatrix} \tag{39}$$

as claimed in (14).

E.2.2. Pseudocode For Criterion Computation

Algorithm 2 Criterion estimation by Model Training Estimate

```
1:  function $\hat{\alpha}_{\text{MoTE},i}(\gamma)$
2:      Initialize $(\theta^{(0)}, \beta^{(0)})$                ▷ Can set $\theta^{(0)}$ to $\theta_{\text{L-Init}}$ as well
3:      $\tilde{Y}_\gamma \leftarrow \tilde{u}_{\beta_i}(X_\gamma)$
4:      for $j = 1, \dots, r$ do
5:          // The training may be replaced with other gradient-based methods as well in practice
6:          $\theta^{(j)} \leftarrow \theta^{(j-1)} - \eta \nabla_\theta \mathcal{L}(\theta^{(j-1)}, \hat{\beta}^{(j-1)}; X_\gamma, \tilde{Y}_\gamma)$
7:          $\hat{\beta}^{(j)} \leftarrow \hat{\beta}^{(j-1)} - \eta \nabla_\beta \mathcal{L}(\theta^{(j-1)}, \hat{\beta}^{(j-1)}; X_\gamma, \tilde{Y}_\gamma)$
8:      end for
9:      // To prevent backpropagation over the GD training
10:     Set $\nabla_\gamma \theta^{(r)} = 0$ and $\nabla_\gamma \hat{\beta}^{(r)} = 0$
11:     Perform estimation
            $$\hat{\beta}^{(\infty)} = \hat{\beta}^{(r)} - \begin{bmatrix} 0 \\ J_{p,\beta}^{(r)\top} \end{bmatrix} (\mathcal{J}_\gamma^{(r)} \mathcal{J}_\gamma^{(r)\top})^{-1} \begin{bmatrix} \hat{u}_{\theta^{(r)}}(X_\gamma) - \tilde{Y}_\gamma \\ \mathcal{D}[\hat{u}_{\theta^{(r)}}, \hat{\beta}^{(r)}](X_p) - f(X_p) \end{bmatrix} \quad (40)$$
12:     return $-\|\hat{\beta}^{(\infty)} - \beta_i\|^2$
13: end function
```

Note that in Line 10 of Alg. 2, we set the gradients $\nabla_\gamma \theta^{(r)}$ and $\nabla_\gamma \hat{\beta}^{(r)}$ to zero. This is done since when implementing the function, it will be possible to write $\theta^{(r)}$ and $\hat{\beta}^{(r)}$ as explicit functions in terms of θ, meaning that when performing back-propagation over the Model Training Estimate criteria, it will also consider these derivatives as well. This can cause memory issues due to the need of back-propagating over many GD steps. Therefore, by explicitly stating that the gradient is zero, it avoids problems during the back-propagation phase. In practice, this can be done via the `stop_gradient` function on JAX, for example.

A speedup that can be applied on MoTE is to instead of performing initial pretraining to obtain the eNTK, we could reuse the NN parameters from the forward PINN in order to compute the eNTK instead. We find that this trick is useful since it gives performance almost as good as performing initial training in each criterion computation, while being more efficient since no additional training needs to be done.

E.2.3. Limitations of Model Training Estimate

MoTE relies on estimation of $\hat{\beta}$ via the eNTK, which as experiments demonstrate, becomes better when the eNTK used comes from a more "trained" NN. This means that while we could make our criterion more accurate, more NN training (i.e., higher r) per query and hence more computational resources would be required.

E.3. Tolerable Inverse Parameter Criterion

E.3.1. Assumptions and Rough Proof of (18)

In this section, we demonstrate the validity of (18), which is adapted from the proof in van der Vaart (2000). For convenience, we will drop the subscript and consider the NN parameters θ and inverse parameters β.

Suppose we fix a β, and let $\theta = \arg\min_{\theta'} \mathcal{L}_{\text{PDE}}(\theta', \beta)$. Since (θ, β) is a minima of $\mathcal{L}_{\text{PDE}}$, we can see that $\nabla_\theta \mathcal{L}_{\text{PDE}}(\theta, \beta) = 0$.

Let $\tilde{\theta}(\beta') = \arg\min_{\theta'} \mathcal{L}_{\text{PDE}}(\theta', \beta')$. Our goal is to approximate $\tilde{\theta}(\beta')$ when $\beta' \approx \beta$.

Let $\Delta\mathcal{L}_{\text{PDE}}(\theta', \beta') = \mathcal{L}_{\text{PDE}}(\theta', \beta') - \mathcal{L}_{\text{PDE}}(\theta', \beta')$ and $\Delta\tilde{\theta}(\beta') = \tilde{\theta}(\beta') - \tilde{\theta}(\beta) = \tilde{\theta}(\beta') - \theta$. We can use this to write

$$\begin{aligned}
\nabla_\theta \mathcal{L}_{\text{PDE}}(\tilde{\theta}(\beta'), \beta') &= \nabla_\theta \mathcal{L}_{\text{PDE}}(\theta + \Delta\tilde{\theta}(\beta'), \beta') && (41)\\
&\approx \nabla_\theta \mathcal{L}_{\text{PDE}}(\theta, \beta') + \big(\nabla^2_\theta \mathcal{L}_{\text{PDE}}(\theta, \beta')\big)\Delta\tilde{\theta}(\beta') && (42)\\
&= \nabla_\theta \mathcal{L}_{\text{PDE}}(\theta, \beta) + \Delta\mathcal{L}_{\text{PDE}}(\theta, \beta') + \big(\nabla^2_\theta \mathcal{L}_{\text{PDE}}(\theta, \beta')\big)\Delta\tilde{\theta}(\beta') && (43)\\
&= \Delta\mathcal{L}_{\text{PDE}}(\theta, \beta') + \big(\nabla^2_\theta \mathcal{L}_{\text{PDE}}(\theta, \beta')\big)\Delta\tilde{\theta}(\beta') && (44)
\end{aligned}$$

where (42) arises from performing Taylor expansion on $\nabla_\theta \mathcal{L}_{\text{PDE}}(\tilde{\theta}(\beta'), \beta')$ around θ.

Since $(\tilde{\theta}(\beta'), \beta')$ is a minima of $\mathcal{L}_{\text{PDE}}$, we know that $\nabla_\theta \mathcal{L}_{\text{PDE}}(\tilde{\theta}(\beta'), \beta') = 0$, and therefore we can solve for $\Delta\tilde{\theta}(\beta')$ to obtain

$$\Delta\tilde{\theta}(\beta') \approx -\big(\nabla^2_\theta \mathcal{L}_{\text{PDE}}(\theta, \beta')\big)^{-1} \Delta\mathcal{L}_{\text{PDE}}(\theta, \beta') \tag{45}$$

which can be rewritten to match the form in (18).

Some readers may question whether (18) is valid for NNs where the learned NN parameter may not be a global minima. We note that despite this, we are only interested in the curvature around a minima anyway, and so we can still inspect the change of that minima as the loss function changes regardless. Furthermore, this technique has been used for NNs in other applications as well, one notable instance being the influence function (Koh and Liang, 2017) which aims to study how the test performance of supervised learning tasks changes as certain training examples are upweighed. In the paper, they are able to design a scoring function based on the same mathematical tool and successfully interpret performances of NNs. In our work, we find that despite the assumptions on the global minima is not met, we are still able to achieve good empirical results as well.

E.3.2. Choice of Loss Function Used in (18)

Note that in (18), we compute the Hessian w.r.t. the forward PINN loss. Some readers may wonder why the overall PINN loss from (5) is not used instead.

This choice is due to two main reasons. First, it is more computationally efficient. Notice that the change in NN parameter in (18) depends on $\mathcal{L}_{\text{PDE}}$, and therefore are independent of the observations and hence independent of the design parameters γ. This leads to the optimizing of the final criterion to not require differentiating (18) with respect to γ, reducing the computational load during optimization. We find that doing so does not cause significant effect in the obtained design parameter γ^*.

Second, this matches more closely to the inverse problem setup as described in Sec. 2. In the IP as described, the objective (3) is usually to find the β whose output matches that of Y. In TIP, the tolerable parameters is the Hessian based on the objective (20), which can also be interpreted in a similar way as (3). Furthermore, through this interpretation, TIP also exhibits a stronger connection to Bayesian methods, as discussed in App. E.3.3.

E.3.3. Bayesian Interpretation of TIP

Interestingly, it is possible to provide a Bayesian viewpoint for the TIP criterion. In brief, TIP can be seen as the application of the Laplace approximation on the posterior distribution of β after observing data (X_γ, Y), assuming a uniform prior and Gaussian observation noise.

Assume that the observed data is generated from the true underlying function with added Gaussian noise. Then, the likelihood function can be written as

$$p(Y|\beta, X_\gamma) = \mathcal{N}(Y|u_\beta(X_\gamma), \sigma^2 I) = \prod_{j=1}^{M} \mathcal{N}(y_j|u_\beta(x_{\gamma,j}), \sigma^2). \tag{46}$$

If we assume a uniform prior over $\mathcal{B}$ (i.e., assume $p(\beta) = c$ for some constant c), then it is simple to show that $p(\beta|X_\gamma, Y) =$

$p(Y|\beta, X_\gamma)/p(Y|X_\gamma)$, where $p(Y|X_\gamma)$ can be treated as a constant. In this case, we can write the log posterior as

$$\log p(\beta|X_\gamma, Y) = \left[\sum_{j=1}^{N} \log \mathcal{N}(y_j|u_\beta(x_{\gamma,j}), \sigma^2)\right] - \log p(Y|X_\gamma) \tag{47}$$

$$= \sum_{j=1}^{N} \left[\frac{1}{\sigma\sqrt{2\pi}} - \frac{(u_\beta(x_{\gamma,j}) - y_j)^2}{2\sigma^2}\right] - \log p(Y|X_\gamma) \tag{48}$$

$$= -\frac{\|u_\beta(X_\gamma) - Y\|^2}{2\sigma^2} + \text{constant.} \tag{49}$$

To make (49) tractable, we can apply Laplace's approximation on the posterior. To do so, we perform a Taylor expansion on $\log p(\beta|X_\gamma, Y)$ around the MAP of the distribution. In this case, we would expect the MAP to be at the true inverse parameter β_0, where $\frac{\partial}{\partial\beta} \log p(\beta|X_\gamma, Y) = 0$. Once expanded, this would give

$$\log p(\beta|X_\gamma, Y) \approx \log p(\beta_0|X_\gamma, Y) + (\beta - \beta_0)^\top \left[\nabla_\beta^2 \log p(\beta_0|X_\gamma, Y)\right](\beta - \beta_0) \tag{50}$$

$$= \log p(\beta_0|X_\gamma, Y) - \frac{1}{2\sigma^2}(\beta - \beta_0)^\top \left[\nabla_\beta^2 \frac{\|u_{\beta_0}(X_\gamma) - Y\|^2}{2\sigma^2}\right](\beta - \beta_0). \tag{51}$$

Note that this can also be written as

$$p(\beta|X_\gamma, Y) \approx p(\beta_0|X_\gamma, Y) \exp\left(-(\beta - \beta_0)^\top \left[\nabla_\beta^2 \frac{\|u_{\beta_0}(X_\gamma) - Y\|^2}{2\sigma^2}\right](\beta - \beta_0)\right) \tag{52}$$

which confirms that the Taylor expansion approximates the posterior distribution as a Gaussian distribution with mean $\mu_{\text{Laplace}} = \beta_0$ and covariance matrix $\Sigma_{\text{Laplace}} = \nabla_\beta^2 \frac{\|u_{\beta_0}(X_\gamma) - Y\|^2}{2\sigma^2}$. Since the posterior is approximated as a multivariate Gaussian distribution, it is simple to approximate the entropy of β as distributed by $p(\beta|X_\gamma, Y)$ using the entropy of multivariate Gaussian distribution as

$$\mathbb{H}[\beta|X_\gamma, Y] \approx \frac{1}{2} \log\det \Sigma_{\text{Laplace}} + \frac{M}{2} \log 2\pi e \tag{53}$$

$$= \frac{1}{2} \log\det \nabla_\beta^2 \|u_{\beta_0}(X_\gamma) - Y\|^2 + \text{constant.} \tag{54}$$

Finally, from (27), we can approximate the EIG as

$$\text{EIG}(\gamma) = -\mathbb{E}_{Y' \sim p(Y|X_\gamma)}\left[\mathbb{H}[\beta|X_\gamma, Y = Y']\right] \tag{55}$$

$$= \mathbb{H}[\beta] - \mathbb{E}_{\beta_0 \sim p(\beta), Y' \sim p(Y|\beta_0, X_\gamma)}\left[\mathbb{H}[\beta|X_\gamma, Y = Y']\right] \tag{56}$$

$$\approx \mathbb{H}[\beta] - \frac{1}{2}\mathbb{E}_{\beta_0 \sim p(\beta), Y' \sim p(Y|\beta_0, X_\gamma)}\left[\log\det \nabla_\beta^2 \|u_{\beta_0}(X_\gamma) - Y'\|^2\right] + \text{constant} \tag{57}$$

where (57) uses the approximation of entropy in (54). Note that $\mathbb{H}[\beta]$ is a constant independent of γ and therefore can be ignored.

Note that in the derivation so far, we have assumed that we are able to compute the PDE solution u_β *and* be able to compute how the solution output changes w.r.t. β. This, fortunately, is made possible using PINNs. Specifically, in the likelihood distribution $p(Y|\beta, X_\gamma)$ from (46) and as sampled from in the expectation in (57), we can replace the u_β with a NN $\hat{u}_{\tilde{\theta}(\beta)}$ with parameter $\tilde{\theta}(\beta)$ as defined in Sec. 4.2. Similarly, inside the expectation term of (57), we can replace $u_{\beta_0}(X_\gamma)$ with $\hat{u}_{\tilde{\theta}(\beta_0)}$, and Y' with a noisy reading of $\hat{u}_{\tilde{\theta}(\beta_0)}$. Ultimately, ignoring additive constants, this gives

$$\text{EIG}(\gamma) \approx -\frac{1}{2}\mathbb{E}_{\beta_0 \sim p(\beta), \varepsilon \sim \mathcal{N}(\varepsilon|0, \sigma^2 I)}\left[\log\det \nabla_\beta^2 \left[\|\hat{u}_{\tilde{\theta}(\beta)}(X_\gamma) - \hat{u}_{\tilde{\theta}(\beta_0)}(X_\gamma) + \varepsilon\|^2\right]_{\beta=\beta_0}\right] \tag{58}$$

For simplicity, we can ignore the additive Gaussian noise in the approximation of Y, i.e., ignore the ε term, and the term inside the expectation is the same as that in (21).

While the Laplace approximation has been used in prior ED works (Beck et al., 2018; Alexanderian et al., 2022), our method is novel in that it incorporates PINNs which allows for the resulting criterion involving Laplace approximation to be differentiable and generic enough for any inverse problems which can be modelled using PINNs.

We comment about the assumptions required to arrive at the approximation in (58).

- We assume that the data is generated with random Gaussian noise. This is a standard assumption as done in other IP and ED methods in the literature.
- We assume that the posterior distribution is unimodal. Note that this assumption does not always hold. One example where this assumption does not hold would be in the case where multiple values of β may represent the same inverse parameter (e.g., β represents NN parameterization of an inverse function). In our experiments, we find that the performance of the method remains good regardless. Furthermore, we believe that the issue can be mitigated by considering the problem under some embedding space $\phi(\beta)$ where two embeddings are similar when their parameterizations represent similar functions. This is likely possible by adjusting our criterion to incorporate ϕ through Lagrange inversion theorem, however we will defer this point to a future work.

  Another case where this assumption may not hold is when there are some degeneracy in the inverse problem solution. In this case, the problem cannot be alleviated anyway unless more observations data are acquired (a simple way to think about this is when there is only one observation reading is allowed, and therefore a good inverse parameter solution will not be obtainable regardless of the ED method).
- We assume that the unimodal distribution is maximal at β_0. Given that the distribution is unimodal, this point would likely hold since the pseudo-observation from the NN is already obtained from using inverse parameter of β_0.

E.3.4. Approximation of Hessian in (18)

Instead of computing the Hessian of $\mathcal{L}_{\text{PDE}}$ directly, we can also employ a trick which avoids computing the Hessian directly, but instead approximates the Hessian based on the first-order derivatives.

Suppose we define

$$R(\theta, \beta) = \begin{bmatrix} |X_p|^{-1/2}\big(\mathcal{D}[\hat{u}_\theta, \beta](X_p) - f(X_p)\big) \\ |X_b|^{-1/2}\big(\mathcal{B}[\hat{u}_\theta, \beta](X_b) - g(X_b)\big) \end{bmatrix}. \tag{59}$$

We can see that

$$\mathcal{L}_{\text{PDE}}(\theta, \beta) = \frac{1}{2} R(\theta, \beta)^\top R(\theta, \beta). \tag{60}$$

We can then write

$$\nabla_\theta \mathcal{L}_{\text{PDE}}(\theta, \beta) = \nabla_\theta R(\theta, \beta)^\top R(\theta, \beta) \tag{61}$$

and

$$\nabla_\theta^2 \mathcal{L}_{\text{PDE}}(\theta, \beta) = \nabla_\theta R(\theta, \beta)^\top \nabla_\theta R(\theta, \beta) + \nabla_\theta^2 R(\theta, \beta)^\top R(\theta, \beta). \tag{62}$$

In the case that (θ, β) are obtained after PINN training has converged, we would have $R(\theta, \beta) \approx 0$, which means we can write

$$\nabla_\theta^2 \mathcal{L}_{\text{PDE}}(\theta, \beta) \approx \nabla_\theta R(\theta, \beta)^\top \nabla_\theta R(\theta, \beta). \tag{63}$$

We therefore can approximate the Hessian as used in (18) using first-order derivatives instead.

E.3.5. Limitations of Tolerable Inverse Parameter

A limitation of the criterion is based on the approximation required in (18), which considers the change in the training loss minima as β changes, where the result may not reflect a global minimal for loss functions that are non-convex, such as that for NNs. Nonetheless, we find that this is fine in practice, as demonstrated by our criteria in the experiments and by other works which relies on similar approximations (Koh and Liang, 2017).

F. Complete Algorithm For PIED

We summarize PIED via a pseudocode presented in Alg. 3. The ED procedure consists of three main phases – learning of the shared PINN parameter initialization, the criterion generation phase which consists of performing forward simulations and consequently defining the criteria, and the criterion optimization phase which proceeds to perform constrained continuous optimization on the criterion.

Note that in our framework, the forward simulation only has to be ran once per ED loop, and can all be ran in parallel using packages which allows for parallelization such as `vmap` on JAX. We also find that the forward simulation can be used without explicitly injecting artificial noise, while still giving observation inputs which work well for IPs involving noisy data.

In the criterion optimization phase, we use projected gradient descent to ensure the resulting design parameter is in the bounded space. However, other constrained optimization algorithms could be used as well, e.g., L-BFGS-B. We repeat the optimization loop over many runs due to the potential non-convexity of the criteria, to obtain a better estimate of the optima.

Algorithm 3 PIED

```
    // Learning shared NN parameters
 1: Randomly initialize $\theta_{\text{L-Init}}$
 2: for $s$ rounds do
 3:     Randomly sample $\beta'_1, \dots, \beta'_k$
 4:     for $j = 1, \dots, k$ do
 5:         $\theta'_j \leftarrow$ NN parameter after training $\theta_{\text{L-Init}}$ with training loss $\mathcal{L}_{\text{PDE}}(\theta, \beta'_j)$
 6:     end for
 7:     $\theta_{\text{L-Init}} \leftarrow \frac{1}{k}\sum_{j=1}^{k} \theta'_j$        ▷ Used as initialization for all proceeding forward and inverse PINNs
 8: end for
    // Criterion generation phase
 9: for $i = 1, \dots, M$ do
10:     Randomize inverse parameter $\beta_i$
11:     $\tilde{u}_{\beta_i} \leftarrow \mathbf{F}(\beta_i)$        ▷ Forward simulation
12:     if use FIST criterion then
13:         Define $\hat{\alpha}_i$ to FIST criterion from Alg. 1
14:     else if use MoTE criterion then
15:         Define $\hat{\alpha}_i$ to MoTE criterion from Alg. 2
16:     else if use TIP criterion then
17:         Define $\hat{\alpha}_i$ to TIP criterion from (21)
18:     end if
19: end for
20: Define aggregated criterion $\alpha(X_\gamma) = \frac{1}{N}\sum_{i=1}^{N} \hat{\alpha}_i(X_\gamma)$
    // Criterion optimization phase
21: Initialize $\gamma_{\text{best}} \in \mathcal{S}_\gamma$ randomly
22: repeat
23:     Initialize $\gamma' \in \mathcal{S}_\gamma$ randomly
24:     for $p$ training steps do        ▷ Gradient-based optimization
25:         $\tilde{\gamma}' \leftarrow \gamma' + \eta \nabla_\gamma \alpha(X_{\gamma'})$        ▷ Gradient ascent since the criterion should be maximized
26:         $\gamma' \leftarrow \text{proj}_{\mathcal{S}_\gamma}(\tilde{\gamma}')$        ▷ Perform projection s.t. $\gamma' \in \mathcal{S}_\gamma$
27:     end for
28:     if $\alpha(X_{\gamma'}) > \alpha(X_{\gamma_{\text{best}}})$ then
29:         $\gamma_{\text{best}} \leftarrow \gamma'$
30:     end if
31: until computational limit hit
32: return $\gamma_{\text{best}}$
```

G. Details About The Experimental Setup

G.1. General ED Loop and IP Setup

Our experiment consists of two phases. In the first phase, we perform the ED loop using PIED or with the other benchmarks. Here, we allow a fixed number of forward simulations using PINNs, which the ED methods can query from as many times as it wants. Each ED methods have time restrictions, where they are allowed to run either for a certain duration, or until they have completed some fixed number of iterations.

After the ED methods have selected the optimal design parameters, the same design parameters are used to test on multiple instances of the IP (we run at least 10 of such instances depending on how much computation resources the specific problem requires). In each instance of the IP, we draw a random ground-truth inverse parameter, and generate the observations according to the model and the random ground-truth inverse parameter. The IP is solved using inverse PINNs to obtain a guess of the inverse parameter. For each instance, we can obtain an error score, which measures how different the inverse parameter estimate is from the ground-truth value.

For each problem, we repeat the ED and IP loop five times, where in each time we obtain multiple values for the IP error. In our results, we report the distribution of all the error scores obtained through the percentile values of the error (i.e., p percent of all IP instances using a certain ED methods have errors of at most x), removing some of the extreme values (in the main paper, we remove the top and bottom ten percent, while in the Appendix we show the distribution via a boxplot removing the outliers). This is done since some inverse parameters result in IPs which are easier than others, and to demonstrate the performance of each ED methods across all possible inverse parameters.

G.2. Specific Problem Setup For Each PDE Scenarios

In this section we describe the PDEs used and the problem setup involved. The definitions of the PDE can be found in App. C.1, and this section will be focused on describing the problem setup for each of the PDEs used in our experiments.

1D damped oscillator example. In this case, we assume we know the system follows the PDE as in (22), with some known value of $x_0 \in [0, 1]$ and $v_0 \in [-1, 1]$. We set $M = 1$, and would like to compute the values for $\mu \in [0, 4]$ and $k \in [0, 4]$. In our IP, we are allowed to make three noisy observations at three timesteps $t_1, t_2, t_3 \in [0, 10]$, which can be chosen arbitrarily. We note that while it is unrealistic for these measurements to be made arbitrarily, we do so in order to be able to construct a simple toy example which can be experimented with. The resulting true observation were computed using the closed form solution in (23), and has added noise with variance 10^{-3}. Each ED benchmarks are given 10 minutes to find the optimal design parameter, or are allowed to run at least 5 rounds of gradient descent optimization or 1k rounds of Bayesian optimizaion, whichever takes longer. Each ED are given 10 forward simulations.

1D wave equation example. Assume a system which follows the wave equation given by (24) over the domain $x \in [0, 6]$ and $t \in [0, 6]$. We assume the IC $u(x, 0)$, and the wave velocity in the form

$$v(x) = \begin{cases} 0 & \text{if } x = 0, \\ v_1 & \text{if } 0 < x < 4, \\ v_2 & \text{if } 4 \leq x < 6, \\ 0 & \text{if } x = 6. \end{cases} \tag{64}$$

In this case, $v_1, v_2 \in [0.5, 2]$ are the inverse parameters to be recovered in the IP. In the ED problem, we restrict the points to be placed only at regular time intervals, i.e., following (30) where we restrict $\gamma_1, \gamma_2, \gamma_3 \in [0, 6]$ and let $t \in \{0, 0.2, 0.4, \ldots, 6\}$. The true observations are numerically generated using code from Binder (2021), with outputs interpolated on continuous domain and truncated to the nearest 6 decimal places to simulate cases where measurements can only be made up to a finite precision. Each ED benchmarks are given 60 minutes to find the optimal design parameter (including the forward simulation of PINNs), or are allowed to run at least 3 rounds of gradient descent optimization or 1k rounds of Bayesian optimizaion, whichever takes longer. Each ED methods are given 5 forward simulations.

2D Eikonal equation example. Assume that the system follows the equation specified in (25), and the goal is to recover the values of $v(x)$ for the entire domain. For each IP instance, we draw a random ground truth $v(x)$ using a NN with a random initialization. For the ED problem, the aim is to find 30 random observations from the 2D input domain $[0, 5] \times [0, 5]$

to observe values of $T(x)$. The observation points are only required to be within the input domain. The true observations are generated using PyKonal package (White et al., 2020), with outputs interpolated on continuous domain and truncated to the nearest 3 decimal places to simulate cases where measurements can only be made up to a finite precision Each ED benchmarks are given 30 minutes to find the optimal design parameter (including the forward simulation of PINNs), or are allowed to run at least 3 rounds of gradient descent optimization or 1k rounds of Bayesian optimizaion whichever takes longer. Each ED methods are given 10 forward simulations.

G.3. PINN and PINN Training Hyperparameters

The architectures of the PINNs and other NNs used are listed in Table 2. Note that we only use multi-layer perceptrons in our experiments. The training process hyperparameters for the forward and inverse PINNs are listed in Table 3.

Table 2. Architectures of used NNs

Problem	Depth	Width	Activation	Output transformation
Damped oscillator	6	8	`tanh`	None
1D wave	3	16	`sin`	None
2D Eikonal (modelling u_β)	6	8	`tanh`	$(x,y) \to y\|x-x_0\|$
2D Eikonal (modelling β)	1	16	`sin`	$(x,y) \to \|y\|+0.2$

Table 3. Training hyperparameters

Problem	Training steps	# PDE Col. Pts.	# IC/BC Col. Pts.	Optimizer
Damped oscillator	30k	300	1	`Adam` (lr= 0.01)
1D wave	200k	15k	2k	`L-BFGS`
2D Eikonal	50k	10k	1	`Adam` (lr= 0.001)

G.4. Scoring Metric

To judge how well our ED methods perform, we will use the error $L(\hat{\beta},\beta)$ of the inverse parameter β. When β has finite dimensions (i.e., represented as a scalar value or as a vector value), then the loss is simply the MSE loss, i.e.,

$$L(\hat{\beta},\beta) = \left\|\hat{\beta}-\beta\right\|_2^2. \tag{65}$$

For the case where β is a function, we select some number of test points $\{x_{T,i}\}_{i=1}^{N_{\text{test}}}$ and compute the MSE loss of the estimated function on those test points, i.e.

$$L(\hat{\beta},\beta) = \sum_{i=1}^{N_{\text{test}}} \left\|\hat{\beta}(x_{T,i})-\beta(x_{T,i})\right\|_2^2. \tag{66}$$

G.5. Benchmarks for the Scoring Criterion

In this section we describe a few scoring criteria we use as a benchmark. We first describe the benchmarks which are based on methods of estimating the expected information gain (EIG).

- **Nested Monte Carlo (NMC) estimator (Myung et al., 2013).** The estimator estimates the double integral with nested summations as given in (28).
- **Mutual Information Neural Estimator (MINE) (Belghazi et al., 2018).** The estimator utilizes Donsker-Varadhan representation of the KL Divergence to show that we can provide a lower bound to the EIG as

$$\begin{aligned}\text{EIG}(X_\gamma) &\geq \mathcal{L}_{\text{DV}}(X_\gamma) && (67)\\ &\triangleq \mathbb{E}_{(y,\beta)\sim p(\beta)p(Y|\beta,X_\gamma)}\left[T_\phi(Y,\beta|X_\gamma)\right] - \log \mathbb{E}_{(Y,\beta)\sim p(\beta)p(y|X_\gamma)}\left[e^{T_\phi(Y,\beta|X_\gamma)}\right] && (68)\end{aligned}$$

  where T_ϕ is a parametrized family of functions.

Note that while the estimator $\mathcal{L}_{\text{DV}}(X_\gamma)$ relies on sampling $p(Y|\beta, X_\gamma)$ and $p(Y|X_\gamma)$ directly, this would require running many forward simulations for different β samples. Instead, to sample from these two distributions, we draw M random samples of β and approximate $p(\beta)$ with a mixture of Dirac-delta distributions, i.e.,

$$p(\beta) \approx \hat{p}(\beta) \triangleq \frac{1}{M}\sum_{i=1}^{M}\delta(\beta - \beta_i) \quad \text{where} \quad \beta_1, \ldots, \beta_M \sim p(\beta). \tag{69}$$

In this case, the forward simulation only needs to be ran for M samples of β and the distributions $p(Y|\beta, X_\gamma)$ and $p(Y|X_\gamma)$ can be efficiently approximated.

- **Variational Bayesian Optimal ED (VBOED) estimator (Foster et al., 2019).** The original paper desicribes multiple estimators for the EIG, however we will use the variational marginal estimator. The estimator utilizes the fact that we can provide an upper bound to the EIG as

$$\text{EIG}(X_\gamma) \leq \mathcal{U}_{\text{marg}}(X_\gamma) \triangleq \mathbb{E}_{(Y,\beta)\sim p(\beta)p(Y|\beta,X_\gamma)}\left[\log\frac{p(Y|\beta, X_\gamma)}{q_\phi(Y|X_\gamma)}\right] \tag{70}$$

where q_ϕ is a variational family parametrized by ϕ. To compute the EIG, we find the ϕ which minimizes $\mathcal{U}_{\text{marg}}(X_\gamma)$. Instead of computing the upper bound exactly, we use an empirical estimation based on samples of (Y, β) generated from the PINNs with added noise. Similar to MINE, we approximate $p(\beta)$ with a mixture of Dirac-delta distributions (69) such that only a limited number PINN forward simulations are required.

Note that while Foster et al. (2019) does propose a variational NMC (VNMC) estimator as well, this requires computing $p(Y, \beta|X_\gamma) = p(\beta)p(Y|\beta, X_\gamma)$ for a randomly sampled β. It is not feasible to compute $p(Y|\beta, X_\gamma)$ on the fly since this would require running a costly forward simulation for the randomly sampled β, and therefore the method is not included for this benchmark.

We also use other benchmarks which are not based on the EIG, listed as follows.

- **Random.** The design parameters are chosen randomly.

- **Grid.** The design parameters are chosen such that the sensor readings are placed regularly in some fashion. For the 1D examples, the sensors are placed such that they all regularly spaced out. For the 2D examples, the sensors are placed such that they are shaped in a regular 2D grid with each sides having as equal number of sensors as possible. Note that no optimization is done, but instead the observation input configuration is fixed per problem.

- **Mutual information (MI) (Krause et al., 2008).** The criterion considers the outputs Y_{X_γ} of the chosen observation input X_γ and the outputs Y_{X_t} of some test set X_t, and defines the score to be the mutual information between Y_{X_γ} and Y_{X_t}, i.e.,

$$\alpha_{\text{MI}}(X_\gamma) = \text{MI}(Y_{X_t \setminus X_\gamma}; Y_{X_\gamma}) = \mathbb{H}[Y_{X_t}] - \mathbb{H}[Y_{X_t \setminus X_\gamma}|Y_{X_\gamma}]. \tag{71}$$

In our experiments, we approximate the observation outputs via a Gaussian process (GP) whose kernel is the covariance of the PDE solutions, i.e., $K(x, x') = \text{Cov}_{\beta\sim p(\beta)}[u_\beta(x), u_\beta(x')]$, where we approximate the covariance using the forward simulations. By approximating the output using a GP, the entropies $\mathbb{H}[Y_{X_t \setminus X_\gamma}]$ and $\mathbb{H}[Y_{X_t \setminus X_\gamma}|Y_{X_\gamma}]$ can be written directly in terms of the approximate kernel function. Also, since we do not perform discretization and treat the problem as a combinatorial optimization one due to the additional point constraints, we chose to let $X_t \setminus X_\gamma = X_t$ for simplicity.

We also run the three criteria proposed as the benchmark, listed below.

- **Few-step Inverse Solver Training (Alg. 1).** For each of the trial, we note the value of parameter perturbation σ_p^2 and training steps r used.

- **Model Training Estimate (Alg. 2).** For each trial, we note how many initial training steps r are used, or if we just re-use the NN parameters from the forward PINN for the eNTK.

- **Tolerable Inverse Parameter (21).** For each trial, we note whether we use the approximation for the Hessian as discussed in App. E.3.4.

In the results, we add "`+ LI`" suffix to indicate the benchmark where the learned NN initialization is used for all of the forward and inverse PINNs involved during ED and IP phases. MINE and VBOED are optimized using Bayesian optimization (Frazier, 2018), while NMC and MI are optimized via gradient descent due to criteria which are efficient enough for gradient computations.

G.6. Implementation and Hardware

All of the code were implemented based on the JAX library (Bradbury et al., 2018), which allows for NN training and auto-differentiation of many mathematical modules within. Criteria which are optimized by Bayesian optimization are done so using BoTorch (Balandat et al., 2020), while criteria optimized using gradient-based methods are done so using JAXOpt (Blondel et al., 2021).

For the 1D damped oscillator experiment, each ED loop and IP solves were done on a machine with AMD EPYC 7543 32-Core Processor CPU and NVIDIA RTX A5000 GPU. The remaining experiments were done on AMD EPYC 7763 64-Core Processor CPU and NVIDIA L40 GPU.

H. Additional Experimental Results

H.1. Additional Results From Experiments on Learned NN Initialization

In Fig. 7, we present examples of the learned NN initialization and also its performance for forward PINNs for the 2D Eikonal equation example. We see that this shows similar trends to the examples from before as shown in Fig. 3.

In Fig. 8, we show the test error for an individual forward PINN when performing forward simulation for a value of β, when using and not using a learned NN initialization. We see that when using a learned NN initialization, the test loss typically is already lower than that from random initialization at the start, and also tends towards convergence much faster. Even when the test loss is higher at the start, it is able to catch up to the performance of the randomly initialized PINN under much fewer training steps.

In Fig. 9, we present the results for using a learned NN initialization for inverse PINNs. We see that, similar to the forward PINNs, when using a learned NN initialization, the training loss is either lower than or converges faster than when using a random initialization. We find that the convergence of inverse parameter error shows a less clear trend, which may be because it is not an objective being minimized directly during the PINN training and hence is harder to account for with our chosen NN initialization.

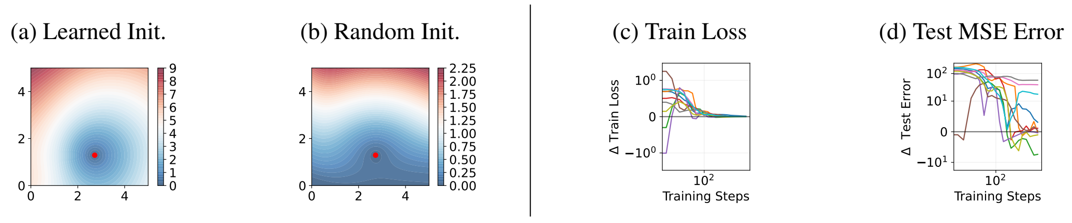

Figure 7. Results for learning a NN initialization for PINNs trained on 2D Eikonal equation case. The interpretation is the same for that in Fig. 3.

H.2. Additional Finite-Dimensional Inverse Problem Experiment Results

In Figs. 10 and 11, we present the IP error for all benchmarks ran for the damped oscillator example and the 1D wave equation example respectively. Note that for the 1D wave equation case we ran the experiments on fewer benchmarks due to time constraints. We can make several observations from the results.

- In our criteria, using a learned NN initialization is able to give slightly better performances, as seen in the damped oscillator case and to a lesser extent in the 1D wave equation case. We believe that the difference is not as dramatic in

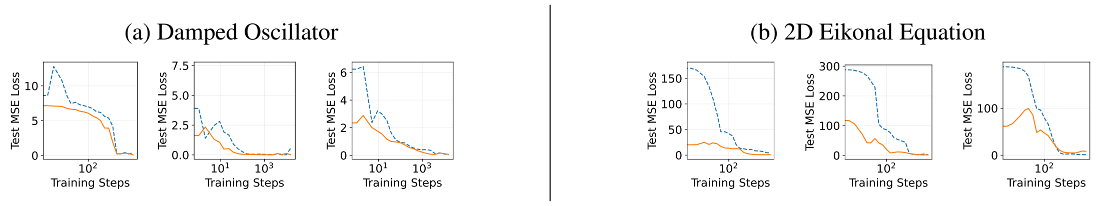

Figure 8. Examples of test error of forward PINNs for each problems for different values of β. Each plot represents the PINN training for one random random value of β. The dotted blue lines represent when the PINN is initialized with a random NN parameters, while the solid yellow lines represent when the PINN is initialized form the learned initialization.

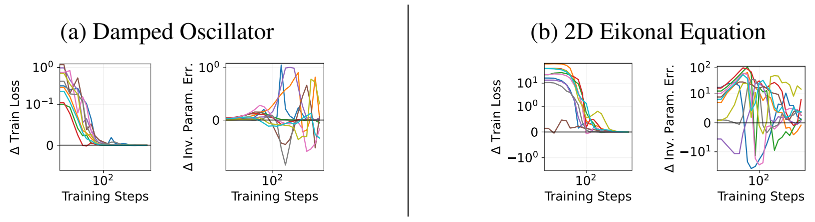

Figure 9. Performances of the inverse PINN when using a learned NN initialization for each of the IP examples. In each case, we present both the training loss for the inverse PINN (left plots) and of the inverse parameter error (right plots). In all plots, we show the difference in loss/error value between using learned NN initialization as compared to using a random initialization.

our experiments since in the PINN training, we do allow for enough training steps already such that the model will likely converge anyway, and so the effects from using a good initialization is less noticeable.

- Performance of the FIST criterion can vary differently according to the noise level and the amount of inverse PINN training that is done. This means that even though FIST can potentailly reach good performance in some instances (as seen in the damped oscillator example), the performance is too sensitive to hyperparameters for it to be practical.
- We see that the optimal observation input may not be trivial even for simple examples. For example, for the damped oscillator case, it is less beneficial to select points which are spread out, as seen in the case for Grid, which purposefully select points that are spread out, having poor performances in this experiment.

  To elaborate, we present observation selection for TIP and Grid method in Fig. 12. As seen in the Grid case, even though the trained inverse PINN may result in a function which is close to the true solution anyway, the learned inverse parameter is more spread out than in the case of TIP for multiple inverse solver runs with the same β.

H.3. Additional Function-Valued Inverse Problem Experiment Results

In Fig. 13, we present results for the additional benchmarks in the 2D Eikonal equation problem. We see that we notice similar trends to the finite-dimensional inverse problem case, where our criteria outperforms most benchmarks and improvements from using a learned NN initialization is more clearly present. As discussed in the main paper, FIST method performs poorly likely due to suboptimal hyperparameters. We also see that TIP shows strong performance compared to the other criteria.

We note that in this case, the Grid benchmark also shows very strong performances. We note that this is likely due to the fact that the inverse function to be estimated is a local quantity, where small changes in the inverse function in one region does not cause large effects on the PDE solution across the whole input domain. In these cases, it is beneficial to select points that are more spread out, which is what the Grid benchmark does. We find that this is not always the case, as demonstrated in the IPs with finite-dimensional inverse parameters, meaning that using a method that can make these judgements on observation inputs automatically is still beneficial.

H.4. Discussions on General Limitations and Societal Impacts

In our experiments, we have mostly conducted experiments based on vanilla PINN architectures. We have not done verification of whether PIED works well for other more complex architectures such as physics-informed deep operators.

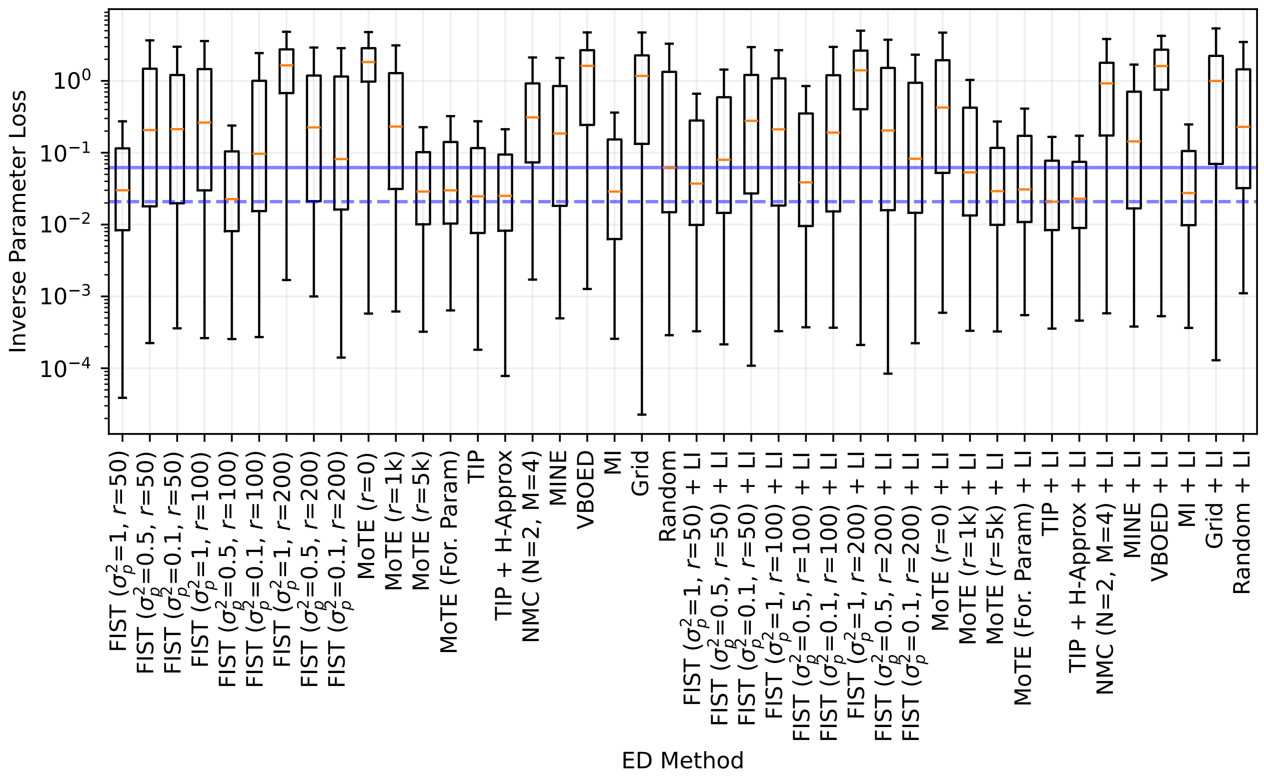

Figure 10. Distribution of error of estimated inverse parameter for each ED methods in the damped oscillator experiments. The thick blue line shows the median error for the Random benchmark, while the dashed line shows the lowest median error obtained.

Future work on this area would be interesting.

We find that by design, FIST and MoTE scales with more training, which means that more computational resources may be required for better results, which we find in our experiments. In some one-shot settings where it is critical to get optimal results, it may be worth dedicating more resources for the ED problems. Meanwhile, TIP and MoTE which reuses the forward PINN parameters do not exhibit this.

The work also relies on the use of PINNs as the forward simulators and IP solvers, and are not applicable to other forward simulators or IP solvers. While PINNs are well-suited for both tasks, they also still pose practical problems such as difficulty in training for certain problems. The problem can be mitigated through more careful selection of collocation points (Wu et al., 2023; Lau et al., 2024), which will be interesting to consider in future works to further boost the performance of PIED.

We believe the work has minimal negative and significant positive societal impacts, since they can be used in many science and engineering applications where costs of data collection from experiments can be prohibitive. This means that the cost barrier in performing effective scientific experiments can be lowered allowing for further scientific discoveries. We note that our work could potentially be applied for a range of scientific research, which may include unethical or harmful research done by malicious actors. However, this risk applies to all tools that accelerate scientific progress, and we believe that existing policies and measures guarding against these risks are sufficient.

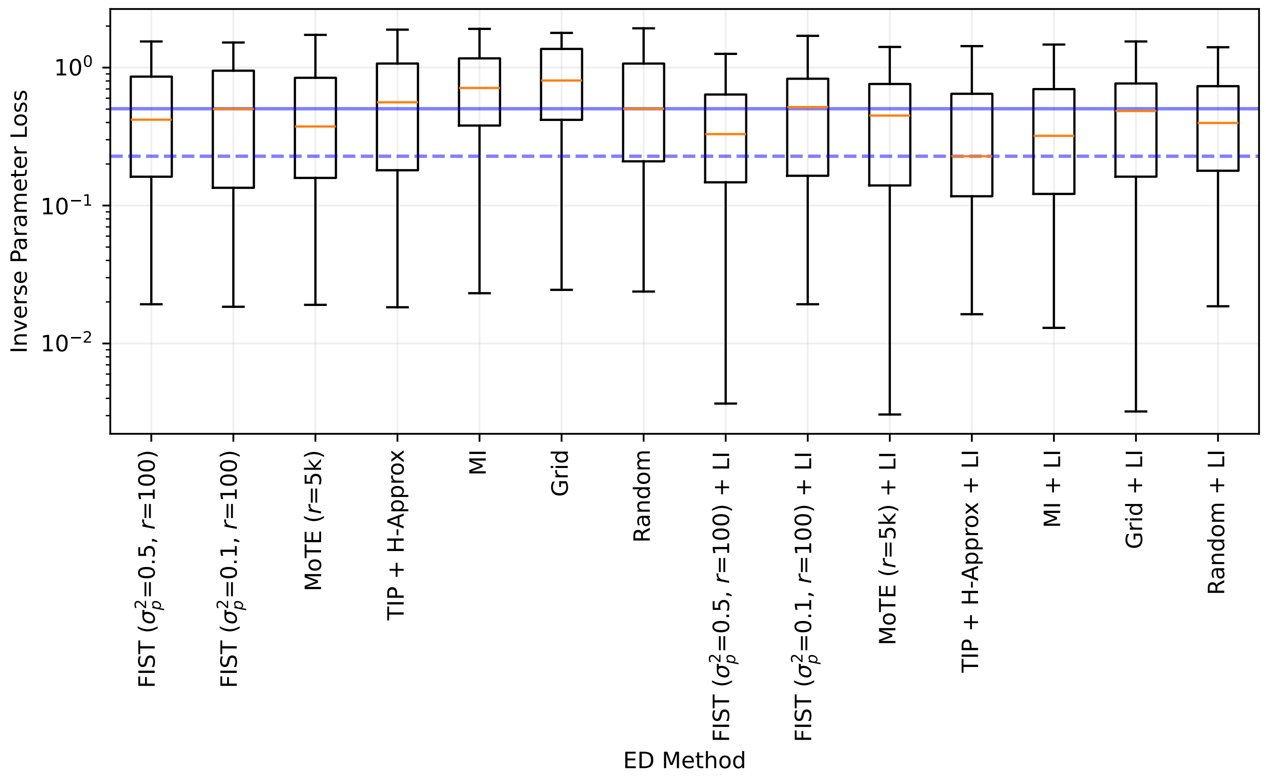

Figure 11. Distribution of error of estimated inverse parameter for each ED methods in the 1D wave equation experiments. The thick blue line shows the median error for the Random benchmark, while the dashed line shows the lowest median error obtained.

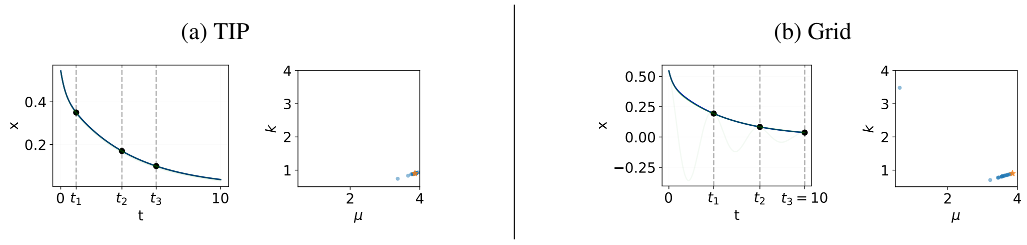

Figure 12. Results from the inverse problems on the damped oscillator problem. We demonstrate the observation input selection for MoTE (Fig. 12a) and Grid (Fig. 12b) methods on a single ground truth inverse parameter but on multiple IP runs. In the left plots, we present the output of the inverse PINNs (green lines) as compared to the true underlying function (blue line). In the right plots, we present the estimated inverse parameter (blue dots) as compared to the true ground truth inverse parameter (orange dot).

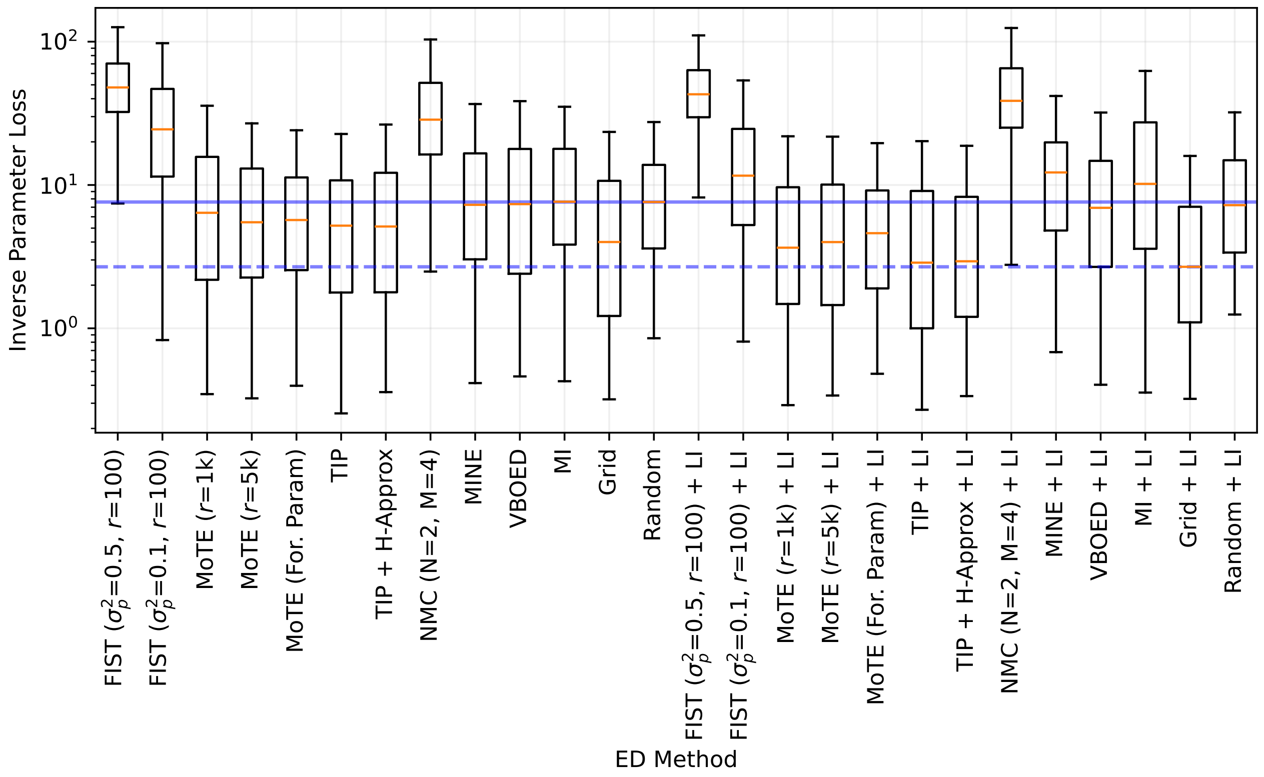

Figure 13. Distribution of error of estimated inverse parameter for each ED methods in the 2D Eikonal equation experiments. The thick blue line shows the median error for the Random benchmark, while the dashed line shows the lowest median error obtained.