Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

doi:10.3934/xx.xxxxxxx

pp. **X–XX**

DIGITAL TWINS IN BAYESIAN INFERENCE: FUNCTIONAL PRIORS FOR PARTIAL DIFFERENTIAL EQUATIONS

Peter Yatsyshin*

The Alan Turing Institute
London, NW1 2DB, UK

Andrew B. Duncan

Department of Mathematics
Imperial College London
London, SW7 2AZ, UK

(Communicated by the associate editor name)

Abstract. We propose an approach to solving Bayesian inverse problems when the number of evaluations of the forward map is restricted due to computational cost. The proposed introduces a generative model for the distribution of input-output pairs, learned from a fixed budget of evaluations of the forward map. This gives rise to an efficient surrogate model of the posterior distribution which can used within a gradient-based sampling scheme to obtain approximate posterior samples. We focus on the scenario where the forward map is function-valued, e.g. the solution of a partial differential equation. In this setting, we adapt a variational autoencoder as the generative prior model, trained on a basis decomposition of the functional output. This approach dramatically reduces the cost of the inference process, enabling efficient sampling as well as upstream activities such as experimental design. We demonstrate the efficacy of this approach on a number of benchmark problems.

1. **Introduction.**

1. Inverse problems
2. Surrogates and Generative modelling of the prior
3. Basis decompsition approach.

In applications we are often faced with the problem of wanting to identify initial conditions, boundary values or other free parameters within a differential equation (ordinary, partial or otherwise) based on measurements of the solution. Such *inverse problems* have motivated a wide array of

Inverse problems arising from differential equations (ordinary, partial or otherwise) are a challenging class of Partial differential equations (PDEs) provide a

2020 *Mathematics Subject Classification.* Primary: 58F15, 58F17; Secondary: 53C35.

Key words and phrases. Uncertainty quantification, inverse problems, Bayesian inference, variational inference, variational autoencoder.

PY was supported by Wave 1 of The UKRI Strategic Priorities Fund under the EPSRC Grant EP/T001569/1, particularly the "Digital Twins for Complex Engineering Systems" theme within that grant, and The Alan Turing Institute. ABD was supported by the Lloyds Register Foundation Programme on Data Centric Engineering and by The Alan Turing Institute under the EPSRC grant [EP/N510129/1].

*Corresponding author: Peter Yatsyshin.

versatile and expressive framework for modelling complex phenomena in physics, engineering and even social sciences. In

Such equations represent changes in system outputs by relating the output of the process to its derivatives. A particular realisation of the PDE is typically associated with a specific set of boundary conditions and a set of values of any free parameters entering the PDE. In the present contribution we address a class of inverse problems, where based on sparse and noisy observations of the PDE solution we want to infer either the parameters of the PDE model, such as boundary conditions or free parameters, or indeed the solution itself. Such inverse problems are highly important in engineering and scientific applications which combine theoretic descriptions with observations. The ubiquity and complexity of inverse problems for PDEs, places inference at the cornerstone of modern machine learning and the emerging fields of data-centric engineering, digital twins and physical surrogate models [Refs].

The relationship between the observed data and a set of parameters in the underlying PDE model can be arbitrarily complex. One way to learn complex non-linear relationships is to use deep neural networks (NN), which are very flexible and efficient in representing non-linear dependencies in high-dimensional spaces. Various NN models have seen high popularity in a number of machine learning applications [8]. However, many state of the art NN approaches often lack guarantees of convergence and error estimates, as well as requiring high volumes of training data and oftentimes suffering from interpretability issues [16].

In PDE inference, the underlying physical or engineering systems are often complex, making large quantities of observation data expensive to obtain. At the same time, available observation data is often corrupted by noise or incomplete. This places a requirement on the inference methodology to adequately propagate uncertainty from observations to the predicted model parameters [17]. Generative adversarial networks (GAN), offer a generative model, which is highly successful in image processing problems. GANs may be adapted to PDE inference by using the PDE as a regularisation constraint, and promising to enable uncertainty quantification [16, 15]. However, GAN training involves a minimax problem, and thus may suffer from mode collapse [8]. If this technical difficulty is overcome, GANs typically provide a generative model with high subjective quality of synthetic samples, but still do not allow to explicitly compute the likelihood functions, which limits their usability for inverse PDE problems [7].

Bayesian approaches treat PDE parameters as random variables (RVs) and aim to model their posterior distributions, conditioned on the observations. Bayesian models are naturally equipped with uncertainty quantification, are usually humanly interpretable, and allow one to incorporate prior knowledge about the parameters into the algorithms [4, 6]. However, classical Bayesian inference involves posterior sampling, which is usually done with expensive Markov Chain Monte-Carlo (MCMC) algorithms. Full posterior sampling in PDE inverse problems requires one to solve the actual PDE on every MCMC step to obtain the forward map of the sampler, which may be prohibitively costly.

A number of ideas have been put forward to reduce the computational load, including manifold sampling [5], probabilistic numerical algorithms for PDE likelihood estimation [9], Gaussian processs (GP) approximations for the joint likelihood over the solution and its derivatives [3], linearising the variational form of the PDE [6], and combining linearised variational form with a generative model [13]. However, in cases when the PDE solutions are obtained using a black-box solver whose internal workings are not directly accessible by the inference algorithm, the likelihood and its derivatives may not be computationally tractable at all.

In general, intractable likelihoods in Bayesian inference can be tackled by Approximate Bayesian Computation (ABC) [2]. Using ABC for PDE inference would require generating parameter samples from the prior, solving the PDE and then rejecting parameter samples which lead to solutions far from the observation data in an appropriate metric. Clearly, although ABC in principle allows to handle inference, the computational costs would admittedly be very high. Another general approach to intractable likelihoods is to use Bayesian Synthetic Likelihood (BSL) [12], which constructs a tractable parametric auxiliary model. However, the validity of BSL must be verified in each specific case by, e.g. establishing the existence of a central limit theorem.

In the present work we propose a new computationally inexpensive method for approximating the Bayesian posteriors of inverse PDE problems. Our main idea is to remove the computational complexity of solving the forward map, given by the PDE, during sampling. The forward map still has to be solved, just not during posterior sampling. Before even embarking on solving the inference problem, we train a surrogate generative model using a VAE. This surrogate model is capable of generating the solutions to PDE very cheaply, as well as giving us access to approximate probability density of the solutions. In other words, VAE acts as a prior, where the prior information are M PDE solutions, which we have obtained before seeing any inference data.

The idea of training a surrogate generative model to act as inference prior is quite appealing. We can view the PDE as defining a manifold of functions $\{u(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}\}$, where $u(\mathbf{x})$ is a PDE solution corresponding to some chosen initial or boundary conditions, constraints or any other PDE parameters. Often PDE solutions can be well approximated by expansions on dense vector spaces (e.g., Sobolev or Fourier spaces):

$$u(\mathbf{x}) = \sum_{n=0}^{\infty} c_n \phi_n(\mathbf{x}) \approx \sum_{n=0}^{K} c_n \phi_n(\mathbf{x}) = u_{\mathbf{w}}(\mathbf{x}), \tag{1}$$

where $\mathbf{w} = [c1 \ldots c_K]$ identifies the approximate PDE solution on the appropriate vector space. The above expression allows us to interpret inference of PDE solutions as regression to a functional space with a well-defined basis or frame. The functional prior on admissible $u_w(x)$ can then be approximated by a distribution over w: $w \sim p_\theta(w)$, where θ denotes the parameters of a suitable distribution family. Furthermore, we can set up highly expressive variational approximations to the prior using Variational Autoencoders (VAEs) [10], which treat $w \sim p_\theta(w)$ as a marginal of a latent variable model:

$$p_\theta(\mathbf{w}) = \int p_\theta(\mathbf{w}, \mathbf{z}) d\mathbf{z}. \tag{2}$$

The functional prior in (1) and (2) can be appropriately combined with problem-specific likelihoods and priors on other unknown PDE parameters to obtain a Bayesian hierarchical model for inference. We emphasize that our approach does not impose any requirements on the PDE solver, treating it as a black box. This is essential in many engineering applications, which typically involve the use of complicated PDE solvers. Moreover, using the VAE as a surrogate functional prior allows us to cheaply draw functions $u_{\mathbf{w}}(x)$ form the PDE prior, thus eliminating the need to solve the forward map on every step of the MCMC sampler, which is currently the strongest computational bottleneck to many inference problems. Furthermore, since VAE yields $p_\theta(\mathbf{w})$ as a tractable and differentiable density we can use highly efficient gradient-based MCMC samplers, reducing the computational cost of solving the inverse problem even further. Our main contributions are summarised in the following list:

- We propose a straightforwardly implementable framework, in which to approximately solve Bayesian inference problems for expensive forward problems;
- Proposed framework is able to approximate appropriate functional priors from black-box simulators, providing an efficient alterntive to approximate Bayesian computation. Note that the black-box simulator does not have to be linear;
- The VAE approximation to the probability measure provides direct access to the density, whose gradient can be computed and readily used in any MCMC simulation to solve the Bayesian inference problem;
- We demonstrate the methodology on a number of PDE examples, including a Bayesian experimental design problem for optimal sensor placement.

2. Approximate probability measure on function spaces.

In order to train the functional prior, we need an ensemble of N PDE solutions. Approximating the solutions following (1), we obtain N iid samples $\{\mathbf{w}_i\}_{i=1}^N$ in a K-dimensional vector space. We seek to obtain a variational approximation $p_\theta(\mathbf{w})$ to the distribution of these N samples. Very expressive distribution families can be obtained by taking a known family, e.g., Gaussian, and treating its parameters as non-linear functions of a latent random variable $\mathbf{z} \in \mathbb{R}^d$. When the non-linear function is expressed with a neural network (NN), one obtains a VAE [11].

As mentioned above, we work within a variational framework, where we approximate a distribution over $\mathbf{w}$ by introducing a latent variable $\mathbf{z}$ and marginalising the joint $p_\theta(\mathbf{w}, \mathbf{z})$ in (2). To set up a VAE, we first need to choose two parametric distribution families $p_\theta(\mathbf{x})$ and $q_\phi(\mathbf{x})$, which are coupled via the variational lower bound. Next, we need to set up two NNs, an encoder $\mathrm{e}(\mathbf{x})$ and a decoder $\mathrm{d}(\mathbf{x})$, with respective parameters η_e and η_d, so that by setting

$$\begin{aligned} \phi &= \mathrm{e}_{\eta_\mathrm{e}}(\mathbf{w}), \\ \theta &= \mathrm{d}_{\eta_\mathrm{d}}(\mathbf{z}), \end{aligned} \tag{3}$$

we turn the densities $p_\theta(\mathbf{w})$ and $q_\phi(\mathbf{z})$ into respective conditional densities, $p_\theta(\mathbf{w} \mid \mathbf{z})$ and $q_\phi(\mathbf{z} \mid \mathbf{w})$. Finally, we place a prior on the latent variable $\mathbf{z}$ and obtain the following hierarchical generative model:

$$\begin{aligned} \mathbf{z} &\sim p(\mathbf{z}), \\ \mathbf{w} &\sim p_\theta(\mathbf{w} \mid \mathbf{z}). \end{aligned} \tag{4}$$

The training of the VAE (4) consists of optimising the NN parameters in (3) by maximising the following variational lower bound:

$$\mathcal{L}(\eta_d, \eta_e) = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{w}_i)} \left[\log p_\theta(\mathbf{w}_i \mid \mathbf{z})\right] - \mathbb{KL}\left[q_\phi(\mathbf{z} \mid \mathbf{w}_i) \mid\mid p(\mathbf{z})\right], \tag{5}$$

where the last term is the KL-divergence between the latent posterior $q_\phi(z \mid x_i)$ and latent prior. In practice the sum over $\mathbf{w}_i$ in (5) is replaced with mini-batch average. The maximisation of (5) is similar in spirit to the expectation-maximisation algorithm. However, do to the complicated nature of dependencies in (3), (5) is usually maximised via a doubly-stochastic gradient ascent. We note that in practice maximising (5) requires extra care in estimating the gradient of the first term, which is prone to have high variance. The two most popular approaches to maximising (5) are reinforce algorithm and reparametrisation trick [11].

Examples which follow have the same VAE architecture for approximating the distribution of $\mathbf{w}$ in (1). We used a full-rank Gaussian as $p_\theta(\mathbf{x})$, a diagonal Gaussian as $q_\phi(\mathbf{x})$, a spherical Gaussian as latent prior $p(\mathbf{z})$, and two multilayer perceptrons as encoder and decoder NNs:

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(0, I_d), \\ p_\theta(\mathbf{w} \mid \mathbf{z}) &= \mathcal{N}(\mu_{\mathbf{z}}, \Sigma_{\mathbf{z}}), \\ q_\phi(\mathbf{z} \mid \mathbf{w}) &= \mathcal{N}(\mu_{\mathbf{w}}, \sigma^2_{\mathbf{w}} I), \\ \left[\mu_{\mathbf{w}}, \sigma^2_{\mathbf{w}}\right] &= \mathrm{e}_{\eta_e}(\mathbf{w}), \\ \left[\mu_{\mathbf{z}}, \Sigma_{\mathbf{z}}\right] &= \mathrm{d}_{\eta_d}(\mathbf{z}). \end{aligned} \tag{6}$$

Depending on the problem, we varied the number of hidden layers and the dimensionality of the latent $\mathbf{z}$.

We note that depending on the inference problem, the VAE can encode additional information. For example, if the target of inference are boundary conditions or PDE parameters, those can be appended to $\mathbf{w}$. If we wish to infer a term in the PDE, we can expand it using (1), and append those coefficients to the expansion of the PDE solutions when training the VAE.

3. Approximate Bayesian inference.

A typical inference problem for PDEs is reconstructing the full solution to the PDE from its sparse noisy observations. Usually the class of the noise distribution is known or assumed. We will assume noise to be uncorrelated Gaussian with mean zero and some unknown variance σ^2. Thus for the PDE with D-dimensional support, inference training set $\mathcal{D}$ is given by

$$\begin{aligned} \mathcal{D} &= \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_{\text{data}}}, \\ y_i &= u(\mathbf{x}_i) + \varepsilon_i,\ \varepsilon \sim \mathcal{N}(0, \sigma^2), \end{aligned} \tag{7}$$

where $\mathbf{y} = [y_1 \dots y_{n_{\text{data}}}]^T \in \mathbb{R}^{n_{\text{data}}}$ are the targets and $X = [\mathbf{x}_1 \dots \mathbf{x}_{n_{\text{data}}}] \in \mathbb{R}^{D \times n_{\text{data}}}$ is the design matrix, which contains the points in the PDE support at which the noisy solution is observed.

We may be looking to infer the noise variance σ^2 and solution $u(x)$, which are consistent with the training data (7). Alternatively, we may be after the initial or boundary conditions of the PDE or indeed after any constraint, parameter or term in the PDE. Following the Bayesian paradigm, we treat any such quantities of interest as random variables, and look to approximate their distributions. These should be consistent with the training data (7). The noise assumption (7) together with the model PDE give rise to the following likelihood:

$$p(\mathbf{y} \mid X, \sigma^2) = \mathcal{N}\left(y_i \mid u(\mathbf{x}_i), \sigma^2 I_{n_{\text{data}}}\right), \tag{8}$$

where $u_i(x)$ is the PDE solution that we may be looking to infer. As discussed, we assume a dense vector space to be associated with the PDE and use an approximate likelihood, replacing $u(\mathbf{x})$ with a K-term expansion (1):

$$p(\mathbf{y} \mid X, \sigma^2) \approx \mathcal{N}\left(y_i \mid u_{\mathbf{w}}(\mathbf{x}_i), \sigma^2 I_{n_{\text{data}}}\right). \tag{9}$$

Algorithm 1 Posterior with VAE prior

```
$\{u_i(\mathbf{x})\}_{i=1}^N \leftarrow N$ numerical PDE solutions, obtained using any black-box solver
$\mathcal{D} \leftarrow$ inference training data (7)
for $i = 1$ to $N$ do
    $\mathbf{w}_i \leftarrow$ truncated expansion (1) of $u_i(\mathbf{x})$
    if inference targets PDE parameters/constraints then
        append these parameters/constraints to $\mathbf{w}_i$
    end if
end for
$\eta_d \leftarrow$ decoder parameters of VAE (6), trained on $\{\mathbf{w}_i\}_{i=1}^N$
    return $p(\mathbf{w}, \mathbf{z}, \xi \mid \mathcal{D}, \eta_d) \leftarrow$ posterior (11)
```

Using Bayes' theorem, we can find the posterior on the K-dimensional vector of expansion coefficients $\mathbf{w}$:

$$p(\mathbf{w}, \sigma^2 \mid \mathbf{y}, X) \propto \mathcal{N}\left(y_i \mid u_{\mathbf{w}}(\mathbf{x}_i), \sigma^2 I_{n_{\text{data}}}\right) p(\mathbf{w}) p(\sigma^2), \tag{10}$$

where an approximate VAE prior $p(\mathbf{w}) \approx p_\theta(\mathbf{w})$ defined by (2) and (6) encodes a manifold of PDE solutions obtained independently from inference training set $\mathcal{D}$ in (7), and we also assume a sensible prior $p(\sigma^2)$ on the observation noise.

The posterior (10) is written in the form, where a marginal is taken over the latent variable $\mathbf{z}$. Since we intend to sample from the posterior using HMC, we should include $\mathbf{z}$ into the sample and marginalise while sampling. This leads to the final expression for the posterior, which uses VAE in (6) as prior:

$$\begin{aligned} &p(\mathbf{w}, \mathbf{z}, \xi \mid \mathcal{D}, \mathrm{d}_{\eta_d}) \propto \\ &\mathcal{N}\left(y_i \mid u_{\mathbf{w}}(\mathbf{x}_i), \text{softplus}(\xi) I_{n_{\text{data}}}\right) \mathcal{N}\left(\mathbf{w} \mid \mu_{\mathbf{z}}, \Sigma_{\mathbf{z}}\right) \mathcal{N}(\mathbf{z} \mid 0, I_d) \mathcal{N}(\xi \mid 0, 1), \\ &[\mu_{\mathbf{z}}, \Sigma_{\mathbf{z}}] = \mathrm{d}_{\eta_d}(\mathbf{z}), \end{aligned} \tag{11}$$

where we made a transformation $\sigma^2 = \text{softplus}(\xi)$ with $\text{softplus}(x) = \log(1 + e^x)$.

As mentioned above, the vector $\mathbf{w}$, which contains the basis expansion of the PDE solution, may be appended with any PDE parameters or expansions of terms in the PDE during the VAE training and inference. The general algorithm for obtaining the posterior is given in algorithm (1). After the posterior (11) is obtained, one can sample from it using HMC, while marginalising over $\mathbf{z}$ and those elements of $\mathbf{w}$ which are not the targets of inference.

4. Numerical Experiments.

In the present section we demonstrate the inference method on several examples. We emphasise that PDE solutions are only required to train the VAE. Moreover, these can be obtained using any black-box PDE solver. Once trained, the VAE can be re-used for inference, leading to high potential savings in computational cost.

4.1. Boundary value problem in 1D.

Consider the following example of a boundary value problem (BVP) on $x \in [-L/2, L/2]$, where $L = 2$ [16]:

$$\begin{aligned} &u_{xx} - u^2 u_x = f(x), \\ &f(x) = -\pi^2 \sin(\pi x) - \pi \cos(\pi x) \sin^2(\pi x), \\ &u(-1) = a, \ u(1) = b, \end{aligned} \tag{12}$$

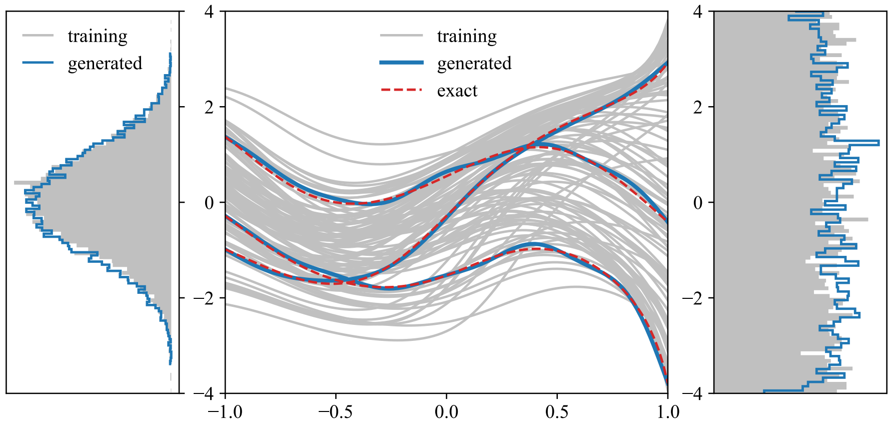

FIGURE 1. Encoding solutions to PDE (12). Grey: 100 of 10^4 solutions to (12), computed at left/right BC, distributed normally/uniformly – as shown by histograms on the sides. Blue: 3 of 10^4 VAE-generated BC-solution pairs, with histograms of all 10^4 BC on the sides. Red: 3 numerical solutions, obtained at the same BC as the 3 generated solutions shown.

where the boundary conditions (BC) are RVs:

$$\begin{aligned} a &\sim \mathcal{N}(-3, 1), \\ b &\sim \mathcal{U}([0, 1]). \end{aligned} \tag{13}$$

We can set up a pedagogical inference problem by noting that when $a \equiv a_e = 0$ and $b \equiv b_e = 0$, the BVP (12) has an exact solution $u_e(x) = \sin(\pi x)$. The inference problem we will solve is to reconstruct the BC ($a = b = 0$) from sparse noisy observations of the solution $u_e(x)$. The inference training data is given by $n_{\text{data}} = 10$ noisy observations at random locations x_i:

$$\begin{aligned} \mathcal{D} &= \{(x_i, u_e(x_i) + \varepsilon_i\}_{i=1}^{n_{\text{data}}}, \\ u_e(x) &= \sin(\pi x), \\ x_i &\in \mathcal{U}([-1, 1]), \\ \varepsilon_i &\sim \mathcal{N}(0, 0.1^2). \end{aligned} \tag{14}$$

We can obtain the expansion (1) for a solution $u(x)$ by combining half-Fourier series with a linear function:

$$\begin{aligned} u(x) &= u_0(x) + u_{BC}(x), \\ u_0(x) &= \sum_{n=1}^{\infty} c_n \sin \frac{n\pi x}{L}, \\ u_{BC}(x) &= \frac{b-a}{L}x + \frac{a+b}{2}, \end{aligned} \tag{15}$$

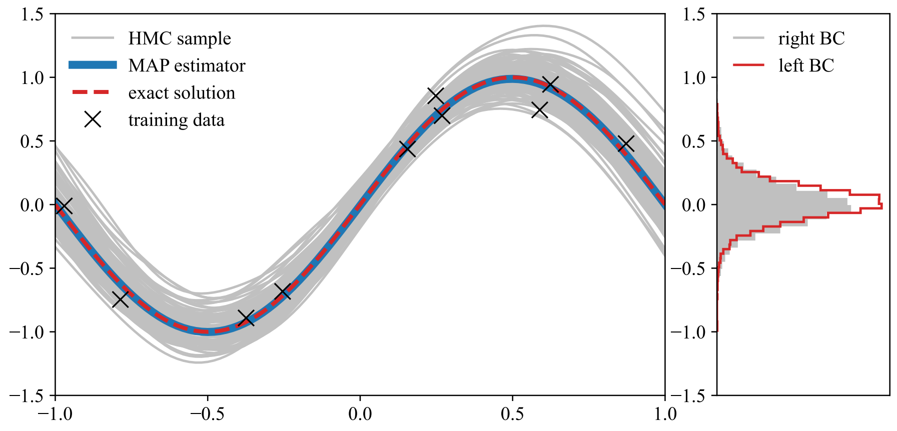

FIGURE 2. Bayesian inference of BC in BVP (12). Left pane: Grey: 100 of 10^4 posterior HMC samples. Blue: MAP estimator of solution $u_e(x)$. Red: exact solution $u_e(x)$. Markers: inference training data. Right pane: histograms of HMC samples of BC.

where the expansion coefficients are given by the Discrete Fourier Transform (DFT):

$$c_n^{(i)} = \frac{1}{L}\int\limits_{-L/2}^{L/2} (u^{(i)}(x) - u_{BC}^{(i)}(x))dx. \tag{16}$$

An approximate prior for the inference problem is obtained by first solving the BVP (12) N times, with different BCs a and b, sampled from (13). This gives a prior set of solutions $\{u_i(x)\}_{i=1}^N$. This prior set is then approximated by computing the expansion (15) for each $u_i(x)$, truncated at the M-th Fourier term. This gives rise to the VAE training set $\{\mathbf{w}_i\}_{i=0}^N$, where $\mathbf{w}_i = [a^{(i)}, b^{(i)}, c_1^{(i)}, \dots c_M^{(i)}]^T \in \mathbb{R}^{M+2}$.

In figure 1 we illustrate that the VAE is capable of encoding the BVP (12) with BC distributed according to (13). We trained the VAE on a sample of 10^4 solutions, truncating (15) at $M = 10$ terms. The central pane of figure 1 shows a subset of 100 solutions in grey, and the side panes show the histograms of BC, also in grey. To assess the quality of VAE encoding, we generated a sample of the 10^4 solutions from the VAE. The histograms of the BC of the generated curves are superimposed in blue on the side panes of figure 1 and demonstrate that the BC distributions have been properly captured by the VAE encoding. Additionally, on the central pane we plot 3 generated solutions in blue, alongside numerical solutions, obtained at the same BC in red. One can see that visually VAE indeed provides a good representation of the manifold of solutions to the BVP problem (12) with BC given by (13). The computational saving offered by replacing numerical solution of the BVP with generation from the VAE is illustrated by the fact that it took about 20 minutes to computing 10^4 solutions to the PDE (12) on a standard desktop, while drawing the same number of samples from the trained VAE takes less than 1 second.

The inference results are summarised in figure 2, which shows a sample from the posterior (grey), along with a Maximal a-Posteriori (MAP) estimator of the

Algorithm 2 Posterior update algorithm

```
$\{\mathbf{w}_i^{(0)}\}_{i=1}^N \leftarrow$ initial sample of VAE training data
$\{\mathcal{D}_i\}_{i=0}^P \leftarrow$ mini-batches of inference training data
for $k = 1$ to $P$ do
    $\eta_d^{(k)} \leftarrow$ decoder parameters of VAE (6), trained on $\{\mathbf{w}_i^{(k-1)}\}_{i=1}^N$
    $p^{(k)}(\mathbf{w}, \mathbf{z}, \xi \mid \mathcal{D}_k, \eta_d^{(k)}) \leftarrow$ posterior (11)
    $\{\mathbf{w}_i^{(k)}\}_{i=1}^N \leftarrow$ $N$-sample from the marginal of $p^{(k)}(\mathbf{w}, \mathbf{z}, \xi \mid \mathcal{D}_k, \eta_d^{(k)})$
end for
    return $\{\mathbf{w}_i^{(P)}\}_{i=1}^N$
```

solution and a posterior average. The posterior sample of the boundary conditions is represented by the two histograms on the right. We can observe a very good agreement between posterior average and the ground truth, both in terms of the solution $u_e(x)$ and in terms of the boundary conditions. We note that increasing the amount of observation data leads to posterior contraction, as expected.

4.1.1. *Iterative posterior updates.* In cases when prior information is poor or the prior does not adequately capture the inference ground truth, but we do have sufficient inference training data, we can the inference data to improve the quality of the VAE through iterative re-training. Of-course, doing so will incur additional computational costs of training several VAEs.

As an example, consider again the problem of inferring the BC of $u_e(x)$ in (14), but this time assume that the prior distributions of left- and right-BC are no longer given by (13), but instead are off of the ground truth:

$$\begin{aligned} a &\sim \mathcal{N}(-2, 1), \\ b &\sim \mathcal{N}(2, 1), \end{aligned} \tag{17}$$

The prior given by (17) is poor, because the true BC ($a_e = b_e = 0$) lie in its tails. Using such prior for inference will lead to high variance and errors in predictions. In order to retrain the VAE, we sub-sample a mini-batch from the available inference training data, and draw a sample from the predictive posterior (11), trained only on the mini-batch data (i.e., n_{data} in (11) is equal to the mini-batch size). We then use the predictive posterior sample to train a new VAE, which will have improved representation of the ground truth. Algorithm 2 expresses P such updates.

The algorithm 2 is also illustrated in figure 3 for our inference problem with mini-batches of size 2. Top left panel shows a draw from the predictive posterior with initial VAE and a mini-batch of size 2 (grey). The MAP estimator is shown in blue, and has poor agreement with the ground truth (red). We plot the data mini-batch with crosses. The draw from this predictive posterior was used to train a new VAE, which in turn was used to obtain a predictive posterior with the second mini-batch of training data. We performed a total of 5 updates, exhausting the available inference training data. The final posterior is shown in the top right panel of figure 3 and captures the ground truth quite well. The bottom panel of figure 3 shows histograms of posterior samples for BC, for each of the VAE updates. We observe a contraction of these histograms towards the distribution, centred around the true BC.

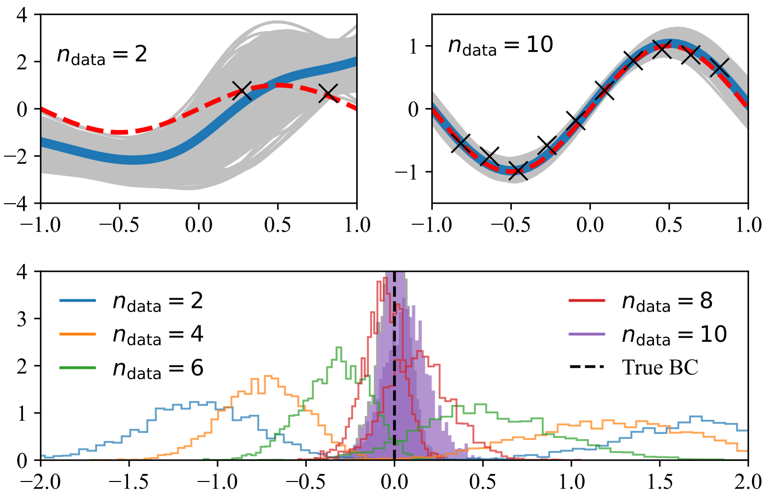

FIGURE 3. Updating VAE during inference. The legend of top left and -right panes is the same as in figure 2 Top left: inference on two data points with a bad initial prior. Top right: inference on ten points after consecutive VAE updates. Bottom: contracting predictive posterior of BC during VAE updates.

4.2. **Diffusion in 2D.**

We may be interested in learning some unknown functional term in a PDE from the sparse observations of its solution. In principle, we can approximate the term in question with an expansion in an appropriate basis or frame and cast the problem as inference of the expansion coefficients. As an example, consider inference of the diffusion coefficient in the following Poisson equation on a unit square $\mathbf{x} = [x, y]^T \in \Omega = [0, 1] \times [0, 1]$:

$$-\nabla \cdot (\kappa(x, y)\nabla u(x, y)) = f, \tag{18}$$

where we assume a Dirichlet BC $u|_{\partial\Omega} = 0$. For simplicity we keep the source term constant, $f = \text{Const}$. The diffusion coefficient $\kappa(x, y)$ must be non-negative, bounded and with a bounded derivative. To satisfy this requirement, we can set $\kappa(x, y) = \exp c(x, y)$, where $c(x, y) \in C^1(\Omega)$, which is significantly less restrictive. Equation (18) can be solved with a finite element spatial discretisation scheme in combination with a Krylov linear solver [1].

Admissible functions $c(x, y)$ may be generated, e.g., from the following Gaussian process with a zero mean and a squared exponential kernel function:

$$\begin{aligned} &\kappa(x, y) = \exp c(x, y), \\ &c(\mathbf{x}) \sim \mathcal{GP}\left(0, K(\mathbf{x}_1, \mathbf{x}_2)\right), \\ &K(\mathbf{x}_1, \mathbf{x}_2) = a \exp \left(\mathbf{x}_1 - \mathbf{x}_2\right)^2/\xi \end{aligned} \tag{19}$$

We have calibrated the kernel parameters $a = 2.0$ and $\xi = 0.25$ and the source term $f = 50.0$, to achieve visually interesting droplet-shaped solutions $u(x, y)$. Different realisations of $c(x, y)$ from (19) lead to different diffuse drops inside the domain Ω. An example is given by the top two rows of panes in figure 4.

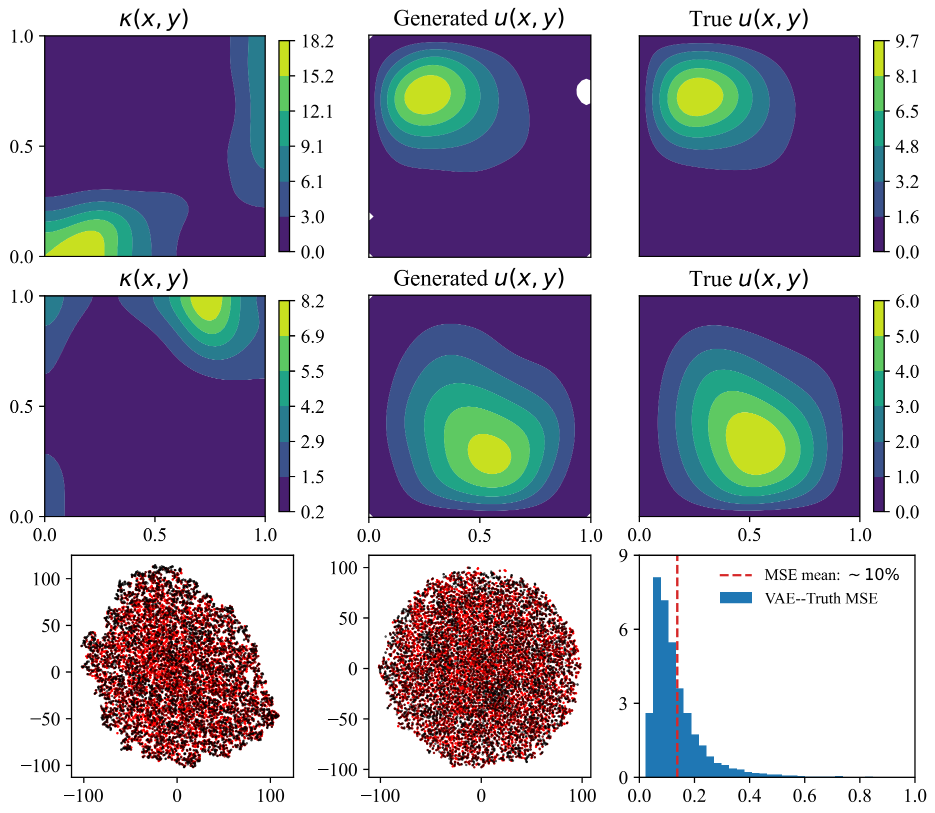

FIGURE 4. VAE approximation of the joint distribution over $c(x,y)$ and $u(x,y)$ in (18)–(20). Top two rows: $\kappa(x,y)$, generated from VAE (left pane), jointly generated solution $u(x,y)$ (middle pane) and true numerical solution (right pane). Bottom row: t-SNE reduction of 10^4 generated (red) and 10^4 true (black) solutions $u(x,y)$ (left pane); same for $\kappa(x,y)$ (middle pane); histogram of L^2-norms between generated and true solutions, computed in real space.

Assuming that we have sparse noisy observations u_i, $i = 1 \dots N$, of a given solution $u(x,y)$ at N discrete points inside the domain Ω, we aim to reconstruct the log-coefficient function $c(x,y)$, which gave rise to the observed solution. We represent $u(x,y)$ and $c(x,y)$ by their respective DFTs. To satisfy the Dirichlet boundary conditions on $u(x,y)$, we expand it in *sin*-series:

$$u(x,y) = \left(\sum_{n=1}^{M_u} a_n \sin(n\pi x)\right) \times \left(\sum_{n=1}^{M_u} b_n \sin(n\pi x)\right), \tag{20}$$

$$c(x,y) = \left(\sum_{n=0}^{M_\kappa} a_n \cos(n\pi x)\right) \times \left(\sum_{n=0}^{M_\kappa} b_n \cos(n\pi y)\right).$$

The VAE, which represents the joint distribution over $c(x,y)$ and $u(x,y)$, can be trained on the DFTs of a large number of realisations of $c(x,y)$, drawn from (19), and the corresponding DFTs of $u(x,y)$, obtained by numerically solving (18).

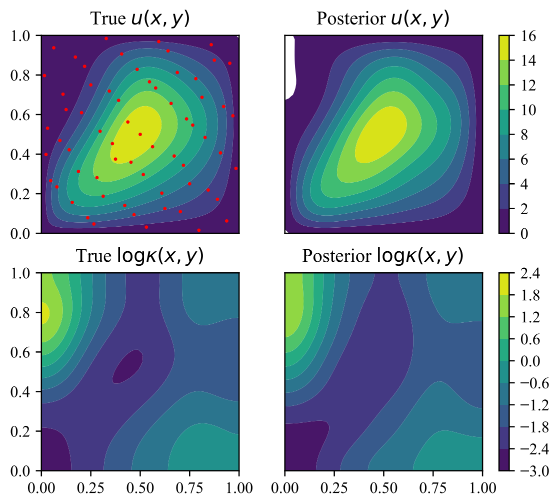

FIGURE 5. Inference of the solution $u(x, y)$ and the diffusion coefficient $\kappa(x)$ in PDE (18). Red markers show 64 points used for inference. Different panes show posterior averages over 10^3 posterior samples. The relative $L_2(\Omega)$-norm errors are $\|u_{\text{truth}} - u\|/\|u_{\text{truth}}\| = 0.02$ and $\|c_{\text{truth}} - c\|/\|c_{\text{truth}}\| = 0.07$

Figure 4 shows several generated solutions from a VAE, which was trained on 10^4 DFT expansions of pairs $c(x, y)$ and $u(x, y)$, expanded according to (20) with $M_u = M_\kappa = 6$. The top two rows of panes show two examples of $\kappa(x, y)$, followed by the true numerical solutions $u(x, y)$ and its generated counterpart. We see that our VAE produces $u(x, y)$, which are visually close to the true PDE solutions. A better quality control of the VAE can be performed by using a clustering algorithm to try and separate the generated and true data. We used a non-parametric dimensionality reduction algorithm t-SNE [14] on two concatenated data sets: the VAE training data and VAE-generated data set of the same size. Our purpose here was to verify that both data sets are samples from the same manifold. As can be seen from the first and second panes of the third row, t-SNE fails to separate the generated $\kappa(x, y)$ and $u(x, y)$ from the training data. This further attests to the quality of the trained VAE. Finally, we quantify the error between the generated and true data by computing a histogram of L^2-norms of difference between the true solutions and those generated from the VAE. This is depicted in the third pane of the bottom row of figure 4, and shows that the expected error is about 10%. Note that the error depicted is computed in real space, and is thus influenced by the truncation of the Fourier series. One can reduce the truncation error by increasing M_κ and M_u, while the error of VAE can be reduced by training on larger data sets.

To demonstrate inference, we first set up the ground truth by first sampling a random $\kappa_{\text{truth}}(x, y)$ from (19) and solving the PDE (18) numerically to find $u_{\text{truth}}(x, y)$. We simulate sparse observation data u_i by evaluating $u_{\text{truth}}(x, y)$ at $N = 64$ points (x_i, y_i), $i = 1 \dots N$, generated from a Sobol sequence in 2D, and adding white noise:

$$u_i = u(x_i, y_i) + 0.001\varepsilon_i, \tag{21}$$

$$\varepsilon_i \sim \mathcal{N}(0, 1). \tag{22}$$

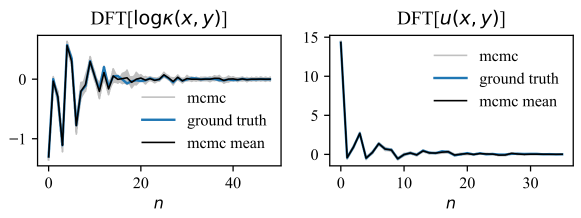

FIGURE 6. Inference of the diffusion coefficient and solution from figure 5 in Fourier space. The DFTs of $c(x, y)$ and $u(x, y)$ defined in (20) are ordered in terms of increasing frequencies. The MSE for DFT$[\kappa]$ is 8.8% and for DFT$[u]$ it is 1.5%.

By MCMC sampling the posterior in equation (11), we obtain the DFT of $u(x, y)$ and $\kappa(x, y)$ in equation (20). The results are summarised in figure 5, which shows the true $u(x, y)$ and $\kappa(x, y)$, along with posterior averages. Note that the quality of inference of $u(x, y)$ is higher than that of $\log \kappa(x, y)$, as evidenced by the relative errors given in the figure caption. This is a manifestation of the low sensitivity of the solution to the diffusion equation to the diffusion coefficient. The inference error measured in the real space in terms of the relative $L_2(\Omega)$-norms, contains the approximation error due to truncated Fourier series (20). It is therefore instructive to consider inference in the Fourier space.

Figure 6 shows the coefficients of expansion of c and u given in (20), ordered according to the increasing frequency. We note again that DFT$[\kappa]$ has higher uncertainty and MSE error than DFT$[u]$, which is due to the sensitivity of the PDE (18). More interestingly, note that higher frequencies contribute the most to both uncertainty and error. This is not surprising, because the observation noise affects higher frequencies the most. Ideally, the cut-off of the Fourier series (20) should be commensurate with the observation noise.

4.2.1. *Design of experiments.* An important and interesting question concerns choosing the free parameters of the statistical model to improve the quality of inference. Such experimental design tasks are usually quite computationally costly, requiring one to solve the forward map multiple times when evaluating likelihoods inside optimisation loops. Therefore, having access to functional priors with tractable likelihood functions, which are cheap to evaluate, can potentially make experimental design computationally much more accessible.

Assume that we already have K sparse observations of a PDE solution $u(\mathbf{x})$, which form our existing experimental data $\mathcal{D} = \{(\mathbf{x}_i, u_i)\}_{i=1}^{K}$, where $u_i = u(\mathbf{x}_i) + \varepsilon_i$ is the noisy observation with an appropriately chosen observation noise model. But now we also have means to obtain P additional observations of $u(\mathbf{x})$ at P new points $\mathbf{d}_i$, $i = 1 \dots P$ of our choosing. The experimental design task is to optimally choose the coordinates for $d = [\mathbf{d}_1 \dots P]$ to maximise the information gain about the solution. In engineering such tasks may be associated with the placement of P additional sensors inside a medium, whose physical behaviour is described by the PDE.

Recall that within our statistical model, the PDE solution is represented by an expansion on a frame of N features as given in (1) and the observation noise is assumed to be iid Gaussian with variance σ^2. The joint data likelihood is given by

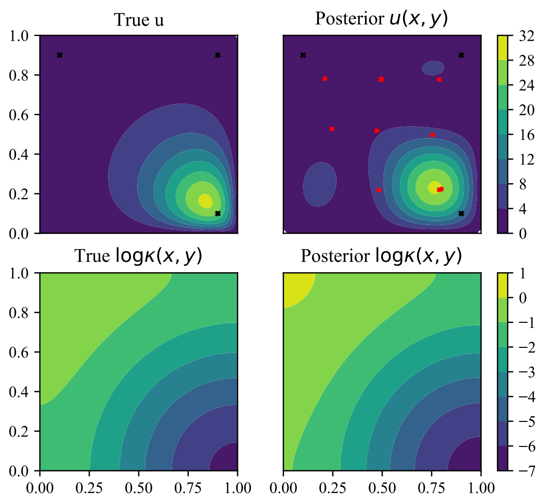

FIGURE 7. Left column of panels shows the ground truth for inference and 3 initial measurement points (black symbols). Right column shows the inference posterior average, where the posterior was found by placing 10 additional measurement points (red symbols), whose coordinates were obtained by minimising (31).

$p(\mathcal{D}) = \prod_{i=1}^{K} p(u_i \mid \mathbf{x}_i)$, with the likelihood of each data point $(u, \mathbf{x})$ given by

$$p(u \mid \mathbf{x}) = \mathcal{N}(u \mid u(\mathbf{x}), \sigma^2), \tag{23}$$
$$u(\mathbf{x}) = \sum_{n=0}^{N} \gamma_n \phi_n(\mathbf{x}),$$

where ϕ_i are our features. We approximate the joint prior distribution on $\gamma = [\gamma_1 \dots \gamma_N]^T$ with a VAE. In order to simplify notation, in this section we denote the posterior on γ due to existing measurements as

$$p(\gamma) := p(\gamma \mid \mathcal{D}). \tag{24}$$

According to (23), when P new observation sites $d = [\mathbf{d}_1 \dots \mathbf{d}_P]$ are set, the joint likelihood of the new measurements $y = [y_1 \dots y_P]$, where $y_i = u(\mathbf{d}_i) + \varepsilon_i$ and $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ is given by

$$p(y \mid \gamma, d) = \mathcal{N}(y \mid u(d), \sigma^2 I_P), \tag{25}$$
$$u(d) = \left[\sum_{n=0}^{N} \gamma_n \phi_n(\mathbf{d}_1), \cdots \sum_{n=0}^{N} \gamma_n \phi_n(\mathbf{d}_P) \right]^T$$

where I_P is the P-dimensional identity matrix. We shall denote the updated posterior due to P new measurements y taken at sites d by $p(\gamma \mid y, d)$.

We can formulate the utility function for d by requiring that it maximise the expected information gain between $p(\gamma)$ and $p(\gamma \mid y, d)$, which is equivalent to

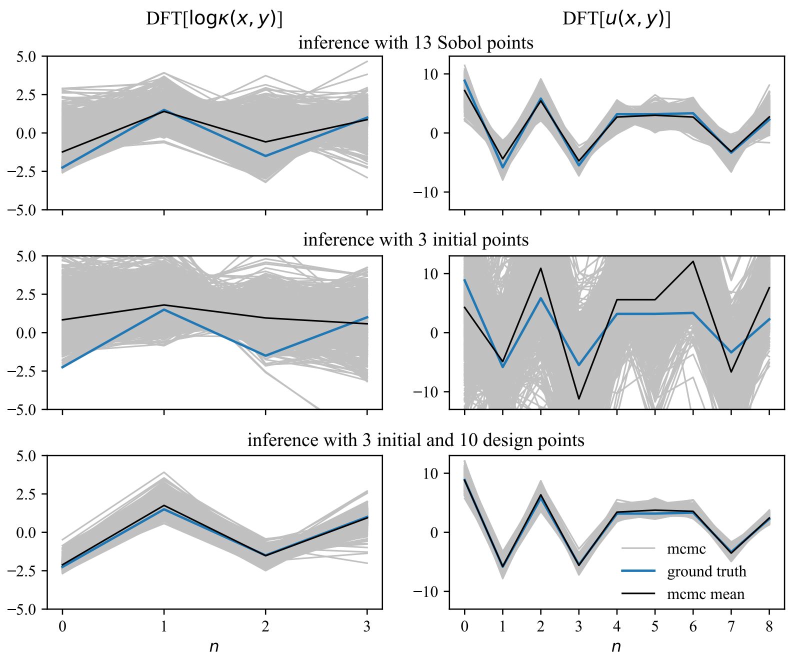

FIGURE 8. Inferred DFT coefficients of $\log \kappa(x, y)$ and $u(x, y)$. For each pair of plots, the number of inference training points is specified above each pair of DFT[$\log \kappa(x, y)$] – DFT[$u(x, y)$] plots. We used inference with 13 Sobol points to benchmark that with 3 initial and 10 design points.

maximising the expected KL-divergence:

$$\begin{aligned} U(d) &= \mathbb{E}_{p(y|d)} \mathbb{D}_{KL}[p(\gamma \mid y, d) \mid\mid p(\gamma)] \\ &= \int d\gamma \int dy \, (\log p(\gamma \mid y, d) - \log p(\gamma))\, p(\gamma, y \mid d) \end{aligned} \tag{26}$$

The expectation in (26) is computed over the predictive distribution of the new (yet unobserved) data $p(y \mid d)$. Applying Bayes theorem

$$p(\gamma \mid y, d) = \frac{p(y \mid \gamma, d) p(\gamma)}{p(y \mid d)}, \tag{27}$$

we cast the expected information gain in a form amenable to estimation:

$$\begin{aligned} U(d) &= \int d\gamma \int dy \, (\log p(y \mid \gamma, d) - \log p(y \mid d))\, p(\gamma, y \mid d) \\ &= \mathbb{E}_{p(y,\gamma|d)} \log p(y \mid \gamma, d) - \mathbb{E}_{p(y|d)} \log p(y \mid d). \end{aligned} \tag{28}$$

Notice that gradient-based maximisation of the utility (28) with respect to d may suffer from high variance of the gradients because of the need to differentiate through the density $p(y, \gamma \mid d)$, which is used to compute the expectations. Here we get around this problem by using Bayesian optimisation, which fits an easily differentiable surrogate function to $U(d)$.

In practice, we can estimate the density of the predictive distribution from the marginal:

$$p(y \mid d) = \int d\gamma \, p(y, \gamma \mid d) \approx \frac{1}{M} \sum_{j=1}^{M} p(y \mid \gamma_j, d), \tag{29}$$

where $\gamma_j \sim p(\gamma)$. This gives rise to the following unbiased utility estimator:

$$\tilde{U}(d) = \frac{1}{K} \sum_{i=1}^{K} \log p(y_i \mid \gamma_i, d) - \frac{1}{K} \sum_{i=1}^{K} \log \left(\frac{1}{M} \sum_{j=1}^{M} p(y_i \mid \gamma_{ij}, d) \right), \tag{30}$$

where $y_i \sim p(y \mid \gamma_i, d)$, with the density given by (25) and $\gamma_{ij} \sim p(\gamma)$ for $i = 1 \ldots N$, $j = 1 \ldots M$.

When the posterior $p(\gamma)$ is given in terms of an MCMC sample of size K, assuming that the MCMC chains which produced the sample are ergodic, we can use a biased estimator to significantly reduce the computational cost:

$$\tilde{U}_{biased}(d) = \frac{1}{N} \sum_{i=1}^{K} \log p(y_i \mid \gamma_i, d) - \frac{1}{K} \sum_{i=1}^{K} \log \left(\frac{1}{K} \sum_{j=1}^{K} p(y_i \mid \gamma_j, d) \right), \tag{31}$$

where γ_i, $i = 1 \ldots K$ is our MCMC sample from the posterior $p(\gamma)$ in equation (24), and y_i is one sample, drawn from $p(y \mid \gamma_i, d)$.

To illustrate experimental design, consider inference for the PDE in (18) again. We set the true $c(x, y) \equiv \log \kappa(x, y)$ by the Fourier expansion in (20) with $M_\kappa = 1$, $a_0 = -b_0 = -1.5$ and $a_1 = b_1 = 1.0$. The true $u(x, y)$ is obtained numerically and shown together with the true $\log \kappa(x, y)$ in the left row of figure 7. The solution is sharply peaked, which should make uniform or Sobol inference points a poor choice. As existing experimental data we choose 3 points, located at $(0.1, 0.9)$, $(0.9, 0.1)$ and $(0.9, 0.9)$ (black crosses in figure). The locations were deliberately chosen off of the center of the "drop" and in the bulk of surrounding "vapour". By maximising the utility function (31), we placed 10 design points. The optimal placement of these new measurement sites is shown with red symbols in figure 7. Notice how the algorithm produces nearly uniform grid of design sites, while at the same time placing several sites exactly at the peak of $u(x, y)$. It is also noticeable from the figure that a missing "grid" point in the bottom left leads to inaccuracy in the inferred $u(x, y)$ inaccuracy.

To get a sense of uncertainty quantification and the efficiency of Bayesian experimental design, it is instructive to look at the MCMC samples of the inferred Fourier coefficients. These are represented in figure 8, ordered from high to low harmonics. To benchmark the design sites, we generated the same number of measurement sites from the Sobol sequence. The Sobol sequence is designed to maximise the distance between the points in nD, covering the domain approximately uniformly. As can be seen from the top row of plots, 13 Sobol sites produce adequate inference. Even though the 3 initial observation sites clearly cannot provide reliable inference (see middle row of plots), they can nevertheless reliably inform the placement of additional measurement sites via the information maximisation criterion. The bottom row of plots shows inference at initial and design points. The result is clearly superior to Sobol points: the uncertainty is lower, and the posterior means are closer to the ground truth as evidenced by the following MSEs. For 13 Sobol sites, the MSE

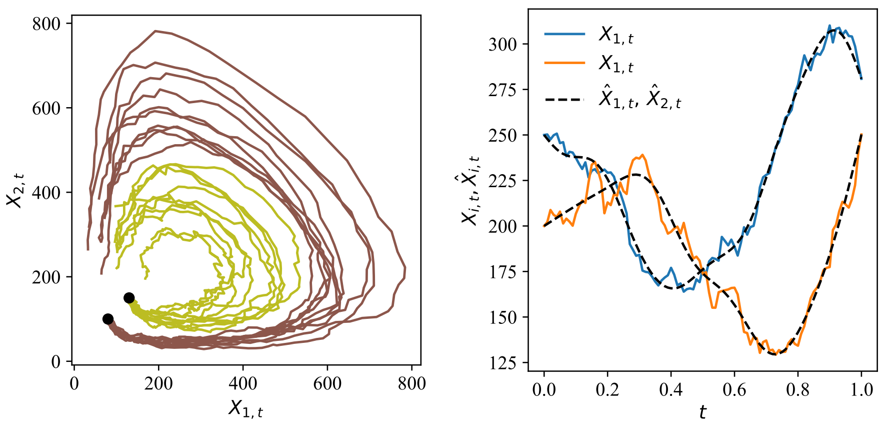

FIGURE 9. Left pane: Phase space trajectories of the stochastic predator-prey model (35). Showing simulations for two different initial conditions, designated by black circles. Right pane: time-dependent stochastic trajectories (solid curves) and approximate DFT representation (36) with $M = 10$ harmonics (dashed curves)

of DFT[$\log \kappa$] is 43% and the MSE of DFT[u] is 17%, while for the design sites the same MSEs are 9% and 6%, respectively.

5. **Stochastic dynamical system.** Another area of active research, where approximate VAE-based posterior can be highly beneficial is inference on models with intractable likelihoods, such as dynamical systems described by stochastic partial differential equations. The intractability of the likelihoods in such systems stems from the need to marginalise the transition probabilities over all possible trajectories of the stochastic system.For illustration, consider an Itô‘ SDE:

$$dX_t = a(X_t)dt + dW_t, \tag{32}$$

where W_t is a Wiener process and $a : \mathbb{R} \to \mathbb{R}$ is a non-linear drift. Suppose we have N sparse observations of a trajectory of such system, $\{y_i = X_{t_i} + \xi_i\}_{i=1}^N$, where $\xi_i \sim \mathcal{N}(0, \delta^2)$ with some fixed variance δ. The joint density over observations y_i and latent realisations X_{t_i} is given by

$$\begin{aligned} p(y_1, y_2 \ldots, X_{t_1}, X_{t_2}, \ldots) &= p(y_1 \mid X_{t_1})p(y_2 \mid X_{t_2}) \ldots p(X_{t_1}, X_{t_2}, \ldots X_{t_N}) \\ &= p(y_1 \mid X_{t_1}) \ldots p(X_{t_N} \mid X_{t_{N-1}}) \ldots p(X_{t_2} \mid X_{t_1}), \end{aligned} \tag{33}$$

where in the second equality we used the Markov property of the Wiener process. When $\delta t = t_i - t_{i-1}$ is small, the densities $p(X_{t_i} \mid X_{t_{i-1}})$ may be approximated using, e.g. an Euler-Maruyama scheme

$$p(X_{t_i} \mid X_{t_{i-1}}) \approx \mathcal{N}(X_{t_i} \mid X_{t_{i-1}} + \delta a(X_{t_{i-1}}), \delta t), \tag{34}$$

but such linear approximation breaks at large δt. In general, the conditional densities $p(X_{t_i} \mid X_{t_{i-1}})$ are intractable, because they require marginalising over all possible trajectories between $X_{t_{i-1}}$ and X_{t_i}.However, using a VAE we can directly approximate the joint probability density $p(X_{t_1}, X_{t_2}, \ldots X_{t_N})$ above with a tractable expression.

For a complete example of inference, consider a noisy predator-prey model, which describes the joint temporal dynamics of two coexisting populations: predator, X_t, and prey, Y_t:

$$\begin{aligned} dX_{1,t} &= (\theta_1 X_{1,t} - \theta_2 X_{1,t}X_{2,t})dt + \sqrt{\theta_1 X_{1,t}}dW_t^{(1)} - \sqrt{\theta_2 X_{1,t}X_{2,t}}dW_t^{(2)}, \\ dX_{2,t} &= -(\theta_3 X_{2,t} - \theta_2 X_{1,t}X_{2,t})dt - \sqrt{\theta_3 X_{2,t}}dW_t^{(3)} + \sqrt{\theta_2 X_{1,t}X_{2,t}}dW_t^{(2)}, \end{aligned} \tag{35}$$

where $\theta = [\theta_1, \theta_2, \theta_3]$ are the model parameters, which we keep fixed at $\theta = [5, 0.035, 6]$. We also fix the the time horizon to $t \in [0, 1]$. Several sample trajectories from the system (35) are shown in the left pane of figure 9. We are interested in the problem of inferring the trajectory of (35), given N sparse noisy observations.

In order to encode the distribution of stochastic trajectories in (35), we first approximate each trajectory using DFT $\hat{X}_i(t) \approx X_{i,t}$:

$$\begin{aligned} &\hat{X}_i(t) = \sum_{n=1}^{M} c_i^n \sin n\pi t + (a_i - b_i)t + a_i, \\ &a_i = X_{i,0},\ b_i = X_{i,1}. \end{aligned} \tag{36}$$

An example of a trajectory and its smooth approximant (36) is shown in the right pane of figure 9 and demonstrates the smoothing effect that the transformation has. Still all the important features are captured by representation (36).

Now we can simulate a large dataset of trajectories of (35) and train a VAE to approximate $p_\theta(\mathbf{w} \mid \mathbf{z})$, where $\mathbf{w} = [\{c_1^n\}_n, a_1, b_1, \{c_2^n\}_n, a_2, b_2]$. The marginal density $\int p_\theta(\mathbf{w} \mid \mathbf{z})p(\mathbf{z})d\mathbf{z}$ provides an tractable and differentiable approximation to the density of the probability measure on the manifold of stochastic trajectories of (35). We obtained the training data for the VAE by first generating an ensemble of 1000 normally distributed initial conditions, and simulating 100 stochastic trajectories of (35) per initial condition. This was done using Euler-Maruyama scheme with a sufficiently small step to ensure numerical stability, which in our case was 10^-2. We then computed the expansion (36) for each simulated trajectory, obtaining the VAE training data set $\mathbf{w} = [\{c_1^n\}, a_1, b_1, \{c_2^n\}, a_2, b_2]$. As in previous examples, we verified that the VAE indeed captures the manifold of stochastic trajectories, and properly encodes the original distribution of initial conditions.

For inference, assume that we have n_{data} noisy observations of the trajectory at distinct time stamps $\{t_i\}_{i=1}^{n_{data}}$, so that our inference training data is $\mathcal{D} = \{(\tilde{X}_{1,t_i}, \tilde{X}_{2,t_i})\}_{i=1}^{n_{data}}$, where $\tilde{X}_{j,t_i} = X_{j,t_i} + \varepsilon_i$ for $j \in \{1, 2\}$ and $\varepsilon_i \sim \mathcal{N}(0, 1)$. We can sample the posterior over the approximate trajectories (36), the observation noise ξ and the latent auxiliary variable $\mathbf{z}$, marginalising over $\mathbf{z}$:

$$\begin{aligned} &p(\mathbf{w}, \mathbf{z}, \xi \mid \mathcal{D}) \propto \\ &\mathcal{N}\left(\tilde{X}_{1,t_i} \mid \hat{X}_{1,t_i}, \text{softplus}(\xi)I_{n_{data}}\right)\mathcal{N}\left(\tilde{X}_{2,t_i} \mid \hat{X}_{2,t_i}, \text{softplus}(\xi)I_{n_{data}}\right) \\ &\mathcal{N}\left(\mathbf{w} \mid \mu_\mathbf{z}, \Sigma_\mathbf{z}\right)\mathcal{N}(\mathbf{z} \mid 0, I_d)\mathcal{N}(\xi \mid 0, 1), \end{aligned} \tag{37}$$

where $\mu_\mathbf{z}$ and $\Sigma_\mathbf{z}$ are obtained from the decoder in (4).

After training the VAE, as an example, we simulated a stochastic trajectory and performed inference on it 10. Not only does our approach make inference on the stochastic dynamical system computationally tractable, but also the process of drawing posterior samples is very fast due to the fact that we only need to run forward the VAE decoder. We marginalise over the latents during HMC sampling by

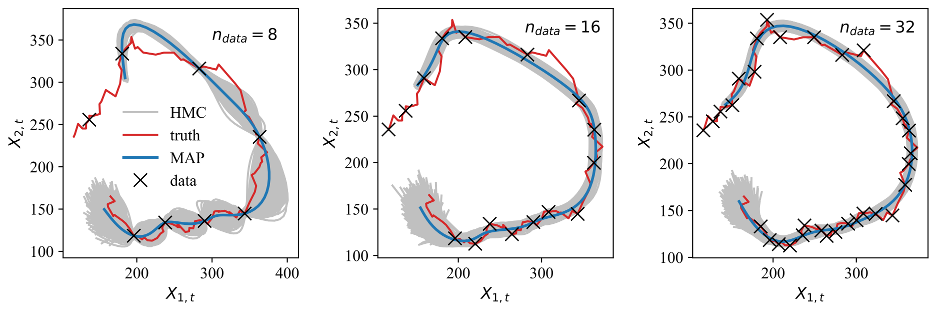

FIGURE 10. Inference on the trajectory of a stochastic dynamical system in (35). The true trajectory (red) is inferred from sparse noisy observations (crosses), using HMC. The posterior samples are shown in grey, and the MAP estimator in blue. As expected, the posterior (37) contracts around the true trajectory

simply ignoring them from the sample. Figure 10 demonstrates a contracting posterior during inference. Each panel shows inference, trained on increasing number of n_{data} sparse noisy observations from the trajectory. Notice the high uncertainty about the initial condition. This is to be expected, because the underlying dynamics of the system is non-deterministic, and a broad set of initial configuration follows very similar evolution patterns, as reflected by the posterior.

6. **Conclusion.** We proposed a Bayesian inference scheme, based on a black-box PDE solver and a VAE. In summary, the proposed method is agnostic to the numerical scheme used for solving the PDE, and produces an efficient and computationally inexpensive functional prior for Bayesian inference. We considered three examples: a non-linear boundary value problem in 1D, a linear elliptic PDE in 2D, and a stochastic differential equation in 2D. In all cases, we chose more or less arbitrarily such aspects of VAE as NN architectures and dimensionality of the latent space. When dealing with a specific problem, one can use Bayesian model selection or optimisation to optimise such choices. There are also avenues for future investigation, based on imposing a covariance structure over the latent space, which would be reflective of the underlying measure on the functional space approximated by the VAE.

Our proposed method has an accuracy bottleneck, associate with the representation of the probability distribution over functions in terms of Fourier-series expansion. Alternative methods of encoding functional distributions, such as conditional neural processes may provide better representation of the PDE solution manifold in regimes where high amounts of training data are available.

REFERENCES

[1] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes and G. N. Wells, The FEniCS Project Version 1.5, **Vol 3** , <strong>Starting Point and Frequency: </strong>Year: 2013</p>, URL http://journals.ub.uni-heidelberg.de/index.php/ans/article/view/20553.

[2] M. A. Beaumont, Approximate Bayesian Computation, **6** , 379–403, URL https://doi.org/10.1146/annurev-statistics-030718-105212.

[3] B. Calderhead, M. Girolami and N. Lawrence, Accelerating bayesian inference over nonlinear differential equations with gaussian processes, in *Advances in Neural Information Processing Systems* (eds. D. Koller, D. Schuurmans, Y. Bengio and L. Bottou), vol. 21, Curran Associates, Inc., URL https://proceedings.neurips.cc/paper/2008/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

[4] M. Girolami, Bayesian inference for differential equations, **408** , 4–16, URL https://linkinghub.elsevier.com/retrieve/pii/S030439750800501X.

[5] M. Girolami and B. Calderhead, Riemann manifold Langevin and Hamiltonian Monte Carlo methods: Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods, **73** , 123–214, URL http://doi.wiley.com/10.1111/j.1467-9868.2010.00765.x.

[6] M. Girolami, E. Febrianto, G. Yin and F. Cirak, The statistical finite element method (statFEM) for coherent synthesis of observation data and model predictions, **375** , 113533, URL https://www.sciencedirect.com/science/article/pii/S0045782520307180.

[7] A. Grover, M. Dhar and S. Ermon, Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models, in *Thirty-Second AAAI Conference on Artificial Intelligence*, URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17409.

[8] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, URL http://www.deeplearningbook.org.

[9] H. Kersting, N. Krämer, M. Schiegg, C. Daniel, M. Tiemann and P. Hennig, Differentiable Likelihoods for Fast Inversion of 'Likelihood-Free' Dynamical Systems, in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, 5198–5208, URL https://proceedings.mlr.press/v119/kersting20a.html.

[10] D. P. Kingma and M. Welling, Auto-Encoding Variational Bayes, URL http://arxiv.org/abs/1312.6114.

[11] D. P. Kingma and M. Welling, An Introduction to Variational Autoencoders, **12** , 307–392, URL http://arxiv.org/abs/1906.02691.

[12] L. F. Price, C. C. Drovandi, A. Lee and D. J. Nott, Bayesian Synthetic Likelihood, **27** , 1–11, URL https://doi.org/10.1080/10618600.2017.1302882.

[13] D. J. Tait and T. Damoulas, Variational Autoencoding of PDE Inverse Problems, URL http://arxiv.org/abs/2006.15641.

[14] L. van der Maaten and G. Hinton, Visualizing data using t-SNE, **9** , 2579–2605, URL http://jmlr.org/papers/v9/vandermaaten08a.html.

[15] L. Yang, D. Zhang and G. E. Karniadakis, Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations, **42** , A292–A317, URL https://epubs.siam.org/doi/abs/10.1137/18M1225409.

[16] Y. Yang and P. Perdikaris, Adversarial Uncertainty Quantification in Physics-Informed Neural Networks, **394** , 136–152, URL http://arxiv.org/abs/1811.04026.

[17] M. Álvarez, D. Luengo and N. D. Lawrence, Latent Force Models, in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, PMLR, 9–16, URL https://proceedings.mlr.press/v5/alvarez09a.html.

Received xxxx 20xx; revised xxxx 20xx; early access xxxx 20xx.

E-mail address: pyatsyshin@turing.ac.uk
E-mail address: a.duncan@imperial.ac.uk