Deep Optimal Sensor Placement for Black Box Stochastic Simulations

Paula Cordero Encinar[1] **Tobias Schröder**[1] **Peter Yatsyshin**[2] **Andrew B. Duncan**[1,2]
[1]Imperial College London
[2] Alan Turing Institute

Abstract

Selecting cost-effective optimal sensor configurations for subsequent inference of parameters in black-box stochastic systems faces significant computational barriers. We propose a novel and robust approach, modelling the joint distribution over input parameters and solution with a joint energy-based model, trained on simulation data. Unlike existing simulation-based inference approaches, which must be tied to a specific set of point evaluations, we learn a functional representation of parameters and solution. This is used as a resolution-independent plug-and-play surrogate for the joint distribution, which can be conditioned over any set of points, permitting an efficient approach to sensor placement. We demonstrate the validity of our framework on a variety of stochastic problems, showing that our method provides highly informative sensor locations at a lower computational cost compared to conventional approaches.

1 INTRODUCTION

A common challenge across many areas of science and engineering is recovering an unobserved parameter or input from noisy, indirect observations. Examples arise in imaging (e.g. in-painting and up-scaling), weather forecasting and oceanography (flow reconstruction), material design (molecular force-field reconstruction) and medicine (computed tomography).

Such *inverse problems* are often ill-posed, meaning that there may be several choices of model parameters which are consistent with the observations, or that the output is highly sensitive to errors in the parameter. Many highly relevant inverse problems are also large-scale, where the map from input parameters to output is sufficiently complex to be effectively *black-box*, and its evaluation is highly computationally intensive.

An inverse problem can be formulated as the following system of equations

$$u = \mathcal{G}(\kappa)$$
$$\mathbf{y} = \mathcal{O}(u) + \eta,$$

where the *forward problem* $\mathcal{G}$ is an associated mathematical model, mapping an (unobserved) input parameter κ to a spatially-varying field u, e.g. the solution of a partial differential equation (PDE). In typical settings, we do not observe u completely, but rather make partial observations $\mathbf{y}$ through an *observation operator* $\mathcal{O}$ subject to noise η.

For this work we are interested in settings where the relationship between κ and u is non-deterministic, so that $\mathcal{G}$ possesses intrinsic stochasticity. This challenge arises in settings where the forward model is not a complete representation of reality, and additional (unobserved) noise sources are introduced to characterise external/environmental effects. Such forward models are typically formulated as stochastic partial differential equations (SPDEs).

The Bayesian approach to inverse problems (Stuart, 2010) is a systematic method for uncertainty quantification of parametric estimates in which the parameter and the observations are viewed as coupled-random variables. By placing a prior for the distribution of the parameter κ, Bayes' rule yields a posterior distribution for κ given the observations, thus providing a probabilistic method to solving inverse problems.

The quality of the inference will depend on how informative the observations are which is characterised by the observation operator $\mathcal{O}$ which is typically a vector of point-wise evaluations of u at a fixed set of *sensor locations* within the problem domain. In settings where we have control over sensor placement, we have a strong incentive to choose them optimally, i.e. to be as informative of κ as possible.

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

Table 1: Comparison of scope of Functional Neural Couplings (ours) and existing methods.

Methods	Direct PDE solves	Neural Operator surrogate	Neural Operator w/ oracle noise	Functional Neural Coupling **(ours)**
Low-cost evaluation	✗	✔	✔	✔
Low-cost inversion	✗	✔	✔	✔
Supports sensor placement pipeline	✔	✔	✔	✔
Supports stochastic PDEs	✔	✗	✔	✔
Tractable likelihood	✗	✗	✗	✔

While optimal sensor placement has been widely studied in the scientific computing literature (Chaloner and Verdinelli, 1995; Rainforth et al., 2024), established approaches are infeasible for larger-scale problems. Typical approaches to sensor placement involve a nested Markov Chain Monte Carlo (MCMC) approach to perform Bayesian Experimental Design (BED). In the setting of a PDE-governed inverse problem, each MCMC step necessitates at least one simulation of the underlying PDE, meaning that the process quickly becomes computationally infeasible (Alexanderian, 2021).

A natural strategy would be to perform optimal sensor placement over a surrogate model (Gramacy, 2020), learnt from forward model evaluations. However, such approaches are not applicable if the parameter is functional or if the forward operator is intrinsically stochastic. In the latter case, the likelihood becomes intractable due to the introduction of auxiliary noise variables, and one must resort to expensive pseudo-marginal MCMC methods (Andrieu and Roberts, 2009).

To overcome these limitations we propose *Functional Neural Couplings*, a novel probabilistic model of the relationship between inputs and outputs of a stochastic forward problem. Functional Neural Couplings provide accurate likelihood evaluations for κ given noisy observations of u, which enables low-cost solution of both the forward and inverse-problem, despite requiring a single training phase. Specifically, we make the following contributions:

1. We introduce *Functional Neural Couplings* (FNC), a probabilistic model of the joint distribution of a functional parameter κ and the functional solution of the forward problem u as a joint Energy-Based Model (EBM).
2. We leverage implicit neural representations (INR) to express functional data as low-dimensional latent codes. This introduces a new way to extend the scope of energy-based models to functional data on diverse geometries and makes our approach resolution independent, i.e. the model can be queried from functions evaluated at arbitrary evaluation points.
3. Finally, we use the Functional Neural Coupling framework to introduce the first computationally feasible method for deep optimal sensor placement for stochastic inverse problems with black-box noise.

1.1 Related Work

Neural Operators Neural Operators are a class of deep learning architectures designed to learn maps between infinite-dimensional function spaces. As data-driven methods, they do not require any knowledge of the underlying PDE. Additionally, they are resolution invariant in the sense that they can generate predictions at any resolution. Examples include DeepONets and Fourier Neural Operators (FNO) (Kovachki et al., 2021; Li et al., 2021). A related approach, CORAL (Serrano et al., 2023) uses an encode-process-decode structure to learn functional operators between functions on non-uniform geometries by leveraging INRs as in our framework.

Some works have leveraged Neural Operators for solving inverse problems, e.g. Molinaro et al. (2023). Long and Zhe (2024) adopts a Bayesian approach by introducing an invertible FNO enhanced with a VAE that allows for posterior inference. However, these methods assume only observational noise and cannot account for stochasticity in the forward operator.

Generative Models On Function Spaces Recent work has sought to extend generative models to distributions on function spaces, which can therefore capture stochastic relationships between the input parameter κ and $\mathcal{G}(\kappa)$. Rahman et al. (2022) propose a Generative Adversarial Neural Operator for learning function data distributions. Baldassari et al. (2023), Franzese et al. (2023), Kerrigan et al. (2023), Lim et al. (2023a) and Pidstrigach et al. (2023) generalise diffusion models to operate directly on a Hilbert space. In contrast to our approach, these methods do not immediately yield a likelihood which can be used for downstream tasks. Mishra et al. (2022) propose a Variational Autoencoder approach which employs a function space decoder. Similar approaches include Neural Processes (Garnelo et al., 2018), Energy-Based Processes (Yang

et al., 2020) and Functional EBMs (Lim et al., 2023b).

Inverse Problems and Sensor Placement Methods that use black-box simulators for the statistical inference of underlying finite dimensional parameters are well-established and typically referred to as likelihood-free or simulation-based inference methods (Cranmer et al., 2020). In recent years, neural networks have been employed to learn synthetic likelihood surrogates with normalising flows (Papamakarios et al., 2019) and energy-based models (Glaser et al., 2022) as in our work. However, these approaches neither apply to functional data nor address sensor placement.

The problem of optimising sensor locations is typically approached as a combinatorial optimisation task by discretising the domain into a finite grid (Barthorpe and Worden, 2020; Yuen et al., 2001; Andersson et al., 2023), while we are able to select sensor placement sites across the entire problem domain.

Sensor placement in the context of inverse PDE problems has mainly been addressed using traditional MCMC approaches (Andrieu and Roberts, 2009; Alexanderian, 2021). The number of simulations of the forward system is orders of magnitude higher in classical MCMC-based approaches than in our approach and thus not suitable for complex situations with expensive simulators such as stochastic PDEs.

2 PROBLEM SETTING

We consider a stochastic forward problem $\kappa \to u = \mathcal{G}(\kappa, \omega)$, where the random variable ω captures the intrinsic stochasticity. The function-valued parameter $\kappa \in \mathcal{A}$ is defined on a domain $\Omega \subseteq \mathbb{R}^{d_x}$. For a forward model which involves the solution of a (S)PDE, the parameter κ could be an initial condition, boundary condition, spatially-varying coefficient field or forcing term, but our framework is not restricted to these settings, or SPDEs more broadly.

At training time we have access to a black-box stochastic simulator of the forward dynamics to obtain a solution $u = \mathcal{G}(\kappa, \omega)$ for any given $\kappa \in \mathcal{A}$. To reflect real-world settings, the realisation or distribution of the stochastic contribution ω is assumed to be *unknown* to us at all times.

Objective At test time, the objective is to find optimal sensor positions $\boldsymbol{\xi}$ which maximise the information gained on $\kappa \in \mathcal{A}$ through inference based on noisy observations $\mathbf{y} = \mathcal{O}(u) + \eta = (u(\boldsymbol{\xi}_j) + \eta_j)_{j=1}^D$ at sensor locations $\{\boldsymbol{\xi}_j\}_{j=1}^D$. Here, the observational noise $\boldsymbol{\eta}$ is a property of the sensor and is different from the intrinsic stochasticity of the system.

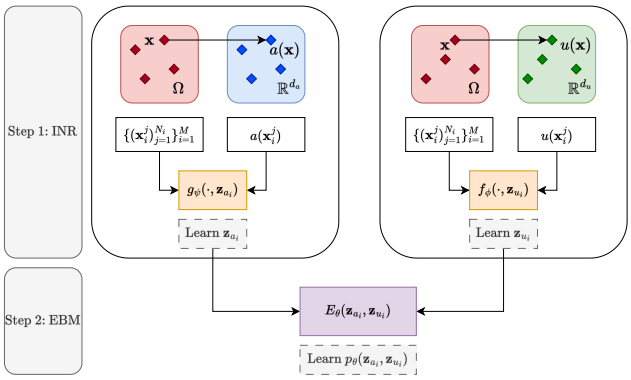


Figure 1: Workflow for the training of the INR and joint energy-based model. Layout based on Serrano et al. (2023).

3 FUNCTIONAL NEURAL COUPLINGS

We propose to approximate the solution operator $\mathcal{G}$ using a probabilistic model p_θ that learns the joint distribution of (κ, u) such that high likelihood regions of p_θ correspond to solutions $u = \mathcal{G}(\kappa)$. To do so, using training data from simulations $\{(\kappa_i(\mathbf{x}_j^i), u_i(\mathbf{x}_j^i))_{j=1}^{N_i}\}_{i=1}^M$, $\mathbf{x}_j^i \in \Omega$, we encode the functions κ_i and u_i into finite-dimensional latent codes $\mathbf{z}_{\kappa,i} \in \mathbb{R}^{d_{z_\kappa}}, \mathbf{z}_{u,i} \in \mathbb{R}^{d_{z_u}}$ by employing implicit neural representations (Dupont et al., 2022). Critically, no evaluation grid has to be defined for this step. On the finite-dimensional representation space, we learn the joint distribution of the latent codes using a joint energy-based model $p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u) \propto \exp(-E_\theta(\mathbf{z}_\kappa, \mathbf{z}_u))$. The neural network architecture of the joint energy-based model is described in the Appendix. Our training workflow is visualised in Figure 1.

3.1 Learning Implicit Neural Representations

Following COIN++ (Dupont et al., 2022), we compress the functional data points $(\kappa_i)_{i=1}^M$ and $(u_i)_{i=1}^M$ into implicit neural representations defined by $\kappa_i(\cdot) = g_\psi(\cdot, \mathbf{z}_{\kappa,i})$, $\mathbf{z}_{\kappa,i} \in \mathbb{R}^{d_{z_\kappa}}$, and $u_i(\cdot) = f_\phi(\cdot, \mathbf{z}_{u,i})$, $\mathbf{z}_{u,i} \in \mathbb{R}^{d_{z_u}}$. We train the neural representations by minimising the mean square error between the function value and the neural network prediction at the evaluation points. Each layer of g_ψ and f_ϕ takes the form of a SIREN layer $\sin(\omega_0(W\mathbf{h} + \mathbf{b} + \boldsymbol{\beta}))$ (Sitzmann et al., 2020). While the weights and biases $W, \mathbf{b}$ of each layer are shared among all data points, the shifts $\boldsymbol{\beta}$ depend differentiably on each data point latent code $\mathbf{z}$ and are trained individually for each functional data point. That is, the training consists of a double optimisation loop, the inner loop updates the modulations $\mathbf{z}_{\kappa,i}, \mathbf{z}_{u,i}$

while the outer loop updates the shared parameters, $\psi = \{W_\psi, \mathbf{b}_\psi\}, \phi = \{W_\phi, \mathbf{b}_\phi\}$. This produces highly compressed latent representations $(\mathbf{z}_{\kappa,i}, \mathbf{z}_{u,i})_{i=1}^M$, which can be decoded efficiently by applying the maps g_ψ and f_ϕ , respectively. Thus, the INR component is crucial to effectively learn functional relationships, without being limited to a particular discretisation of the domain.

Due to the generalisation capabilities of the INR, we train it on datasets of increasing size until the reconstruction error stabilises on a validation dataset. It is important to remark that this is a practical choice to optimise one of the most computationally expensive steps — learning the INR — without sacrificing performance. The latent codes $\mathbf{z}$ are then computed for the entire dataset, keeping the shared parameters of the INR fixed.

3.2 Functional Neural Couplings with Joint Energy-Based Models

Energy-based models (EBMs) (Lecun et al., 2006) are unnormalised statistical models of the form $\exp(-E_\theta)$, where the energy-function E_θ is typically modelled with a scalar-valued neural network. We assume lossless compression in the INR and learn the joint distribution of κ and $u = \mathcal{G}(\kappa)$ as a joint energy-based model over tuples $\mathbf{z}_i = (\mathbf{z}_{\kappa,i}, \mathbf{z}_{u,i})$. Since unnormalised models are not amenable to optimisation with maximum likelihood estimation, we explore training the model with contrastive divergence (Hinton et al., 2006), score-based methods (Hyvärinen, 2005; Vincent, 2011; Song and Ermon, 2020) and energy discrepancy (Schröder et al., 2023), achieving the best results with energy discrepancy. Note that the joint energy-based model component plays a central role in modelling the stochastic relationship between parameters and solutions.

4 SENSOR PLACEMENT WITH BAYESIAN EXPERIMENTAL DESIGN

The goal is to determine optimal sparse sensor placement positions $\boldsymbol{\xi} = \{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_D\} \subset \Omega$ for the inference of $(\kappa, u = \mathcal{G}(\kappa))$ based on $\mathbf{y} = u(\boldsymbol{\xi}) + \boldsymbol{\eta}$. We assess the utility of a sensor position by calculating the expected information gain over the prior as measured by relative entropy, i.e. we use the utility function $U(\boldsymbol{\xi}) := \mathbb{E}_{p(\mathbf{y}|\boldsymbol{\xi})} D_{\mathrm{KL}}(p(\mathbf{z}_\kappa, \mathbf{z}_u|\mathbf{y}, \boldsymbol{\xi}) \,\|\, p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u))$, which can be evaluated at any point $\boldsymbol{\xi}$ thanks to the INR component. Using Bayes theorem, the utility can be rewritten as

$$U(\boldsymbol{\xi}) = \mathbb{E}_{p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u) p(\mathbf{y}|\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{\xi})} \left[\log \frac{p(\mathbf{y}|\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{\xi})}{p(\mathbf{y}|\boldsymbol{\xi})} \right]. \quad (1)$$

Since $p(\mathbf{y}|\boldsymbol{\xi})$ is unknown, a naive Monte Carlo estimation provides a nested Monte Carlo estimator (Rainforth et al., 2018), which approximates the inner and outer integrals of the utility but suffers from a slow rate of convergence. Therefore, following on Foster et al. (2020), we use the prior contrastive estimation (PCE) bound given by

$$\hat{U}_{\mathrm{PCE}}(\boldsymbol{\xi}) = \mathbb{E} \left[\log \frac{p(\mathbf{y}|\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi})}{\frac{1}{L+1} \sum_{l=0}^{L} p(\mathbf{y}|\mathbf{z}_{\kappa,l}, \mathbf{z}_{u,l}, \boldsymbol{\xi})} \right]$$

where the expectation is computed over the distribution $p_\theta(\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}) p(\mathbf{y}|\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi}) p_\theta(\mathbf{z}_{\kappa,1:L}, \mathbf{z}_{u,1:L})$. This alternative estimator significantly speeds up training and is a valid lower bound of the utility which becomes tight as $L \to \infty$ (Foster et al., 2020). Further details are given in the Appendix.

The actual selection of optimal locations can be conducted sequentially, which leads to Bayesian adaptive design (BAD) (Rainforth et al., 2024) for sensor placement. This framework iteratively places sensors by incorporating observations of the solution from previously identified optimal locations, using them to guide the selection of the best sensor positions in subsequent steps. In settings where multiple measurements can be taken in parallel, each step of the optimisation can select a batch of optimal sensor placement positions instead of just one. More specifically, considering the sequence of locations and outcomes up to step t of the experiment, denoted by $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{t-1}\}$ and $\{\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}\}$, respectively, we aim to maximise the utility given the history $h_{t-1} = \{(\boldsymbol{\xi}_k, \mathbf{y}_k)\}_{k=1}^{t-1}$,

$$U(\boldsymbol{\xi}_t|h_{t-1}) = \mathbb{E} \left[\log \frac{p(\mathbf{y}|\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{\xi}_t, h_{t-1})}{p(\mathbf{y}|\boldsymbol{\xi}_t, h_{t-1})} \right],$$

where $h_0 = \emptyset$ and the expectation is over $p(\mathbf{z}_\kappa, \mathbf{z}_u|h_{t-1}) p(\mathbf{y}_t|\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{\xi}_t, h_{t-1})$. This can also be interpreted as the utility in Eq. (1) with an updated prior and likelihood. However, in most cases $\mathbf{y}_t$ is independent of h_{t-1} given $(\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{\xi}_t)$, thus only the prior needs to be updated.

4.1 Inference from sparse observations

At inference time, we have noisy observations of the system at the selected optimal locations $\mathcal{D} = \{(\boldsymbol{\xi}_j, \mathbf{y}_j)\}_{j=1}^D$ with $\mathbf{y}_j = u(\boldsymbol{\xi}_j) + \eta_j$, where the observational noise $\eta_j \sim \mathcal{N}(0, \sigma^2)$ is to be distinguished from the random component inherent in the forward model which is captured by the energy-based model. The posterior distribution of the latent representations $(\mathbf{z}_\kappa, \mathbf{z}_u)$

conditioned on the observed data is given by

$$p(\mathbf{z}_\kappa, \mathbf{z}_u|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{z}_\kappa, \mathbf{z}_u)p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u) = \prod_{j=1}^{n} p(\boldsymbol{\xi}_j, \mathbf{y}_j|\mathbf{z}_\kappa, \mathbf{z}_u)p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u).$$

The noise assumption together with the underlying forward model, results in $p(\boldsymbol{\xi}_j, \mathbf{y}_j|\mathbf{z}_\kappa, \mathbf{z}_u) = \mathcal{N}(\mathbf{y}_j; f_\phi(\boldsymbol{\xi}_j, \mathbf{z}_u), \sigma^2)$. The desired parameter-solution pair $(\kappa, u = \mathcal{G}(\kappa))$ can now be sampled from the posterior using stochastic gradient Langevin dynamics (Welling and Teh, 2011).

5 NUMERICAL EXPERIMENTS

In this section, we test the performance of the proposed Functional Neural Coupling framework for optimal sensor placement in stochastic versions of a 1D boundary value problem, the 2D Darcy flow problem and the 2D Navier-Stokes equation. We compare our method with approaches based on Fourier neural operators. Our training data consists of M pairs of parameters and their corresponding solutions evaluated at N_i point observations that can be different across the M function realisations, that is, $\{(\kappa_i(\mathbf{x}_j^i), u_i(\mathbf{x}_j^i))_{j=1}^{N_i}\}_{i=1}^{M}$ with $\mathbf{x}_j^i \in \Omega$. While the method can handle functional parameters through the INR encoding, we assume for simplicity in the first presented example that κ is parametrised by a real-valued vector of finite dimension. We emphasise that direct evaluations of $\mathcal{G}$ are only required to train the INRs and the energy-based model. Once trained, no further evaluations of $\mathcal{G}$ are required, and the INRs and EBM are used exclusively for optimal sensor placement and inference, drastically reducing the computational burden when $\mathcal{G}$ is a computationally intensive simulator model. Additional experimental details (dataset generation, implementation, run times) and further numerical experiments are included in the Appendix.

Benchmarks We compare our method for optimal sensor placement against using a Fourier Neural Operator (FNO) surrogate for the forward model (Li et al., 2021). In this case, the posterior distribution of the parameter given observations of the solution is obtained following Section 5.5 of Li et al. (2021), which is subsequently used to perform sensor placement. As neural operators can only learn deterministic maps, they fail to incorporate the effect that a spatio-temporal external random signal has on the system described. To provide a gold standard baseline, we additionally consider the FNO with oracle noise from Salvi et al. (2022) which assumes that the driving noise of the stochastic model, that we previously denoted as ω, is observed and taken as input to the model. The FNO with oracle noise is an unattainable baseline in practice as it has the benefit of having full knowledge of the noise driving the forward model. In contrast, our approach does not rely on observing the driving noise ω, which is typically the case in real-world situations where the system's intrinsic stochasticity cannot be directly measured. For each surrogate method (ours, FNO and FNO with oracle noise), we determine optimal sensor locations using both adaptive and batch non-adaptive approaches. We compare the efficiency of optimally selected positions with the use of points from a Quasi Monte Carlo sequence (Niederreiter, 1992) within the domain.

Note that we do not compare our approach with traditional methods for sensor placement, as our objective is to develop an efficient, lightweight framework for Bayesian sensor placement and classical approaches are computationally prohibitive or infeasible for the type of problems considered, often requiring run times on the order of several days. Table 1 offers a clear comparison of the key advantages of our approach over other methods, highlighting its distinctive characteristics.

5.1 Boundary Value Problems In 1D

Consider the boundary value problem (BVP) on the interval $[-1, 1]$ given by the non-linear PDE

$$u''(x) - u^2(x)u'(x) = f(x), \quad u(-1) = X_a, \;\; u(1) = X_b,$$
$$f(x) = -\pi^2 \sin(\pi x) - \pi \cos(\pi x) \sin^2(\pi x),$$
$$X_a \sim \mathcal{N}(a, 0.3^2), \;\; X_b \sim \text{Unif}(b - 0.3, b + 0.4),$$

where $a, b \sim \text{Unif}(-3, 3)$. The training data consists of pairs of parameters (a, b) and their corresponding solution for a realisation of the random variables X_a and X_b. Once trained the INR and joint energy-based models, we carry out a toy sensor placement task, in which we aim to select two optimal locations. We observe that, using our Functional Neural Coupling surrogate, posterior samples based on information from batch non-adaptive optimal locations match closer the ground truth compared to those from optimal adaptive locations or points of a random sequence (see Figure 2 for the case with true parameters $a = -2$, $b = 1.5$). Table 2 shows the mean and standard deviation of the relative L^2 error norm across 100 experiments with different random values for (a, b) for the different methods. The performance of our approach is comparable to that of FNO with oracle noise, which has complete knowledge of the intrinsic stochasticity. In addition, our method consistently exhibits a lower standard deviation compared to the other approaches.

Table 2: Relative L^2 error norm for the posterior mean of the solution and MSE for the boundary conditions a, b for sensor placement on the BVP. We average over 100 experiments with different random values for (a, b).

Method	Design points	$\|\widehat{u} - u_{\text{tr}}\|^2 / \|u_{\text{tr}}\|^2$	MSE($\hat{a}$)	MSE($\hat{b}$)
Functional Neural Coupling (Ours)	Adaptive BED	0.124 ± 0.101	0.135 ± 0.099	1.441 ± 1.003
	Batch non-adaptive BED	$\mathbf{0.097 \pm 0.081}$	$\mathbf{0.103 \pm 0.193}$	$\mathbf{1.074 \pm 0.906}$
	Random sequence	0.331 ± 0.259	0.476 ± 0.888	4.162 ± 3.885
FNO surrogate (Li et al., 2021)	Adaptive BED	0.137 ± 0.308	0.441 ± 1.337	1.224 ± 2.084
	Batch non-adaptive BED	0.125 ± 0.255	0.428 ± 1.239	1.111 ± 1.785
	Random sequence	0.251 ± 0.507	0.484 ± 0.947	3.839 ± 7.457
FNO w/ oracle noise surrogate (Salvi et al., 2022)	Adaptive BED	0.116 ± 0.250	$\underline{0.021 \pm 0.058}$	1.580 ± 2.888
	Batch non-adaptive BED	$\underline{0.090 \pm 0.131}$	0.041 ± 0.105	$\underline{1.046 \pm 1.620}$
	Random sequence	0.356 ± 0.613	0.494 ± 1.412	8.372 ± 11.856

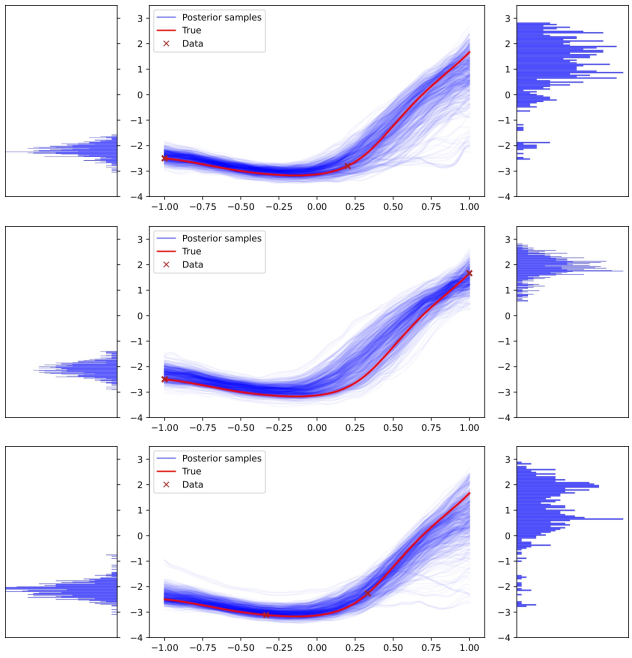


Figure 2: Sensor placement task for the BVP using our Functional Neural Coupling surrogate. Posterior samples based on two optimal design points, adaptive (top) and batch non-adaptive (middle) versus two points from a random sequence (bottom). True values $a = -2$, $b = 1.5$.

5.2 Steady-State Diffusion In 2D

We consider learning the diffusion coefficient κ of the stochastic 2D Darcy flow equation defined by the PDE $-\nabla \cdot \big(\kappa(\mathbf{x})\nabla u(\mathbf{x})\big) = f(\mathbf{x}) + \alpha\omega$ with domain $\mathbf{x} \in \Omega = [0, 1]^2$ and Dirichlet boundary conditions $u|_{\partial\Omega} = 0$. The force term is of the form $f(x) = 0.5$, ω is space white noise and the diffusion coefficient κ is generated as the push-forward of a Gaussian random field.

We perform a sensor placement task in which, based on five initial observations (intentionally chosen in non informative places) of a randomly chosen solution, we maximise the estimated utility function to find optimal locations for fifteen additional measurement sites.

Figures 3 and 4 present the inference results of our Functional Neural Coupling surrogate, comparing sensor placement at selected locations using the optimal adaptive method, the batch non-adaptive approach, and points from a Quasi Monte Carlo sequence within the domain. Qualitatively, the results obtained with adaptive sensor placements better match the ground truth both for the solution and the diffusion coefficient. Table 3 shows that the relative L^2 error norms between the ground truth and the posterior mean for the solution and the diffusion coefficient averaged over 50 sensor placement experiments. It is important to note that no errors are reported for the inference based on Quasi-Monte Carlo points, as Quasi-Monte Carlo sequences are deterministic. We observe that, across different types of sensor locations, our Functional Neural Coupling surrogate outperforms the FNO surrogate, which was trained with the same data, and achieves performance close to that of the FNO with oracle noise surrogate, which has complete knowledge of the system.

5.3 Navier Stokes Equation

We study a stochastic version of the 2D Navier Stokes equation for a viscous, incompressible fluid in vorticity form on the unit torus

$$\partial_t w(x,t) + u(x,t) \cdot \nabla w(x,t) = \nu \Delta w(x,t) + f(x) + \alpha\varepsilon,$$
$$\nabla \cdot u(x,t) = 0, \quad w(x,0) = w_0(x),$$

where $x \in (0,1)^2$, $t \in (0,T]$, u is the velocity field, $w = \nabla \times u$ is the vorticity, $\nu \in \mathbb{R}_+$ is the viscosity coefficient, which is set to 10^{-4}, f is the deterministic forcing function, which takes the form $f(x) = 0.1[\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2))]$, and ε is the stochastic forcing given by $\varepsilon = \dot{W}$ for W being a Q-Wiener process which is coloured in space. The

Table 3: Relative L^2 error norm for the posterior mean of the diffusion coefficient and the solution for the sensor placement experiment on the Darcy flow equation. We take the average over 50 sensor placement loops.

Method	Design points	$\|\log\widehat{\kappa} - \log\kappa_{\text{tr}}\|^2/\|\log\kappa_{\text{tr}}\|^2$	$\|\widehat{u} - u_{\text{tr}}\|^2/\|u_{\text{tr}}\|^2$
Functional Neural Coupling (Ours)	Adaptive BED	$\mathbf{0.234 \pm 0.078}$	$\mathbf{0.102 \pm 0.083}$
	Batch non-adaptive BED	0.337 ± 0.091	0.192 ± 0.087
	Quasi-Monte Carlo sequence	0.711	0.328
FNO surrogate (Li et al., 2021)	Adaptive BED	0.306 ± 0.130	0.117 ± 0.106
	Batch non-adaptive BED	0.551 ± 0.172	0.220 ± 0.118
	Quasi-Monte Carlo sequence	1.182	0.379
FNO w/ oracle noise surrogate (Salvi et al., 2022)	Adaptive BED	$\underline{0.155 \pm 0.101}$	$\underline{0.093 \pm 0.089}$
	Batch non-adaptive BED	0.291 ± 0.089	0.124 ± 0.110
	Quasi-Monte Carlo sequence	0.459	0.255

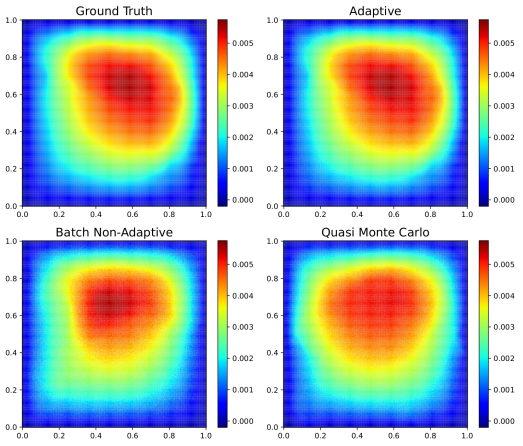


Figure 3: Top left: Observed solution. Posterior mean solutions from observations at adaptively chosen locations (top right), batch non-adaptively selected locations (bottom left) and Quasi Monte Carlo points (bottom right).

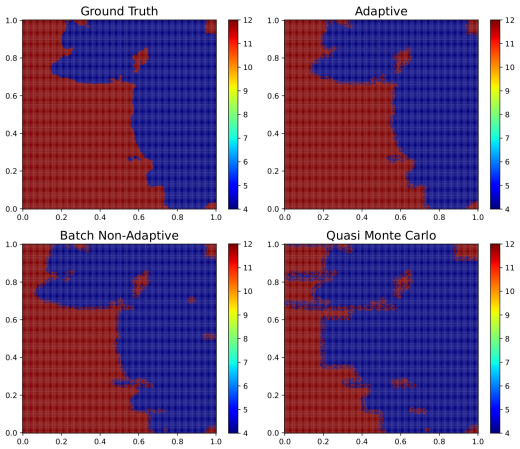


Figure 4: Top left: True diffusion coefficient. Posterior mean coefficients from observations at adaptively chosen locations (top right), batch non-adaptively selected locations (bottom left) and Quasi Monte Carlo points (bottom right).

initial condition $w_0(x)$ is generated according to a Gaussian random field with periodic boundary conditions. Following Salvi et al. (2022), we solve the previous equation for each realisation of the Q-Wiener process using a pseudo-spectral solver and a time step of size 10^{-3}. The training data is generated by evaluating the solution at time points $t = 1, 2, 3$ on a 16×16 spatial mesh. We learn a Functional Neural Coupling between the initial vorticity ω_0 and the vorticity at times $t = 1, 2, 3$.

The sensor placement task consists in finding optimal locations for fifteen additional measurement sites based on five initial observations of a randomly chosen solution. Inference is performed using observations of the vorticity at $t = 1, 2, 3$ in the chosen locations. As illustrated in Figure 5, inference results using adaptive sensor placement locations closely match the ground truth. Table 4 shows that our Functional Neural Coupling approach outperforms the FNO surrogate and performs close to the FNO with oracle noise which is unattainable in practice.

6 DISCUSSION AND LIMITATIONS

In this paper we introduce Functional Neural Couplings, a novel approach to learning resolution-invariant surrogate models for stochastic functional operators by modelling the joint distribution of operator input and output with an energy-based model without assuming any knowledge about the intrinsic driving noise. The combination of probabilistic generative models and the implicit neural representation unlocks a unique set of properties of our method: it is resolution independent (meaning that even if trained on a lower resolution it can be directly evaluated on a higher resolution), it captures the effects of stochasticity present in the for-

Table 4: Relative L^2 errors for the posterior mean of the initial vorticity w_0 and the vorticity at $t = 1, 2, 3$, $\underline{w}_t$, for the sensor placement experiment on the Navier Stokes equation. We take the average over 20 sensor placement experiments.

Method	Design points	$\|\widehat{w}_0 - w_{\mathrm{tr},0}\|^2/\|w_{\mathrm{tr},0}\|^2$	$\|\widehat{\underline{w}}_t - \underline{w}_{\mathrm{tr},t}\|^2/\|\underline{w}_{\mathrm{tr},t}\|^2$
Functional Neural Coupling (Ours)	Adaptive BED	$\mathbf{0.293 \pm 0.077}$	$\mathbf{0.175 \pm 0.091}$
	Batch non-adaptive BED	0.321 ± 0.083	0.239 ± 0.090
	Quasi-Monte Carlo sequence	0.578	0.422
FNO surrogate (Li et al., 2021)	Adaptive BED	0.382 ± 0.067	0.242 ± 0.095
	Batch non-adaptive BED	0.454 ± 0.092	0.288 ± 0.089
	Quasi-Monte Carlo sequence	0.652	0.576
FNO w/ oracle noise surrogate (Salvi et al., 2022)	Adaptive BED	$\underline{0.221 \pm 0.065}$	$\underline{0.103 \pm 0.079}$
	Batch non-adaptive BED	0.301 ± 0.080	0.169 ± 0.083
	Quasi-Monte Carlo sequence	0.454	0.332

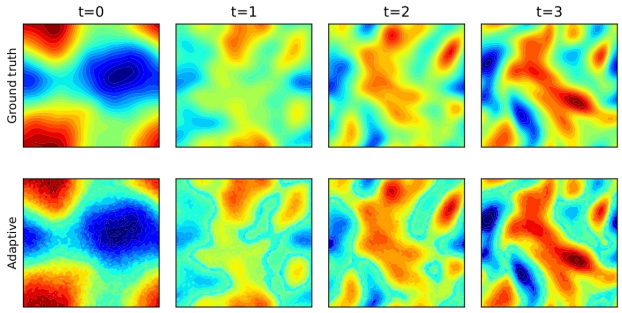


Figure 5: Top: ground truth solution. Bottom: posterior mean based on observations at adaptively chosen sensor locations using our Functional Neural Coupling surrogate.

ward model (as is the case for SPDEs), and it outputs an unnormalised joint distribution which allows inference and downstream tasks such as sensor placement through sampling from the posterior distribution. To the best of our knowledge, our work is the first that makes use of generative models in sensor placement of inverse problems for SPDEs avoiding costly traditional MCMC-based methods and providing a tractable likelihood model. The numerical experiments demonstrate that our approach outperforms the FNO surrogate in the sensor placement task, as the latter are unsuitable for stochastic forward problems. While our approach does not require full knowledge of the driving noise, it is still able to achieve performance comparable to the FNO having full access to the driving noise, thus yielding an accurate and practically useful surrogate. We also show improved accuracy when performing inference with a principled sensor placement strategy using BED tools, compared to using points from a Quasi-Monte Carlo sequence. In both cases, our Functional Neural Coupling serves as a surrogate for the joint distribution.

While Functional Neural Couplings are a promising method for probabilistic surrogate modelling of expensive stochastic systems, they make the assumption that the density of parameter-solution pairs is positive everywhere. This may pose challenges to our method when the data distribution is sparsely distributed on $\mathcal{A} \times \mathcal{U}$, for example, in fully deterministic settings or when only few parameter choices lead to stable behaviour of the system. In particular, the training of the energy-based model may fail when the implicit neural representation is chosen too high-dimensional, thus we may need to sacrifice the accuracy with which the functions are described through the finite dimensional latent variables to allow for stable training. In addition, the energy based model will not necessarily generalise to parts of the state space not covered by the prior from which $a \in \mathcal{A}$ was sampled. Thus, the specific training data needs to be chosen judiciously. Alternatively, an adaptive data-generation approach as proposed in Papamakarios et al. (2019) might be required, particularly when the simulator is expensive.

For future work, we are interested in exploring improved sample efficient methods for the modelling and training of energy-based models for functional data and studying sequential strategies that use the observation data more effectively for fine-tuning the base energy-based model.

Acknowledgements

PCE is supported by EPSRC through the Modern Statistics and Statistical Machine Learning (StatML) CDT programme, grant no. EP/S023151/1. TS is supported by the EPSRC-DTP scholarship partially funded by the Department of Mathematics, Imperial College London. We thank the anonymous reviewers for their comments.

References

Alen Alexanderian. Optimal experimental design for infinite-dimensional Bayesian inverse problems governed by PDEs: A review. *Inverse Problems*, 37(4): 043001, 2021.

Tom R. Andersson, Wessel P. Bruinsma, Stratis Markou, James Requeima, Alejandro Coca-Castro, Anna Vaughan, Anna-Louise Ellis, Matthew A. Lazzara, Dani Jones, Scott Hosking, and et al. Environmental sensor placement with convolutional gaussian neural processes. *Environmental Data Science*, 2:e32, 2023.

Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697 – 725, 2009.

Lorenzo Baldassari, Ali Siahkoohi, Josselin Garnier, Knut Solna, and Maarten V. de Hoop. Conditional score-based diffusion models for Bayesian inference in infinite dimensions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and Garth N. Wells. DOLFINx: the next generation FEniCS problem solving environment. preprint, 2023.

Robert James Barthorpe and Keith Worden. Emerging trends in optimal structural health monitoring system design: From sensor placement to system evaluation. *Journal of Sensor and Actuator Networks*, 9 (3), 2020. ISSN 2224-2708.

Yoshua Bengio and Olivier Delalleau. Justifying and Generalizing Contrastive Divergence. *Neural Computation*, 21(6):1601–1621, 06 2009. ISSN 0899-7667.

Miguel Á. Carreira-Perpiñán and Geoffrey Hinton. On contrastive divergence learning. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pages 33–40. PMLR, 06–08 Jan 2005.

Kathryn Chaloner and Isabella Verdinelli. Bayesian Experimental Design: A Review. *Statistical Science*, 10(3):273 – 304, 1995.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48): 30055–30062, 2020.

Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Golinski, Yee Whye Teh, and Arnaud Doucet. COIN++: Neural Compression Across Modalities. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Adam Foster, Martin Jankowiak, Eli Bingham, Paul Horsfall, Yee Whye Teh, Tom Rainforth, and Noah D. Goodman. Variational Bayesian Optimal Experimental Design. In *Neural Information Processing Systems*, 2019.

Adam Foster, Martin Jankowiak, Matthew O'Meara, Yee Whye Teh, and Tom Rainforth. A Unified Stochastic Gradient Approach to Designing Bayesian-Optimal Experiments. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2959–2969. PMLR, 26–28 Aug 2020.

Adam Foster, Desi R. Ivanova, Ilyas Malik, and Tom Rainforth. Deep Adaptive Design: Amortizing Sequential Bayesian Experimental Design. In *International Conference on Machine Learning*, 2021.

Giulio Franzese, Giulio Corallo, Simone Rossi, Markus Heinonen, Maurizio Filippone, and Pietro Michiardi. Continuous-Time Functional Diffusion Processes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International conference on machine learning*, pages 1704–1713. PMLR, 2018.

Pierre Glaser, Michael Arbel, Arnaud Doucet, and Arthur Gretton. Maximum Likelihood Learning of Energy-Based Models for Simulation-Based Inference. *arXiv preprint arXiv:2210.14756*, 2022.

Robert B Gramacy. *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. CRC press, 2020.

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18:1527–1554, 2006.

Aapo Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

Noble Kennamer, Steven Walton, and Alexander Ihler. Design amortization for Bayesian optimal experimental design. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*

and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023.

Gavin Kerrigan, Justin Ley, and Padhraic Smyth. Diffusion Generative Models in Infinite Dimensions. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 9538–9563. PMLR, 2023.

Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural Operator: Learning Maps Between Function Spaces. *CoRR*, abs/2108.08481, 2021.

Yann Lecun, Sumit Chopra, Raia Hadsell, Marc Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Scholkopt, A. Smola, and B. Taskar, editors, *Predicting structured data*. MIT Press, 2006.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, 2021.

Jae Hyun Lim, Nikola B. Kovachki, Ricardo Baptista, Christopher Beckham, Kamyar Azizzadenesheli, Jean Kossaifi, Vikram Voleti, Jiaming Song, Karsten Kreis, Jan Kautz, Christopher Pal, Arash Vahdat, and Anima Anandkumar. Score-based Diffusion Models in Function Space. *arXiv preprint arXiv:2302.07400*, 2023a.

Jen Ning Lim, Sebastian Vollmer, Lorenz Wolf, and Andrew Duncan. Energy-based models for functional data using path measure tilting. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206, pages 1904–1923. PMLR, 2023b.

Da Long and Shandian Zhe. Invertible Fourier Neural Operators for Tackling Both Forward and Inverse Problems. *arXiv preprint arXiv:2402.11722*, 2024.

Swapnil Mishra, Seth Flaxman, Tresnia Berah, Harrison Zhu, Mikko Pakkanen, and Samir Bhatt. π VAE: a stochastic process prior for Bayesian deep learning with MCMC. *Statistics and Computing*, 32(6):96, 2022.

Roberto Molinaro, Yunan Yang, Björn Engquist, and Siddhartha Mishra. Neural inverse operators for solving PDE inverse problems. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. Society for Industrial and Applied Mathematics, USA, 1992.

George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd international conference on artificial intelligence and statistics*, pages 837–848. PMLR, 2019.

Jakiw Pidstrigach, Youssef Marzouk, Sebastian Reich, and Sven Wang. Infinite-Dimensional Diffusion Models. *arXiv preprint arXiv:2302.10130*, 2023.

Md Ashiqur Rahman, Manuel A Florez, Anima Anandkumar, Zachary E Ross, and Kamyar Azizzadenesheli. Generative adversarial neural operators. *arXiv preprint arXiv:2205.03017*, 2022.

Tom Rainforth, Robert Cornish, Hongseok Yang, and Andrew Warrington. On Nesting Monte Carlo Estimators. In *International Conference on Machine Learning*, 2018.

Tom Rainforth, Adam Foster, Desi R Ivanova, and Freddie Bickford Smith. Modern Bayesian experimental design. *Statistical Science*, 39(1):100–114, 2024.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.

Cristopher Salvi, Maud Lemercier, and Andris Gerasimovics. Neural stochastic PDEs: Resolution-invariant learning of continuous spatiotemporal dynamics. *Advances in Neural Information Processing Systems*, 35:1333–1344, 2022.

Tobias Schröder, Zijing Ou, Jen Ning Lim, Yingzhen Li, Sebastian Josef Vollmer, and Andrew Duncan. Energy Discrepancies: A Score-Independent Loss for Energy-Based Models. In *Neural Information Processing Systems*, 2023.

Louis Serrano, Lise Le Boudec, Armand Kassaï Koupaï, Thomas X Wang, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Operator Learning with Neural Fields: Tackling PDEs on General Geometries. In *Neural Information Processing Systems*, 2023.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Neural Information Processing Systems*, 2020.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors,

Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

Yang Song and Stefano Ermon. Improved Techniques for Training Score-Based Generative Models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

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

Ilya Sutskever and Tijmen Tieleman. On the convergence properties of contrastive divergence. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 789–795, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Improved architectures and training algorithms for deep operator networks. *Journal of Scientific Computing*, 92(2):35, 2022.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011.

Mengjiao Yang, Bo Dai, Hanjun Dai, and Dale Schuurmans. Energy-based processes for exchangeable data. In *International Conference on Machine Learning*, pages 10681–10692. PMLR, 2020.

Ka-Veng Yuen, Lambros S. Katafygiotis, Costas Papadimitriou, and Neil C. Mickleborough. Optimal Sensor Placement Methodology for Identification with Unmeasured Excitation . *Journal of Dynamic Systems, Measurement, and Control*, 123(4):677–686, 02 2001. ISSN 0022-0434.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] Further details are provided in the Appendix.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] It is provided in the supplementary material.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] We only include one theoretical result, Proposition 1 (in the Appendix). The proof is provided and the statement of the theorem includes all the necessary assumptions.
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] It is provided in the supplementary material.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] All the details are provided in the Appendix.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] The type of computational resources used to run the experiments is specified in the Appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] In particular, we reuse code (available on GitHub) from previous work, which is appropriately cited where necessary.
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Methodology: Further Details

A.1 Implicit Neural Representations

Since we are using the latent representations to learn a surrogate model for the operator $\mathcal{G}$, similar functions should map to similar latent representations. This is guaranteed via the following proposition.

Proposition 1. *Let $g_\psi(\cdot,\cdot) : \Omega \times \mathbb{R}^{d_z} \to \mathbb{R}$ be an implicit neural representation. If the activation functions of the neural network are continuous and differentiable almost everywhere, then the function g_ψ is continuous and differentiable almost everywhere in both variables. In addition, if the domain $\Omega \subseteq \mathbb{R}^{d_x}$ is bounded and the neural network activation functions are Lipschitz, then g_ψ is Lipschitz with respect to the second variable $\mathbf{z}$, i.e. there exists a constant L such that for all $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^{d_z}$*

$$\|g_\psi(\cdot,\mathbf{z}) - g_\psi(\cdot,\mathbf{z}')\|_{L_2} \leq L\|\mathbf{z}-\mathbf{z}'\|_{\mathbb{R}^{d_z}}.$$

Proof. Recall that g_ψ is composed of layers of the form $\sin(w_0(W\mathbf{h}+\mathbf{b}+\beta))$, where $\beta = \beta(\mathbf{z})$ is a hypernetwork itself with continuous and differentiable almost everywhere activation functions, in particular in our implementation we use ReLU activations which satisfy this condition. Therefore, it immediately follows that g_ψ is continuous and differentiable almost everywhere in both variables.

On the other hand, we have that linear functions are Lipschitz and if the activation functions are also Lipschitz by composition we have that $g_\psi(\mathbf{x},\cdot)$ is Lipschitz for every $\mathbf{x} \in \Omega$, i.e, there exists a constant $\tilde{L}$ such that for every $\mathbf{x} \in \Omega$

$$\|g_\psi(\mathbf{x},\mathbf{z}) - g_\psi(\mathbf{x},\mathbf{z}')\|_{\mathbb{R}} \leq \tilde{L}\|\mathbf{z}-\mathbf{z}'\|_{\mathbb{R}^{d_z}}.$$

Putting this together with the fact that the domain $\Omega \in \mathbb{R}^{d_x}$ is bounded, it follows that g_ψ is Lipschitz with respect to the second variable $\mathbf{z}$. In particular, in our case we use sine and ReLU activation functions which are Lipschitz with constant 1. □

A.2 Energy-Based Model Training

Energy-Based Models are a class of parametric unnormalised probabilistic models of the form $\exp(-E_\theta)$ originally inspired by statistical physics. Despite their flexibility, energy-based models have not seen widespread adoption in machine learning applications due to the challenges involved in training them. In particular, because the normalising constant is intractable, these models cannot be optimised using maximum likelihood estimation. Alternative training methods have been explore such as contrastive divergence (Hinton et al., 2006), score-based methods (Hyvärinen, 2005; Vincent, 2011; Song and Ermon, 2020) and energy discrepancy (Schröder et al., 2023).

Contrastive divergence (Hinton et al., 2006) is a method that approximates the gradient of the log-likelihood via short runs of a Markov Chain Monte Carlo (MCMC) process. Although using short MCMC runs greatly reduces both the computation per gradient step and the variance of the estimated gradient (Carreira-Perpiñán and Hinton, 2005), it comes at the cost of producing poor approximations of the energy function. This issue arises, in part, because contrastive divergence is not the gradient of any objective function (Sutskever and Tieleman, 2010; Bengio and Delalleau, 2009) which significantly limits the theoretical understanding of its convergence.

Score-based methods provide an alternative way for training based on minimising the expected squared distance of the score function of the true distribution $\nabla_x \log p_{\text{data}}$ and the score function given by the model $\nabla_x \log p_\theta$, which by definition are independent of the normalising constant (Hyvärinen, 2005; Vincent, 2011). However, these methods only use gradient information and are therefore short-sighted (Song and Ermon, 2019) as they do not resolve the global characteristics of the distribution when limited data are available.

Energy discrepancy attempts to solve the problems of the two previous methods by proposing a new loss function that compares the data distribution and the energy-based model (Schröder et al., 2023). This loss is given by

$$\mathcal{L}_{t,M,w}(\theta) := \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{w}{M} + \frac{1}{M}\sum_{j=1}^{M} \exp\left(E_\theta(\mathbf{z}_i) - E_\theta(\mathbf{z}_i + \sqrt{t}\boldsymbol{\xi}_i + \sqrt{t}\boldsymbol{\xi}'_{i,j})\right)\right),$$

where $\boldsymbol{\xi}_i, \boldsymbol{\xi}'_{i,j} \sim \mathcal{N}(0, \mathbf{I}_{d_{z_a}+d_{z_u}})$ are i.i.d. random variables and t, M, w are tunable hyper-parameters. Energy discrepancy depends only on the energy function and is independent of the scores and MCMC samples from the energy-based model.

We applied these three methods to train the energy-based model in our framework, obtaining the best results with the latter. Notably, for the 2D examples, both contrastive divergence and score matching failed to converge.

A.3 Optimal Bayesian Experimental Design For Sensor Placement

The sensor placement task consists in optimally selecting $\boldsymbol{\xi}$ to maximise the information gain about the parameter and solution of the stochastic PDE. In our proposed framework, parameter and solution are approximated by $g_\psi(\cdot, \mathbf{z}_\kappa)$ and $f_\phi(\cdot, \mathbf{z}_u)$, respectively, where $\mathbf{z}_\kappa \in \mathbb{R}^{d_{z_\kappa}}$ and $\mathbf{z}_u \in \mathbb{R}^{d_{z_u}}$ are the associated latent embeddings. Mathematically, the utility function for $\boldsymbol{\xi}$ needs to maximise the expected information gain over the prior $p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u)$, as measured by relative entropy. This is equivalent to maximising the expected KL-divergence

$$\begin{aligned} U(\boldsymbol{\xi}) &= \mathbb{E}_{p(\mathbf{y}|\boldsymbol{\xi})}\big[D_{\mathrm{KL}}\big(p(\mathbf{z}_\kappa, \mathbf{z}_u|\boldsymbol{\xi}, \mathbf{y}) \,\|\, p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u)\big)\big] \\ &= \int d(\mathbf{z}_\kappa, \mathbf{z}_u) \int d\mathbf{y}\big(\log p(\mathbf{z}_\kappa, \mathbf{z}_u|\boldsymbol{\xi}, \mathbf{y}) - \log p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u)\big) p(\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{y}|\boldsymbol{\xi}), \end{aligned}$$

where the expectation is computed over the predictive distribution of the new (yet unobserved) data $p(\mathbf{y}|\boldsymbol{\xi})$. Applying Bayes theorem, we rewrite the above expression in a form amenable to estimation

$$\begin{aligned} U(\boldsymbol{\xi}) &= \int d(\mathbf{z}_\kappa, \mathbf{z}_u) \int d\mathbf{y} \left(\log \frac{p(\mathbf{y}|\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{\xi}) p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u)}{p(\mathbf{y}|\boldsymbol{\xi})} - \log p_\theta(\mathbf{z}_\kappa, \mathbf{z}_u)\right) p(\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{y}|\mathbf{d}) \\ &= \int d(\mathbf{z}_\kappa, \mathbf{z}_u) \int d\mathbf{y}\big(\log p(\mathbf{y}|\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{\xi}) - \log p(\mathbf{y}|\boldsymbol{\xi})\big) p(\mathbf{z}_\kappa, \mathbf{z}_u, \mathbf{y}|\boldsymbol{\xi}) \\ &= \mathbb{E}_{p(\mathbf{z}_\kappa, \mathbf{z}_u, \mathbf{y}|\boldsymbol{\xi})} \log p(\mathbf{y}|\mathbf{z}_\kappa, \mathbf{z}_u, \boldsymbol{\xi}) - \mathbb{E}_{p(\mathbf{y}|\boldsymbol{\xi})} \log p(\mathbf{y}|\boldsymbol{\xi}). \end{aligned} \tag{2}$$

Finding the optimal sensor location $\boldsymbol{\xi}$ is very challenging since the density $p(\mathbf{y}|\boldsymbol{\xi})$ in Eq. (2) is intractable. A naive Monte Carlo approach will require the use of a nested Monte Carlo estimator which results in high variance and converges relatively slow. Moreover, the utility $U(\boldsymbol{\xi})$ must be computed separately for each location $\boldsymbol{\xi}$, which makes the framework highly inefficient especially in high-dimensional settings, as $U(\boldsymbol{\xi})$ is fed into an outer optimisation loop to select the optimal sensor position.

To circumvent these inefficiencies amortised variational approaches have been proposed in the literature (Foster et al., 2019, 2020, 2021; Kennamer et al., 2023). By constructing a variational lower bound to $U(\boldsymbol{\xi})$, we can obtain a unified framework that can be simultaneously optimised with respect to both the variational φ and position parameters $\boldsymbol{\xi}$ using stochastic gradient ascent (Robbins and Monro, 1951). Foster et al. (2020) proposed the adaptive contrastive estimation (ACE) bound given by

$$\hat{U}_{\mathrm{ACE}}(\boldsymbol{\xi}, \varphi) = \mathbb{E}\left[\log \frac{p(\mathbf{y}|\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi})}{\frac{1}{L+1}\sum_{l=0}^{L} \frac{p_\theta(\mathbf{z}_{\kappa,l}, \mathbf{z}_{u,l}) p(\mathbf{y}|\mathbf{z}_{\kappa,l}, \mathbf{z}_{u,l}, \boldsymbol{\xi})}{q_\varphi(\mathbf{z}_{\kappa,l}, \mathbf{z}_{u,l}|\mathbf{y})}}\right],$$

where $q_\varphi(\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}|\mathbf{y})$ is the inference network which takes as input φ, $\mathbf{y}$ and outputs a distribution over $(\mathbf{z}_\kappa, \mathbf{z}_u)$ and the expectation is taken with respect to $p_\theta(\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}) p(\mathbf{y}|\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi}) q_\varphi(\mathbf{z}_{\kappa,1:L}, \mathbf{z}_{u,1:L}|\mathbf{y})$. Foster et al. (2020) prove that $\hat{U}_{\mathrm{ACE}}(\boldsymbol{\xi}, \varphi) \leq U(\boldsymbol{\xi})$. By replacing the inference network $q_\varphi(\mathbf{z}_\kappa, \mathbf{z}_u|\mathbf{y})$ with the prior $p_\theta(\mathbf{z}_a, \mathbf{z}_u)$ to generate contrastive samples, we obtain the prior contrastive estimation (PCE) bound

$$\begin{aligned} \hat{U}_{\mathrm{PCE}}(\boldsymbol{\xi}) &= \mathbb{E}\left[\log \frac{p(\mathbf{y}|\mathbf{z}_{a,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi})}{\frac{1}{L+1}\sum_{l=0}^{L} p(\mathbf{y}|\mathbf{z}_{\kappa,l}, \mathbf{z}_{u,l}, \boldsymbol{\xi})}\right] \\ &= -\mathbb{E}\left[\log\left(1 + \sum_{l=1}^{L} \frac{p(\mathbf{y}|\mathbf{z}_{\kappa,l}, \mathbf{z}_{u,l}, \boldsymbol{\xi})}{p(\mathbf{y}|\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi})}\right)\right] + C, \end{aligned}$$

where the expectation is over $p_\theta(\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}) p(\mathbf{y}|\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi}) p_\theta(\mathbf{z}_{\kappa,1:L}, \mathbf{z}_{u,1:L})$. This alternative estimator significantly speeds up the training since no variational parameters need to be learnt.

To optimise the PCE bound with respect to $\boldsymbol{\xi}$ we need an unbiased gradient estimator of $\partial \hat{U}_{\mathrm{PCE}}/\partial \boldsymbol{\xi}$. This can be achieved by using a reparametrisation trick, where we introduce a random variable ε independent of $\boldsymbol{\xi}$

together with a representation of $\mathbf{y}$ as a deterministic function of ε, that is, $\mathbf{y} = \mathbf{y}(\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi}, \varepsilon)$. Recalling that $\mathbf{y}_i \sim \mathcal{N}(f_\phi(\boldsymbol{\xi}, \mathbf{z}_{u_i}), \sigma^2)$ we can write $\mathbf{y} = f_\phi(\boldsymbol{\xi}, \mathbf{z}_{u,0}) + \sigma\varepsilon$, where $\varepsilon \sim \mathcal{N}(0,1)$. This allows us to derive the following reparametrised gradient

$$\frac{\partial \hat{U}_{\mathrm{PCE}}}{\partial \boldsymbol{\xi}} = \mathbb{E}\left[\frac{\partial g}{\partial \boldsymbol{\xi}} + \frac{\partial g}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \boldsymbol{\xi}}\right],$$

where the expectation is over $p_\theta(\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0})p(\varepsilon)p_\theta(\mathbf{z}_{\kappa,1:L}, \mathbf{z}_{u,1:L})$ and

$$g(\mathbf{y}, \mathbf{z}_{\kappa,0:L}, \mathbf{z}_{u,0:L}, \boldsymbol{\xi}) = -\log\left(1 + \sum_{l=1}^{L} \frac{p(\mathbf{y}|\mathbf{z}_{\kappa,l}, \mathbf{z}_{u,l}, \boldsymbol{\xi})}{p(\mathbf{y}|\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi})}\right).$$

A Monte Carlo approximation of this expectation leads to a much lower variance estimator for the true $\boldsymbol{\xi}$-gradient. Note that $\mathbb{E}[\partial g/\partial \boldsymbol{\xi}]$ can be efficiently computed in `PyTorch` as $\partial \mathbb{E}[g]/\partial \boldsymbol{\xi}$ by detaching $\mathbf{y}$ from the computational graph. On the other hand,

$$\frac{\partial g}{\partial \mathbf{y}} = -\frac{1}{\exp(-g)} \sum_{l=1}^{L} \frac{p(\mathbf{y}|\mathbf{z}_{\kappa,l}, \mathbf{z}_{u,l}, \boldsymbol{\xi})}{p(\mathbf{y}|\mathbf{z}_{\kappa,0}, \mathbf{z}_{u,0}, \boldsymbol{\xi})} \frac{1}{\sigma^2} \left(f_\phi(\boldsymbol{\xi}, \mathbf{z}_{u,l}) - f_\phi(\boldsymbol{\xi}, \mathbf{z}_{u,0})\right),$$

which can be evaluated using the numerically stabilised logsumexp function, and

$$\frac{\partial \mathbf{y}}{\partial \boldsymbol{\xi}} = \frac{\partial f_\phi(\boldsymbol{\xi}, \mathbf{z}_{u,0})}{\partial \boldsymbol{\xi}}.$$

B Numerical Experiment: Stochastic Lotka-Volterra Model

Another active field of research, where our framework can bring significant benefits is inference on models with intractable likelihoods, such as dynamical systems described by stochastic PDEs. The intractability of the density in such systems arises from the need to marginalise the transition probabilities over all possible trajectories of the stochastic system. To illustrate this, take the general Itô SDE

$$dX_t = a(X_t)dt + dW_t, \quad X_0 = x_0, \quad 0 \le t \le T,$$

where W_t is a Wiener process and $a : \mathbb{R} \to \mathbb{R}$ is a sufficiently regular non-linear drift coefficient. Given N sparse noisy observations of a trajectory, $\{y_i = X_{t_i} + \eta_i\}_{i=1}^N$, where η_i are i.i.d. $\mathcal{N}(0, \sigma^2)$. Then, the joint distribution over observations and latent realisations is given by

$$p(y_1, \ldots, y_N, X_{t_1}, \ldots, X_{t_N}) = p(X_{t_1}, \ldots, X_{t_N}) \prod_{i=1}^{N} p(y_i|X_{t_i}) = p(X_{t_1}) \prod_{j=1}^{N-1} p(X_{t_{j+1}}|X_{t_j}) \prod_{i=1}^{N} p(y_i|X_{t_i}),$$

where for the second equality we have used that the process $\{W_t\}_{t\ge 0}$ has independent increments. We have that $y_i|X_{t_i} \sim \mathcal{N}(X_{t_i}, \sigma^2)$, however in general the distribution of $X_{t_{i+1}}|X_{t_i}$ cannot be expressed analytically. If the observations occur frequently, so that the time between observations t_i and t_{i+1} is small, then we can approximate the distribution of $X_{t+\Delta t}|X_t$ using a Gaussian approximation,

$$X_{t+\Delta t}|X_t \sim \mathcal{N}(X_t + a(X_t)\Delta t, \Delta t).$$

The validity of this approximation breaks down as the step size Δt increases, necessitating alternative approaches in scenarios where the observations are infrequent. Our Functional Neural Couplings method can be used to obtain the distribution over possible trajectories $\{X_t\}_{0\le t\le T}$ from a dataset consisting of point observations of different trajectories, without needing to evaluate each trajectory of the training set at the same time steps.

To provide a specific example, consider a stochastic version of the Lotka-Volterra model, which describes the joint temporal dynamics of two coexisting populations, predator, $X_{1,t}$, and prey, $X_{2,t}$,

$$\begin{aligned} dX_{1,t} &= (\theta_1 X_{1,t} - \theta_2 X_{1,t} X_{2,t})dt + \sqrt{\theta_1 X_{1,t}}dW_t^{(1)} - \sqrt{\theta_2 X_{1,t} X_{2,t}}dW_t^{(2)}, \\ dX_{2,t} &= -(\theta_3 X_{2,t} - \theta_2 X_{1,t} X_{2,t})dt - \sqrt{\theta_3 X_{2,t}}dW_t^{(3)} + \sqrt{\theta_2 X_{1,t} X_{2,t}}dW_t^{(2)}, \end{aligned} \tag{3}$$

where $0 \leq t \leq 1$, $X_{1,0} = a$, $X_{2,0} = b$, W_t is a Wiener process and $\theta = [\theta_1, \theta_2, \theta_3]$ are the model parameters, which for our experiment we keep fixed at $\theta = [5, 0.035, 6]$. We train our functional neural coupling surrogate to learn a probability distribution over possible trajectories $\{X_{1,t}, X_{2,t}\}_{0 \leq t \leq T}$ from a dataset consisting of point observations of simulated trajectories obtained using an Euler-Maruyama scheme with a sufficiently small time step $\Delta t = 0.01$ to ensure numerical stability.

Given the inference data, $\mathcal{D} = \{(y_{1,t_i}, y_{2,t_i})\}_{i=1}^{M}$, consisting on M noisy observations of a trajectory, i.e. $y_{j,t_i} = X_{j,t_i} + \eta_i$ where $\eta_i \sim \mathcal{N}(0, \sigma^2)$ and $\sigma = 0.2$. The posterior distribution over the latent representation $(\mathbf{z_1}, \mathbf{z_2})$ is

$$\begin{aligned} p(\mathbf{z_1}, \mathbf{z_2}|\mathcal{D}) &\propto \prod_{i=1}^{M} p(y_{1,t_i}, y_{2,t_i}|g_\psi(t_i, \mathbf{z_1}), f_\phi(t_i, \mathbf{z_2}))p_\theta(\mathbf{z_1}, \mathbf{z_2}) \\ &= \prod_{i=1}^{M} \mathcal{N}(y_{1,t_i}|g_\psi(t_i, \mathbf{z_1}), \sigma^2)\mathcal{N}(y_{2,t_i}|f_\phi(t_i, \mathbf{z_2}), \sigma^2)p_\theta(\mathbf{z_1}, \mathbf{z_2}), \end{aligned}$$

where g_ψ, f_ϕ denote the INRs for the prey and predator trajectories, respectively.

We perform Bayesian optimal experimental design to place five sensor/measurement times based on two initial low-informative observations. Table 5 shows the relative L^2 error norm for the posterior means of the prey and predator trajectories when averaged across 100 different prey-predator trajectories. Our approach performs comparably to the FNO surrogate with oracle noise, where complete knowledge of the driving noise is assumed, and consistently outperforms the standard FNO surrogate across all types of design points.

Table 5: Relative L^2 errors for the posterior means of the prey and predator trajectories for the sensor placement task. We average over 100 different prey-predator trajectories.

Method	**Design points**	$\|\widehat{X}_1 - X_{\text{tr},1}\|^2/\|X_{\text{tr},1}\|^2$	$\|\widehat{X}_2 - X_{\text{tr},2}\|^2/\|X_{\text{tr},2}\|^2$
Neural Coupling (Ours)	Adaptive BED	$\mathbf{0.070 \pm 0.066}$	$\mathbf{0.068 \pm 0.059}$
	Batch non-adaptive BED	0.081 ± 0.100	0.078 ± 0.068
	Quasi-Monte Carlo sequence	0.101 ± 0.090	0.095 ± 0.073
FNO surrogate (Li et al., 2021)	Adaptive BED	0.092 ± 0.104	0.091 ± 0.199
	Batch non-adaptive BED	0.111 ± 0.163	0.123 ± 0.187
	Quasi-Monte Carlo sequence	0.176 ± 0.211	0.184 ± 0.337
FNO w/ oracle noise surrogate (Salvi et al., 2022)	Adaptive BED	$\underline{0.056 \pm 0.050}$	$\underline{0.045 \pm 0.041}$
	Batch non-adaptive BED	0.061 ± 0.052	0.057 ± 0.055
	Quasi-Monte Carlo sequence	0.089 ± 0.049	0.082 ± 0.060

C Experimental details: Datasets

C.1 Boundary Value Problem

The dataset to train our neural coupling surrogate is of the form

$$\left\{ \left((a_i, b_i), u_i(x_j^i) \right)_{j=1}^{N_i} \right\}_{i=1}^{M},$$

where $a, b \sim \text{Unif}[-3, 3]$, $x_j^i \in \Omega = [-1, 1]$ and u represents the solution corresponding to a realisation of the random variables X_a, X_b. The solutions u_i are computed using DOLFINx (Baratta et al., 2023). To train the FNO with oracle noise baseline (Salvi et al., 2022), we also store the particular realisation of X_a and X_b for each data point.

The dataset consists of 10000 pairs of boundary conditions and solutions. Each solution is evaluated at $N_i = 30$ points in the domain $[-1, 1]$.

C.2 Steady-State Diffusion

The 2D Darcy flow equation is a second-order linear elliptic equation of the form

$$-\nabla \cdot \big(\kappa(\mathbf{x})\nabla u(\mathbf{x})\big) = f(\mathbf{x}) + \alpha\omega$$

with domain $\mathbf{x} \in \Omega = [0,1]^2$ and Dirichlet boundary conditions $u|_{\partial\Omega} = 0$. The force term is kept fixed $f(x) = 0.5$, ω is space white noise and the diffusion coefficient is generated according to $\kappa \sim \mu$, where $\mu = \psi_{\#}\mathcal{N}(0, (-\Delta + 9I)^{-2})$ with zero Neumann boundary conditions on the Laplacian. The mapping $\psi : \mathbb{R} \to \mathbb{R}$ takes the value 12 on the positive part of the real line and 3 on the negative and the push-forward is defined point-wise. The solution is computed using the finite element method.

The dataset consists of 10000 pairs of diffusion coefficients and associated solutions, evaluated on a uniform 16×16 mesh. The particular realisation of ω is also stored for benchmark comparisons.

C.3 Navier Stokes Equation

We consider a stochastic version of the 2D Navier Stokes equation for a viscous, incompressible fluid in vorticity form on the unit torus

$$\partial_t w(x,t) + u(x,t) \cdot \nabla w(x,t) = \nu \Delta w(x,t) + f(x) + \alpha\varepsilon,$$
$$\nabla \cdot u(x,t) = 0, \quad w(x,0) = w_0(x),$$

where $x \in (0,1)^2$, $t \in (0,T]$, u is the velocity field, $w = \nabla \times u$ is the vorticity, $\nu \in \mathbb{R}_+$ is the viscosity coefficient, which is set to 10^{-4}, f is the deterministic forcing function, which takes the form $f(x) = 0.1[\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2))]$, and ε is the stochastic forcing given by $\varepsilon = \dot{W}$ for W being a Q-Wiener process which is coloured in space. The initial condition is generated according to $w_0 \sim \mathcal{N}(0, 3^{3/2}(-\Delta + 49I)^{-3}))$ with periodic boundary conditions. Following Salvi et al. (2022), we solve the previous equation for each realisation of the Q-Wiener process using a pseudo-spectral solver, where time is advanced with a Crank-Nicolson update using a time step of size 10^{-3}. The SPDE is solved on a 64×64 mesh in space.

To form the training dataset, we subsample the trajectories w_t by a factor of 4 in space (resulting in a 16×16 spatial resolution) and only consider the initial vorticity w_0 and the vorticity at times $t = 1, 2, 3$. The dataset consists of 20000 trajectories.

C.4 Stochastic Lotka-Volterra Model

To obtain the training dataset, we simulate trajectories for the prey and predator. Specifically, we generate 1000 uniformly distributed initial conditions and simulate 100 stochastic trajectories of the model for each initial condition using an Euler-Maruyama scheme for Eq. (3) with a sufficiently small time step $\Delta t = 0.01$ to ensure numerical stability. Each trajectory is evaluated in 30 different time steps.

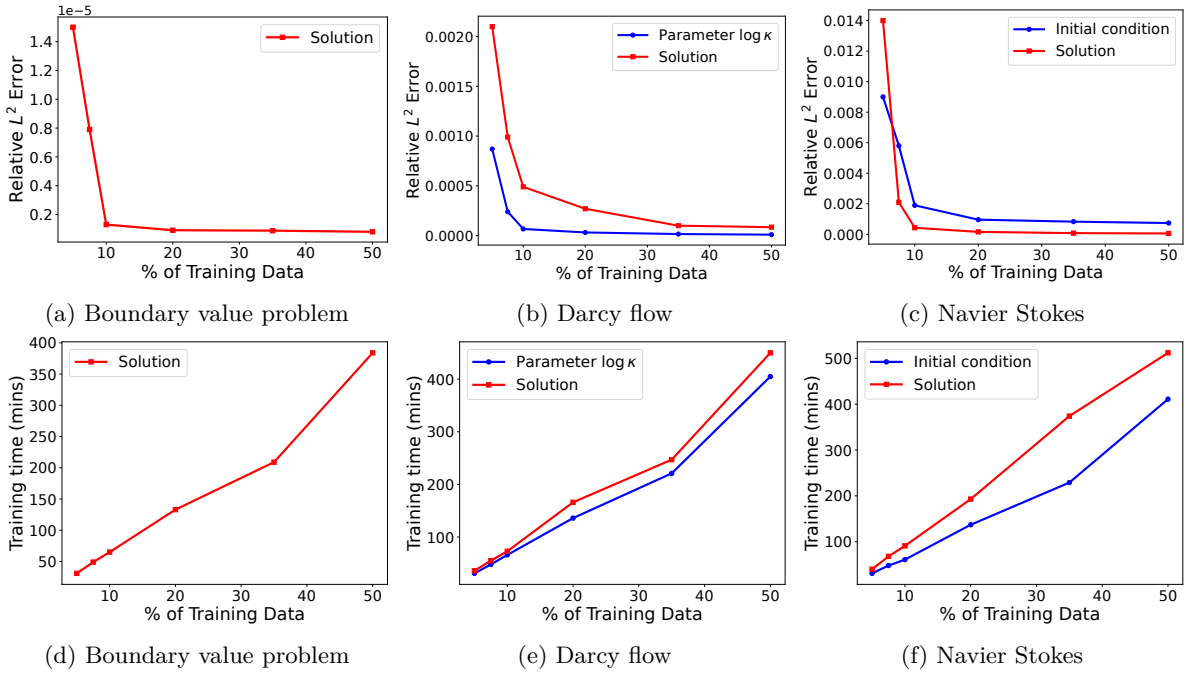
D Experimental details: Implementation

All experiments were conducted on a GPU server consisting of eight Nvidia GTX 3090 Ti GPU cards, 896 GB of memory and 14TB of local on-server data storage. Each GPU has 10496 cores as well as 24 GB of memory.

D.1 Training

The training is done in two steps. First, we train the modulated INRs to encode the functional data in a finite dimensional latent space. Due to the generalisation capabilities of the INR, we train it on datasets of increasing size until the reconstruction error stabilises on a validation dataset. Figure 6 shows the evolution of the relative L^2 error for the INR reconstructed function, computed on a validation dataset, along with the training times for different percentages of the total dataset size (provided in section C). It is important to remark that this is a practical choice to optimise one of the most computationally expensive steps — learning the INR — without sacrificing performance. In all experiments, we use 300 epochs, a batch size of 16 and for the other hyper-parameters we keep the same values as Dupont et al. (2022). In subsequent experiments, we use 10% of the dataset for training the INR, as this provides a balance between performance and computational cost.

Once, the INRs have been fitted, we obtain the latent representations of the functions of interest in the whole training dataset. As the functional data is encoded using only a few steps of gradient descent (specifically 3 steps, for details see Dupont et al. (2022)), the resulting standard deviation of the latent representations is very small, falling within the range of $[10^{-3}, 10^{-1}]$. Therefore, these raw latent modulations are not appropriate for subsequent processing. To address this, we standardise the codes by subtracting the mean and dividing by the standard deviation. Subsequently, the standardised latent modulations are concatenated in a single vector in cases where we seek for a prior over different functional parameters and solutions. The concatenated latent embeddings are then used to train the joint energy-based model. The final hyper-parameters for the joint energy-based model training are presented in Table 6. In addition, in all experiments we use Adam optimiser with a learning rate of 10^{-3} and an exponential scheduler. Note that the means and standard deviations are stored to renormalise the samples generated by the joint energy-based model, which are then used as modulations for the INR to recover the functions in real space.



(a) Boundary value problem (b) Darcy flow (c) Navier Stokes

(d) Boundary value problem (e) Darcy flow (f) Navier Stokes

Figure 6: Evaluation of INR performance across different training data percentages. Top: mean relative L^2 error norm for the INR reconstructed function, computed on a validation dataset of size 200. Bottom: training times of the INRs.

Table 6: Final hyper-parameters for joint energy-based model in the different experiments.

Dataset	t	M	w	Epochs
Boundary Value Problem	1	4	1	1000
Steady-State Diffusion	0.5	16	1	1000
Navier Stokes	1	32	1	1000
Lotka-Volterra Model	1	4	1	1500

D.2 Architectures

D.2.1 Implicit Neural Representation

We have only made minor changes to the COIN++ model proposed by Dupont et al. (2022), so that it can take arbitrary point evaluations of the functions of interest. The number of layers and the dimension of each layer remain the same. The initialisation scheme is the same as the one proposed in Sitzmann et al. (2020).

D.2.2 Energy-Based Model

In all our experiments, each training point for the joint energy-based model consists of the concatenation of the latent representations of the stochastic PDE solution and the functional or vector-valued parameter of the stochastic PDE, and our goal is to understand the connection between the two by learning their joint probability density. To do this, we build on Wang et al. (2022) to design the neural network architecture for the energy function. First, we uplift each part of the input vector into a latent space (using an encoder) so that they have the same dimension (equal to 128) and then propagate and merge them, with shared connections between the two branches. The architecture of the network is illustrated in Figure 7.

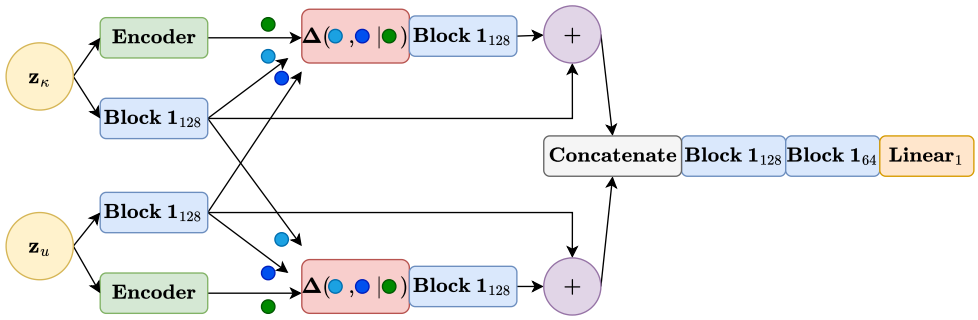


Figure 7: Energy-based model neural network architecture, where $\mathbf{z}_\kappa$ and $\mathbf{z}_u$ represent the finite dimensional latent embeddings of two functions, such as, the parameter and solution of the PDE.

The specific structure of each element of the architecture is the following

- Encoder block
$$\text{Encoder}(\mathbf{x}) = \text{Linear}(\sigma \circ \text{Linear}(\boldsymbol{y}) + \boldsymbol{y}), \quad \boldsymbol{y} = \sigma \circ \text{Linear}(\mathbf{x}),$$
where σ is a GELU (Hendrycks and Gimpel, 2016) activation function.

- Block 1
$$\text{Block } 1_k(\mathbf{x}) = \sigma \circ \text{Linear}(\mathbf{x}),$$
where σ is a RELU activation and k denotes the output dimension.

- Δ operation
$$\Delta(\mathbf{x}, \mathbf{y}|\mathbf{z}) = (1 - \mathbf{z}) \cdot \mathbf{x} + \mathbf{z} \cdot \mathbf{y},$$
where $\cdot$ denotes point-wise multiplication. Note that the vectors $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ need to have the same dimension and the operation is just an interpolation.

- $+$ operation is a point-wise addition.

The dimension remains constant at 128 in the 2 branches of the architecture. Therefore, when both vectors are concatenated, it becomes 256. The last two Block 1's reduce the dimension from 256 to 128 and from 128 to 64, respectively.

D.3 Optimal Bayesian Experimental Design for Sensor

To estimate the utility function as presented in Appendix A.3 we use the following number of samples $N, L = 50$ for all experiments. To obtain them, we sample from the prior, or updated posterior in subsequent iterations, using stochastic gradient Langevin dynamics for a total of 1000 steps, and take the last ones to calculate the utility.

D.4 Choice of Hyperparameters

Our method requires selecting the dimensions of the latent representations of the functional parameters and the solutions of the stochastic PDEs for the different numerical experiments. The latent dimension is selected based on the criterion that the mean relative L^2 error norm between the true function and the function reconstructed from the latent embedding using the INR remains below a certain threshold across all test data points. At the same time, we want the dimension to be low to ensure that the resulting distribution of the latent representations has a positive probability density that can be easily modelled by the energy-based model.

Tables 7, 8, 9 and 10 show the mean relative L^2 error norms for different latent dimensions computed on a validation set of size 200 not seen during training for the solutions of the boundary value problem, the Darcy flow, the Navier Stokes equation, and the Lotka-Volterra model, respectively.

The final latent dimensions for the different numerical experiments are the following.

Boundary Value Problem Recall that in this case the parameter is given by a two-dimensional real-valued vector. The latent dimension for the solution is 11.

Steady-State Diffusion The latent dimensions for the diffusion coefficient and the solution are 32 and 16, respectively.

Navier Stokes Equation The latent dimensions for the initial vorticity and the vorticity at three different time snapshots $t = 1, 2, 3$ are both 32.

Lotka Volterra Model The latent dimension for both prey and predator trajectories is 11.

Table 7: Mean relative L^2 error norm for the INR reconstructed solution of the boundary value problem on a validation set of size 200.

Dimension	$\|\widehat{u} - u_{\text{tr}}\|_{L^2}^2 / \|u_{\text{tr}}\|_{L^2}^2$
15	6.4×10^{-7}
13	9.2×10^{-7}
11	1.3×10^{-6}
9	9.7×10^{-6}
7	8.6×10^{-5}

Table 8: Mean relative L^2 error norm for the INR reconstructed diffusion coefficient and solution of the Darcy flow equation on a validation set of size 200.

Dimension	$\|\log \widehat{\kappa} - \log \kappa_{\text{tr}}\|_{L^2}^2 / \|\log \kappa_{\text{tr}}\|_{L^2}^2$	$\|\widehat{u} - u_{\text{tr}}\|_{L^2}^2 / \|u_{\text{tr}}\|_{L^2}^2$
64	5.4×10^{-6}	7.1×10^{-8}
48	8.3×10^{-5}	6.5×10^{-6}
32	6.7×10^{-5}	5.8×10^{-5}
16	2.6×10^{-4}	4.9×10^{-4}
8	1.9×10^{-2}	3.2×10^{-3}

D.5 Times

We present the data generation, training and inference times to assess the efficiency of our framework compared to classical approaches that use PDE solves within the MCMC. For data generation, we first obtain 10% of the training dataset, which is the amount needed to train the INRs, while the remaining data can be generated in parallel with the training of the INRs for the functional parameter and the stochastic PDE solution. Additionally, the training of both INRs can also be performed in parallel.

Table 9: Mean relative L^2 error norm for the INR reconstructed initial vorticity and the vorticity at $t = 1, 2, 3$, $\underline{w}_t$ of the Navier Stokes equation on a validation set of size 200.

Dimension	$\|\widehat{w}_0 - w_{\text{tr},0}\|^2_{L^2}/\|w_{\text{tr},0}\|^2_{L^2}$	$\|\widehat{\underline{w}}_t - \underline{w}_{\text{tr},t}\|^2_{L^2}/\|\underline{w}_{\text{tr},t}\|^2_{L^2}$
64	3.5×10^{-4}	5.7×10^{-5}
48	8.1×10^{-4}	1.1×10^{-4}
32	1.9×10^{-3}	4.3×10^{-4}
16	9.5×10^{-3}	9.2×10^{-4}
8	2.2×10^{-2}	6.8×10^{-3}

Table 10: Mean relative L^2 error norm for the INR reconstructed prey and predator trajectories on a validation set of size 200.

Dimension	$\|\widehat{X}_1 - X_{\text{tr},1}\|^2_{L^2}/\|X_{\text{tr},1}\|^2_{L^2}$	$\|\widehat{X}_2 - X_{\text{tr},2}\|^2_{L^2}/\|X_{\text{tr},2}\|^2_{L^2}$
15	8.5×10^{-7}	7.8×10^{-7}
13	1.1×10^{-6}	9.9×10^{-7}
11	3.3×10^{-6}	3.0×10^{-6}
9	9.9×10^{-6}	8.4×10^{-6}
7	1.0×10^{-4}	7.1×10^{-5}

After training the INRs we need to compute the latent codes $\mathbf{z}$ on the whole dataset, however this time is around a minute and therefore insignificant compared to the other times reported in Table 11. The inference times correspond to the average time required to obtain an approximation of the posterior distribution (conditioned on the final number of observations used in the sensor placement task) by sampling 1000 iterations per chain for two independent chains using stochastic gradient Langevin dynamics. We note that the inference times are negligible compared to the training and data generation times, which are both performed off-line and only have to be done once. This makes our approach highly efficient for downstream tasks like sensor placement. For instance, in the case of the stochastic Navier Stokes equation, the adaptive Bayesian sensor placement loop requires about 20 minutes to identify fifteen new locations using our surrogate model. In contrast, conventional methods would take several days to achieve the same result, which exceeds the combined training and inference time of our approach.

Table 11: Data generation, training and inference times for the different numerical examples.

Dataset	Data generation		Training		Inference
	10%	Rest	INR	EBM	
Boundary Value Problem	1 mins	9 mins	65 mins	34 mins	1.1 mins
Steady-State Diffusion	8 mins	70 mins	73 mins	40 mins	2.5 mins
Navier Stokes	30 mins	4 hours	91 mins	47 mins	3.2 mins
Lotka Volterra Model	3 mins	27 mins	67 mins	38 mins	1.4 mins

D.6 Benchmarks

In this section, we explain in more detail the FNO (Li et al., 2021) and FNO with oracle noise (Salvi et al., 2022) baselines. For all the numerical examples except for the Navier Stokes equation, the FNO with oracle noise reduces to an FNO with noise as an additional input, as the dynamics in these cases are temporally independent. The architecture and hyperparameters used for training the benchmarks are described below.

Boundary Value Problem The FNO and the FNO with oracle noise are trained on 20000 epochs with a learning rate of 3×10^{-4}. The hyperparameters used are as follows: 16 modes, 64 hidden channels and 4 Fourier

layers. It is worth noting that the FNO learns a mapping between function spaces, whereas our initial condition is represented by a two-dimensional real-valued vector. To address this, we learn a mapping between a linear interpolation of the boundary conditions and the solution.

Steady State Diffusion The FNO and the FNO with oracle noise are trained on 20000 epochs with a learning rate of 1×10^{-4}. The hyperparameters used are as follows: 12 modes per dimension, 32 hidden channels and 4 Fourier layers.

Navier Stokes Equation The FNO is trained on 20000 epochs with a learning rate of 1×10^{-4}. The hyperparameters used are as follows: 12 modes, 32 hidden channels and 4 Fourier layers. For the FNO with oracle noise we use the same hyperparameters and training details specified in Salvi et al. (2022).

Lotka-Volterra Model In this example the neural operator baselines learn a mapping between the prey and predator trajectories. Both baselines are trained on 15000 epochs with a learning rate of 5×10^{-3}. The hyperparameters used are as follows: 16 modes, 64 hidden channels and 4 Fourier layers.

The neural operator benchmarks learn a deterministic mapping between inference parameters and the corresponding solution, in contrast to our neural coupling model, which learns a joint probability distribution over both parameters and solutions.

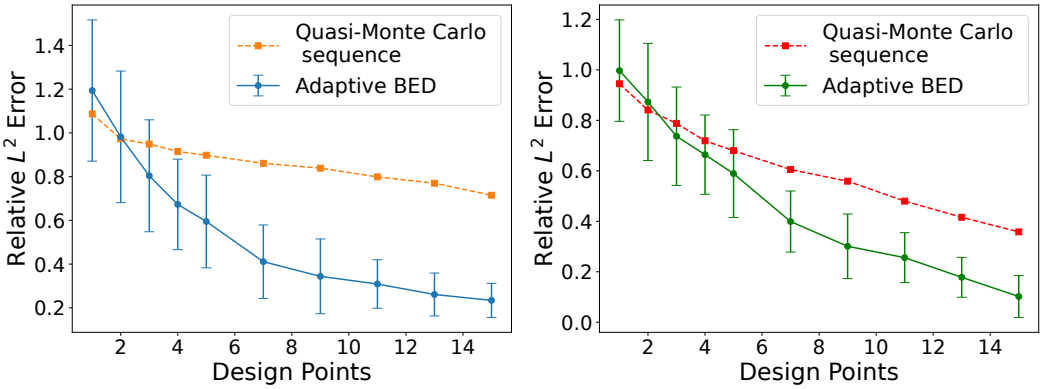
Once these baselines are trained on simulation data, they can be used for the sensor placement task. To achieve this, we need to be able to sample from the posterior distribution of the initial parameters, conditioned on the observations of the solution. In particular, the posterior distribution takes the form

$$p(\kappa|\mathcal{D}) \propto \prod_{i=1}^{M} p(y_i|\kappa, \mathbf{x}_i)p(\kappa) = \prod_{i=1}^{M} \mathcal{N}(y_i|\mathbf{NO}(\kappa)(\mathbf{x}_i), \sigma^2)p(\kappa),$$

where **NO** denotes the neural operator surrogate which takes κ as input and $p(\kappa)$ denotes the prior distribution over κ.

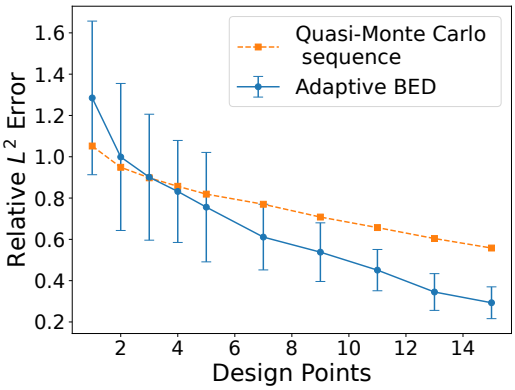
E Additional Results

In the numerical experiments presented in the main text, we assume a fixed computational budget, meaning that the number of sensor locations for observations used in inference is fixed in advance. However, it is interesting to analyse how the posterior contracts to the true value as new observations are included. To investigate this, we track the evolution of the relative L^2 error for the posterior mean for an increasing number of observations, comparing those taken from a quasi-random sequence with those obtained through adaptive optimisation of the locations. Results for the Darcy flow and Navier Stokes equation are provided in Figure 8. We observe that the errors decrease faster for optimally selected points, chosen in an adaptive way, versus randomly chosen locations.

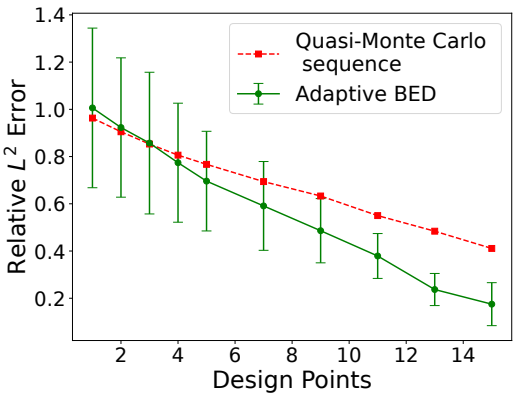


(a) Darcy flow. Diffusion coefficient $\log \kappa$

(b) Darcy flow. Solution u



(c) Navier Stokes. Initial vorticity w_0



(d) Navier Stokes. Vorticity at $t = 1, 2, 3$, $\underline{w}_t$

Figure 8: Relative L^2 error norm for the posterior mean of the initial condition or parameter and the solution for the sensor placement experiment on the Darcy flow and Navier Stokes equations using an increasing number of observations. We take the average over 50 sensor placement loops.