Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 383 (2021) 113895
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Accelerating uncertainty quantification of groundwater flow modelling using a deep neural network proxy

Mikkel B. Lykkegaard[a,*], Tim J. Dodwell[a,b], David Moxey[a]

[a] College of Engineering, Mathematics and Physical Sciences, University of Exeter, UK
[b] The Alan Turing Institute, London, NW1 2DB, UK

Received 22 June 2020; received in revised form 21 December 2020; accepted 26 April 2021
Available online xxxx

Abstract

Quantifying the uncertainty in model parameters and output is a critical component in model-driven decision support systems for groundwater management. This paper presents a novel algorithmic approach which fuses Markov Chain Monte Carlo (MCMC) and Machine Learning methods to accelerate uncertainty quantification for groundwater flow models. We formulate the governing mathematical model as a Bayesian inverse problem, considering model parameters as a random process with an underlying probability distribution. MCMC allows us to sample from this distribution, but it comes with some limitations: it can be prohibitively expensive when dealing with costly likelihood functions, subsequent samples are often highly correlated, and the standard Metropolis–Hastings algorithm suffers from the curse of dimensionality. This paper designs a Metropolis–Hastings proposal which exploits a deep neural network (DNN) approximation of a groundwater flow model, to significantly accelerate MCMC sampling. We modify a delayed acceptance (DA) model hierarchy, whereby proposals are generated by running short subchains using an inexpensive DNN approximation, resulting in a decorrelation of subsequent fine model proposals. Using a simple adaptive error model, we estimate and correct the bias of the DNN approximation with respect to the posterior distribution on-the-fly. The approach is tested on two synthetic examples; a isotropic two-dimensional problem, and an anisotropic three-dimensional problem. The results show that the cost of uncertainty quantification can be reduced by up to 50% compared to single-level MCMC, depending on the precomputation cost and accuracy of the employed DNN.

Keywords: Groundwater flow; Uncertainty quantification; Markov chain Monte Carlo; Surrogate models; Deep neural networks

1. Introduction

Modelling of groundwater flow and transport is an important decision support tool when, for example, estimating the sustainable yield of an aquifer or remediating groundwater pollution. However, the input parameters for mathematical models of groundwater flow (such as subsurface transmissivity and boundary conditions) are often impossible to determine fully or accurately, and are hence subject to various uncertainties. In order to make informed decisions, it is of critical importance to decision makers to obtain robust and unbiased estimates of the total model

* Corresponding author.
E-mail addresses: m.lykkegaard@exeter.ac.uk (M.B. Lykkegaard), t.dodwell@exeter.ac.uk (T.J. Dodwell), d.moxey@exeter.ac.uk (D. Moxey).

https://doi.org/10.1016/j.cma.2021.113895

uncertainty, which in turn is a product of the uncertainty of these input parameters [1]. A popular way to achieve this, in relation to groundwater flow or any inverse problem in general, is stochastic or Bayesian modelling [2–4]. In this context, a probability distribution, the *prior*, is assigned to the input parameters, in accordance with any readily available information. Given some real-world measurements corresponding to the model outputs (e.g. sparse spatial measurements of hydraulic head, Darcy flow or concentration of pollutants), it is possible to reduce the overall uncertainty and obtain a better representation of the model by conditioning the prior distribution on this data. The result is a distribution of the model input parameters given data, which is also referred to as the *posterior*.

Obtaining samples from the posterior distribution directly is not possible for all but the simplest of problems. A popular approach for generating samples is the Metropolis–Hastings type *Markov Chain Monte Carlo* (MCMC) method [5]. Samples are generated by a sequential process. First, given a current sample, a new proposal for the input parameters is made using a so-called proposal distribution. Evaluating the model with this new set of parameters, a *likelihood* is computed — a measure of misfit between the model outputs and the data. The likelihoods of the proposed and current samples are then compared. Based on this comparison, the proposal is either accepted or rejected, and the whole process is repeated, generating a Markov chain of probabilistically feasible input parameters. The key point is that the distribution of samples in the chain converges to the *posterior* – the distribution of input parameters given the data [5]. This relatively simple algorithm can lead to extremely expensive Bayesian computations for three key reasons. First, each step of the chain requires the evaluation of (often) an expensive mathematical model. Second, the sequential nature of the algorithm means subsequent samples are often highly correlated — even repeated if a step is rejected. Therefore the chains must often be very long to obtain good statistics on the distribution of outputs of the model. Third, without special care, the approach does not generally scale well to large numbers of uncertain input parameters; the so-called curse of dimensionality. Addressing these scientific challenges is at the heart of modern research in MCMC algorithms. As with this paper there is a particular focus on developing novel and innovative proposal distributions, which seek to de-correlate adjacent samples and limit the computational burden of evaluating expensive models.

Broadly in the literature, simple Darcy type models and other variants of the diffusion equation have long been a popular toy example problems for demonstrating MCMC methodologies in the applied mathematics community (see e.g. [6–8]). There appears to be much less interest in MCMC in the applied groundwater modelling community. This may be because of the computational cost of running MCMC on highly parametrised, expensive models, or the lack of an easy-to-use MCMC software framework, akin to the parameter estimation toolbox PEST [9].

An exciting approach to significantly reduce the computational cost has been proposed in multi-level, multi-fidelity and Delayed Acceptance (DA) MCMC methods. In each case, to alleviate computational cost, a hierarchy of models is established, consisting of a fine model and (possibly multiple) coarse, computationally cheap approximations. Typically, the coarser models are finite element solutions of the PDE on a mesh with a coarser resolution, but as we show in this paper, can be taken to be any general approximation similar to the multi-fidelity philosophy [10]. Independent of the approach, the central idea is the same: to obtain significant efficiency gains by exploiting approximate coarse models to generate ‘good’ proposals cheaply, using additional accept/reject steps to filter out highly unlikely proposals before evaluating the fine, expensive model. Previous studies of two-stage approaches include [11] who modelled multi-phase flow with coarse level proposals evaluated by a coarse-mesh single-phase flow model (an idea that was developed further in [12]), [13] and [14]. We note that the latter of which, instead of simply using a coarser discretisation, implemented a data-driven polynomial chaos expansion as a surrogate model. We intend to demonstrate how the development of novel techniques in MCMC and machine learning can be combined to help realise the potential of MCMC in this field.

In this work, we propose a combination of multiple cutting-edge MCMC techniques to allow for efficient inversion and uncertainty quantification of groundwater flow. We propose an improved delayed acceptance (DA) MCMC algorithm, adapted from the approach proposed by [15]. In our case, similarly to multi-level MCMC [7], proposals are generated by computing a subchain using a Deep Neural Network (DNN) as an approximate model — leading to cheaply computed, decorrelated proposals passed on to the fine model. For our first example, the subchain is driven by the preconditioned Crank–Nicolson (pCN) proposal distribution [16] to ensure the proposed Metropolis–Hastings algorithm is robust with respect to the dimension of the uncertain parameter space. For our second example, proposals for the subchains are generated using the Adaptive Metropolis (AM) proposal [17], since the posterior distribution in this case is highly non-spherical and multiple parameters are correlated. Finally, we propose an enhanced error model, in which the DNN is trained by sampling the prior distribution, yet the bias of the approximation is adaptively estimated and corrected on-the-fly by testing the approximations against the full model in an adaptive delayed acceptance setting [18].

2. Preliminaries

In this section we briefly introduce the forward model, defining the governing equations underpinning groundwater flow and their corresponding weak form, enabling us to solve the equations using FEM methods. We then formulate our model as a Bayesian inverse problem with random input parameters, effectively resulting in a stochastic model, which can be accurately characterised by sampling from the posterior distribution of parameters using MCMC. The simple Metropolis–Hastings MCMC algorithm is then introduced and extended with the preconditioned Crank–Nicolson (pCN) and Adaptive Metropolis (AM) transition kernels.

2.1. Governing equations for groundwater flow

Consider steady groundwater flow in a confined, inhomogeneous aquifer which occupies the domain Ω with boundary Γ. Assuming that water is incompressible, the governing equations for groundwater flow can be written as the scalar elliptic partial differential equation:

$$-\nabla \cdot (-T(\mathbf{x})\nabla h(\mathbf{x})) = g(\mathbf{x}) \quad \text{for all} \quad \mathbf{x} \in \Omega \tag{1}$$

subject to boundary conditions on $\Gamma = \Gamma_N \cup \Gamma_D$ defined by the constraint equations

$$h(\mathbf{x}) = h_D(\mathbf{x}) \quad \text{on } \Gamma_D \quad \text{and} \quad (-T(\mathbf{x})\nabla h(\mathbf{x})) \cdot \boldsymbol{n} = q_N(\mathbf{x}) \quad \text{on } \Gamma_N. \tag{2}$$

Here $T(\mathbf{x})$ is the heterogeneous, depth-integrated transmissivity, $h(\mathbf{x})$ is hydraulic head, $h_D(\mathbf{x})$ is fixed hydraulic head at boundaries with Dirichlet constraints, $g(\mathbf{x})$ is fluid sources and sinks, $q(\mathbf{x})$ is Darcy velocity, $q_N(\mathbf{x})$ is Darcy velocity across boundaries with Neumann constraints and $\Gamma_D \subset \partial\Omega$ and $\Gamma_N \subset \partial\Omega$ define the boundaries comprising of Dirichlet and Neumann conditions, respectively. Following standard FEM practice (see e.g. [19]), Eq. (1) is converted into weak form by multiplying by an appropriate test function $w \in H^1(\Omega)$ and integrating by parts, so that

$$\int_\Omega \nabla w \cdot (T(\mathbf{x})\nabla h)\, d\mathbf{x} + \int_{\Gamma_N} w\, q_N(\mathbf{x})\, ds = \int_\Omega w\, g(\mathbf{x})\, d\mathbf{x}, \quad \forall w \in H^1(\Omega), \tag{3}$$

where $H^1(\Omega)$ is the Hilbert space of weakly differentiable functions on Ω. To approximate the hydraulic head solution $h(\boldsymbol{x})$, a finite element space $V_\tau \subset H^1(\Omega)$ on a finite element mesh $\mathcal{Q}_\tau(\Omega)$. This is defined by a basis of piecewise linear Lagrange polynomials $\{\phi_i(\mathbf{x})\}_{i=1}^M$, associated with each of the M finite element nodes. As a result (3) can be rewritten as a system of sparse linear equations

$$\mathbf{A}\mathbf{h} = \mathbf{b} \quad \text{where} \quad A_{ij} = \int_\Omega \nabla\phi_i \cdot T(\mathbf{x})\nabla\phi_j(\mathbf{x})\, d\mathbf{x} \quad \text{and} \tag{4}$$

$$b_i = \int_\Omega \phi_i(\mathbf{x})\, g(\mathbf{x})\, d\mathbf{x} - \int_{\Gamma_N} \phi_i(\mathbf{x}) q_N(\mathbf{x})\, ds, \tag{5}$$

where $\mathbf{A} \in \mathbb{R}^{M\times M}$ and $\mathbf{b} \in \mathbb{R}^M$ are the global stiffness matrix and load vector, respectively. The vector $\mathbf{h} \coloneqq [h_1, h_2, \ldots, h_M] \in \mathbb{R}^M$ is the solution vector of hydraulic head at each node within the finite element mesh so that $h(\mathbf{x}) = \sum_{i=1}^M h_i\phi_i(\mathbf{x})$. In our numerical experiments, these equations are solved using the open source general-purpose FEM framework `FEniCS` [20]. While there are well-established groundwater simulation software packages available, such as MODFLOW [21] and FEFLOW [19], FEniCS was chosen because of its flexibility and ease of integration with other software and analysis codes.

2.2. Aquifer transmissivity

The aquifer transmissivity $T(\mathbf{x})$ is not known everywhere on the domain, therefore a typical approach is to model it as a log-Gaussian random field. There exists extensive literature on modelling groundwater flow transmissivity using log-Gaussian random fields (see e.g. [22,23,14]). Whilst this may not always prove a good model, particularly in cases with highly correlated extreme values and/or preferential flow paths [24,25] as seen when considering faults and other discontinuities [26,27], the log-Gaussian distribution remains relevant for modelling transmissivity in a range of aquifers [28,29,14].

Our starting point is a covariance operator with kernel $C(\mathbf{x}, \mathbf{y})$, which defines the correlation structure of the uncertain transmissivity field. For our numerical experiments, we consider the ARD (Automatic Relevance Determination) squared exponential kernel, a generalisation of the 'classic' squared exponential kernel, which allows for handling directional anisotropy:

$$C(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2}\sum_{j=1}^{d}\left(\frac{x_j - y_j}{l_j}\right)^2\right), \tag{6}$$

where d is the spatial dimensionality of the problem and $\boldsymbol{l} \in \mathbb{R}^d$ is a vector of lengths scales corresponding to each spatial dimension. We emphasise that the covariance kernel is a *modelling choice*, and that different options are available, such as the Matern kernel which offers additional control over the smoothness of the field.

In our work, transmissivity was modelled as a discrete log-Gaussian random field expanded in an orthogonal eigenbasis with k Karhunen–Loève (KL) eigenmodes. To achieve this we construct a covariance matrix $\mathbf{C} \in \mathbb{R}^{M\times M}$, where entries are given by $C_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$ for each pair of nodal coordinates within the finite element mesh $i, j = 1, \ldots, M$. Once constructed, the largest k eigenvalues $\{\lambda_i\}_{i=1}^k$ and associated eigenvectors $\{\boldsymbol{\psi}_i\}_{i=1}^k$ of $\mathbf{C}$ can be computed. The transmissivity at the nodes $\mathbf{t} := [t_1, t_2, \ldots, t_M]$, is given by

$$\log \boldsymbol{t} = \boldsymbol{\mu} + \sigma\, \boldsymbol{\Psi}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{\theta}, \quad \text{where} \quad \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_k \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Psi} = [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_k], \tag{7}$$

where $\boldsymbol{\mu}$ defines the log of the mean transmissivity field, σ is a scalar parametrising the variance and $\boldsymbol{\theta}$ is a vector of Gaussian random variables such that $\boldsymbol{\theta} \sim \mathcal{N}(0, \mathbb{I}_k)$ as in [30]. The random field can be interpolated from nodal values across Ω, using the shape functions $\{\phi_i(\mathbf{x})\}_{i=1}^M$ so that $T(\mathbf{x}) = \sum_{i=1}^M t_i\phi_i(\mathbf{x})$.

Truncating the KL eigenmodes at the kth mode limits the amount of small scale features that can be represented. This, along with interpolating the field, has a smoothing effect on the recovered transmissivity fields, which may or may not be desirable, depending on the application. Fig. 1 shows some examples of realisations of Gaussian random fields with a square exponential kernel, which illustrates the effect of the covariance length scale l and the number of admitted KL eigenmodes k. For relatively large length scales l, there is a limit to k, above which adding higher frequency eigenvalues does not provide any additional information. In this context, the proportion of signal energy encompassed by the truncation can be understood as the ratio between the sum of truncated eigenvalues and the sum of all eigenvalues: $\sum_{i=1}^k \lambda_i / \sum_{j=1}^M \lambda_j$.

2.3. The Bayesian inverse problem

To setup the Bayesian inverse problem and thereby quantify the uncertainty in the transmissivity field $T(\mathbf{x})$, the starting point is to define a statistical model which describes distribution of the mismatch between observations and model predictions. The observations are expressed in a single vector $\mathbf{d}_{obs} \in \mathbb{R}^m$ and for a given set of model input parameters $\boldsymbol{\theta}$, the model's prediction of the data is defined by the *forward map*, $\mathcal{F}(\boldsymbol{\theta}) : \mathbb{R}^k \to \mathbb{R}^m$. The statistical model assumes the connection between model and observations through the relationship

$$\boldsymbol{d}_{\text{obs}} = \mathcal{F}(\boldsymbol{\theta}) + \boldsymbol{\epsilon} \tag{8}$$

where we take $\boldsymbol{\epsilon} \sim N(0, \boldsymbol{\Sigma}_\epsilon)$ which represents the uncertainty of the connection between model and data, capturing both model mis-specification and measurement noise as sources of this uncertainty.

The backbone of a Bayesian approach is Bayes' theorem, which allows for computing *posterior* beliefs of model parameters using both *prior* beliefs and *observations*. Bayes' theorem states that the posterior probability of a parameter realisation $\boldsymbol{\theta}$ given data $\boldsymbol{d}_{\text{obs}}$ can be computed as

$$\pi(\boldsymbol{\theta}|\boldsymbol{d}_{\text{obs}}) = \frac{\pi_0(\boldsymbol{\theta})\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta})}{\pi(\boldsymbol{d}_{\text{obs}})} \tag{9}$$

where $\pi(\boldsymbol{\theta}|\boldsymbol{d}_{\text{obs}})$ is referred to as the *posterior distribution*, $\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta})$ is called the *likelihood*, $\pi_0(\boldsymbol{\theta})$ the *prior distribution* and

$$\pi(\boldsymbol{d}_{\text{obs}}) = \int_\Theta \pi(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta})d\boldsymbol{\theta} = \int_\Theta \pi_0(\boldsymbol{\theta})\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta})d\boldsymbol{\theta} \tag{10}$$

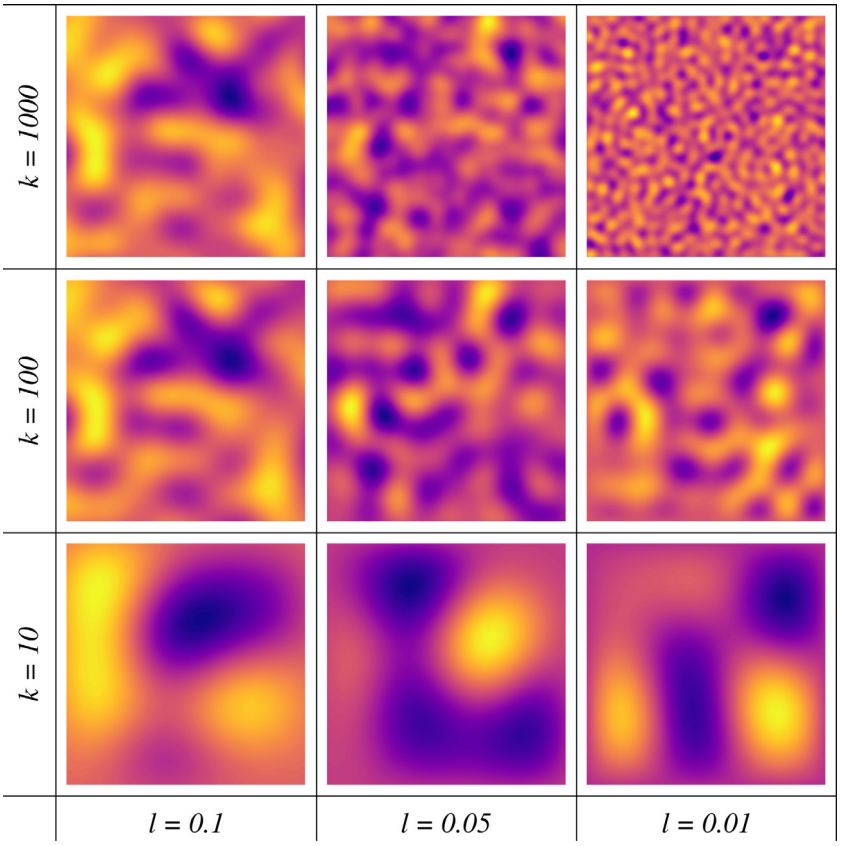


Fig. 1. A selection of Gaussian random process realisations for $\boldsymbol{x} \in [0,1]^2$, with a square exponential kernel using different covariance length scales l and number of KL eigenmodes k. All displayed realisations were generated using the same appropriately truncated random vector $\boldsymbol{\xi}$ with identical eigenvectors for each l.

is a normalising constant, sometimes referred to as the *evidence*. In most cases this integral does not have a closed-form solution and is infeasible to estimate numerically in most real-world applications, particularly when the dimension of the unknown parameter space is large and the evaluation of the model (required to compute $\mathcal{L}(\mathbf{d}_{obs}|\boldsymbol{\theta})$) is computationally expensive. A family of methods called Markov Chain Monte Carlo (MCMC) are often employed to approximate the solution [31]. Importantly MCMC, whilst computationally expensive, allows indirect sampling from the posterior distribution and avoids the explicit need to estimate (10). Moreover, it can be designed to be independent of the dimension of the parameter space and has no embedded unquantifiable bias. In this paper we consider a subclass of MCMC methods called the Metropolis–Hastings [32,33,5] algorithm, which is described in Algorithm 1. The algorithm generates a Markov chain $\{\boldsymbol{\theta}^{(n)}\}_{n\in\mathbb{N}}$ with a distribution converging to $\pi(\boldsymbol{d}_{obs}|\boldsymbol{\theta})$. It is difficult (often impossible) to sample directly from the posterior, hence at each step, at position $\boldsymbol{\theta}^{(i)}$ in the chain, a proposal is made $\boldsymbol{\theta}'$ from a simpler known (proposal) distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i)})$. An accept/reject step then determines whether the proposal comes from (probabilistically) the posterior distribution or not. This accept/reject step is a achieved by essentially computing the ratio of the densities of the current state to the proposal. To do this we exploit Bayes's Theorem. The key observation in MCMC is that the normalising constant $\pi(\boldsymbol{d}_{\text{obs}})$ is independent of $\boldsymbol{\theta}$, and so

$$\pi(\boldsymbol{\theta}|\boldsymbol{d}_{\text{obs}}) \propto \pi_0(\boldsymbol{\theta})\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}). \tag{11}$$

Therefore when comparing the ratio of the densities, the normalising constant (since independent of $\boldsymbol{\theta}$) cancels.

Algorithm 1: Metropolis–Hastings Algorithm

1. Given a parameter realisation $\boldsymbol{\theta}_i$ and a transition kernel $q(\boldsymbol{\theta}'|\boldsymbol{\theta}_i)$, generate a proposal $\boldsymbol{\theta}'$.
2. Compute the likelihood ratio between the proposal and the previous realisation:

$$\alpha = \min\left\{1, \frac{\pi_0(\boldsymbol{\theta}')\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}')}{\pi_0(\boldsymbol{\theta}^{(i)})\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}^{(i)})}\frac{q(\boldsymbol{\theta}^{(i)}|\boldsymbol{\theta}')}{q(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i)})}\right\}$$

3. If $u \sim U(0,1) > \alpha$ then set $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$, otherwise, set $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}'$.

In our model problem, the prior density of the parameters $\pi_0(\boldsymbol{\theta})$ represents the available *a priori* knowledge about the transmissivity of the aquifer. From our statistical model (8) we see that our $\boldsymbol{d}_{obs} - \mathcal{F}(\boldsymbol{\theta}) \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\epsilon)$, hence

$$\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}) = \exp\left(-\frac{1}{2}(\mathcal{F}(\boldsymbol{\theta}) - \boldsymbol{d}_{\text{obs}})^\intercal \boldsymbol{\Sigma}_e^{-1}(\mathcal{F}(\boldsymbol{\theta}) - \boldsymbol{d}_{\text{obs}})\right). \tag{12}$$

Importantly we note that for each step of the Metropolis–Hastings algorithms we are required to compute $\mathcal{L}(\boldsymbol{d}_{obs}|\boldsymbol{\theta}')$. This requires the evaluation of the forward mapping $\mathcal{F}(\boldsymbol{\theta}')$ which can be computationally expensive. Moreover, due to the sequential nature of MCMC-based approaches, consecutive samples are correlated and hence many samples are required to obtain good statistics on the outputs.

The proposal distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(n)})$ is the key element which drives the Metropolis–Hastings algorithm and control the effectiveness of the algorithm. A common choice is a simple random walk, for which $q_{\text{RW}}(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i)}) = \mathcal{N}(\boldsymbol{\theta}^{(i)}, \boldsymbol{\Sigma})$, yet as shown in [34], the basic random walk does not lead to a convergence that is independent of the input dimension m. Better choices would be the *preconditioned Crank–Nicolson* proposal (pCN, [16]), which has dimension independent acceptance probability, or the *Adaptive Metropolis* algorithm (AM, [17]), which adaptively aligns the proposal distribution to the posterior during sampling. Moreover, unlike the Metropolis-Adjusted Langevin Algorithm (MALA), No-U-Turn Sampler (NUTS) and Hamiltonian Monte Carlo, none of these proposals rely on gradient information, which can be infeasible to compute for expensive forward models.

To generate a proposal using the pCN transition kernel, one computes

$$\boldsymbol{\theta}' = \sqrt{1-\beta^2}\,\boldsymbol{\theta}^{(i)} + \beta\boldsymbol{\xi} \tag{13}$$

where $\boldsymbol{\xi}$ is a random sample from the prior distribution, $\boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$. This expression corresponds to the transition kernel $q_{\text{pCN}}(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i)}) = \mathcal{N}(\sqrt{1-\beta^2}\boldsymbol{\theta}^{(i)}, \beta^2\boldsymbol{\Sigma})$. Moreover, for the pCN transition kernel, the acceptance probability simplifies to

$$\alpha = \min\left\{1, \frac{\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}')}{\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}^{(i)})}\right\} \quad \text{following the identity} \quad \frac{p_0(\boldsymbol{\theta}^{(i)})}{p_0(\boldsymbol{\theta}')} = \frac{q_{\text{pCN}}(\boldsymbol{\theta}^{(i)}|\boldsymbol{\theta}')}{q_{\text{pCN}}(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i)})} \tag{14}$$

as given in [7]. Additional details of derivation of the pCN proposal are provided in Appendix A.

Similarly, to generate a proposal using the AM transition kernel, we draw a random sample

$$\boldsymbol{\theta}' \sim \mathcal{N}(\boldsymbol{\theta}^{(i)}, \boldsymbol{\Sigma}^{(i)}) \tag{15}$$

where $\boldsymbol{\Sigma}^{(i)}$ is an iteratively updated covariance structure

$$\boldsymbol{\Sigma}^{(i)} = \begin{cases} \boldsymbol{\Sigma}^{(0)}, & \text{if } i \leq i_0, \\ s_d\,\text{Cov}(\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)} \ldots \boldsymbol{\theta}^{(i)}) + s_d\,\gamma\,\mathbb{I}_d, & \text{otherwise.} \end{cases}$$

Hence, proposals are drawn from a distribution with an initial covariance $\boldsymbol{\Sigma}^{(0)}$ for a given period i_0, after which adaptivity is 'switched on', and used for the remaining samples. The adaptive covariance $\boldsymbol{\Sigma}^{(i)} = s_d\,\text{Cov}(\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)} \ldots \boldsymbol{\theta}^{(i)}) + s_d\,\gamma\,\mathbb{I}_d$ can be constructed iteratively during sampling using the following recursive formula:

$$\boldsymbol{\Sigma}^{(i+1)} = \frac{i-1}{i}\boldsymbol{\Sigma}^{(i)} + \frac{s_d}{i}(i\bar{\boldsymbol{\theta}}^{(i-1)}\bar{\boldsymbol{\theta}}^{(i-1)\intercal} - (i+1)\bar{\boldsymbol{\theta}}^{(i)}\bar{\boldsymbol{\theta}}^{(i)\intercal} + \boldsymbol{\theta}^{(i)}\boldsymbol{\theta}^{(i)\intercal} + \gamma\mathbb{I}_d) \tag{16}$$

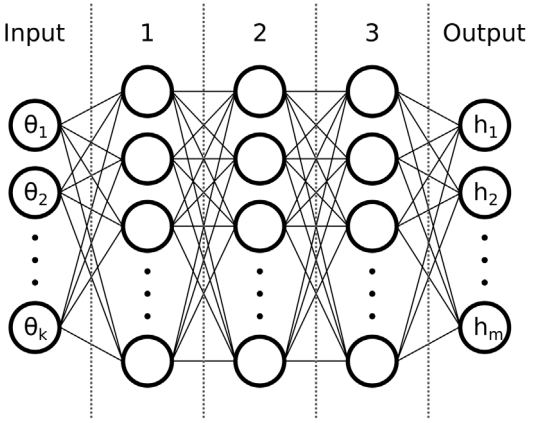


Fig. 2. Graph showing the structure of a feedforward DNN.

where $\bar{\cdot}$ is the arithmetic mean, $s_d = 2.4^2/d$ is a scaling parameter, d is the dimension of the proposal distribution and γ is a parameter which prevents $\boldsymbol{\Sigma}_i$ from becoming singular [17]. This, on the other hand, corresponds to the transition kernel $q_{\mathrm{AM}}(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)} \ \ldots \ \boldsymbol{\theta}^{(i)}) = \mathcal{N}(\boldsymbol{\theta}^{(i)}, \boldsymbol{\Sigma}^{(i)})$, which is not guaranteed to be ergodic, since it will depend on the history of the chain. However, the Diminishing Adaptation condition [35] holds, as adaptation will naturally decrease as sampling progresses.

2.4. Deep neural network

The approximate/surrogate model in our experiments is a feed-forward deep neural network (DNN), a type of artificial neural network with multiple hidden layers, as implemented in the open-source neural-network library `Keras` [36] utilising the `Theano` backend [37].

Artificial neural networks have previously been successfully applied as fast model proxies in inverse geophysics problems. Examples include [38], who used a neural network with two hidden layers for Monte Carlo sampling in the context of a crosshole traveltime inversion, and [39] who used a neural network with a single hidden layer and a Differential Evolution Adaptive Metropolis sampler for electromagnetic inversion.

The DNN approximates the forward map, accepting a vector of KL coefficients $\boldsymbol{\theta} \in \mathbb{R}^k$, and returning an approximation of the vector of approximate model output $\hat{\mathcal{F}}(\boldsymbol{\theta}) \in \mathbb{R}^m$ – in this paper a vector of hydraulic heads at given sampling points, i.e. $\hat{\mathcal{F}}(\boldsymbol{\theta}) : \mathbb{R}^k \mapsto \mathbb{R}^m$. Fig. 2 shows the graph of one particular DNN employed in our experiments.

Each edge in Fig. 2 is equipped with a weight $w^l_{i,j}$ where l is index of the layer that the weight feeds into, i is the index of nodes in the same layer and j is the index of nodes in the previous layer. These weights can be arranged in $n \times m$ matrices $\boldsymbol{W}_l$ for each layer l. Similarly, each node is equipped with a bias b^l_i where l is index of its layer and i is the index of node, and these biases can be arranged in vectors $\boldsymbol{b}_l$. Data is propagated through the network such that the output $\boldsymbol{y}_l$ of a layer l with activation function $\mathcal{A}_l(\cdot)$ is

$$\boldsymbol{y}_l = \mathcal{A}_l\left(\boldsymbol{b}_l + \boldsymbol{W}_l \ \boldsymbol{y}_{l-1}\right). \tag{17}$$

Activation functions $A(\cdot)$ are applied element-wise on their input vectors $\boldsymbol{x}$ so that

$$\mathcal{A}(\boldsymbol{x}) = (A(x_1), \ A(x_2) \ \ldots \ A(x_n))^{\mathsf{T}}$$

Many different activation functions are available for artificial neural networks, and we here give a short description of the ones employed in our experiments: the *sigmoid* and the *rectified linear unit* ('ReLU'). The transfer function of the nodes in the first layer of each DNN was of the type *sigmoid*:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{18}$$

squashing the input vector into the interval $(0, 1)$, effectively resulting in a strictly positive output from the first hidden layer. The remaining hidden layers consisted of nodes with the *de facto* standard hidden layer activation function for deep neural networks, the *rectified linear unit* ('ReLU'):

$$R(x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise.} \end{cases}$$

To fit an artificial neural network to a given set of data, the network is initially compiled using random weights and biases and then trained using a dataset of known inputs and their corresponding outputs. The weights and biases are updated iteratively during training by way of an appropriate optimisation algorithm and a loss function, and if appropriately set up, will converge towards a set of optimal values, allowing the DNN to predict the response of the forward model to some level of accuracy [40]. Our particular DNNs were trained using the mean squared error (MSE) loss function

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(h_i - \hat{h}_i)^2$$

for m output variables, and the RMSprop optimiser, a stochastic, gradient based and adaptive algorithm, suggested by [41] and widely used for training DNNs.

3. Adaptive delayed acceptance proposal using a deep neural network

In this section we describe a modified adaptive delayed acceptance proposal for MCMC, using ideas from multi-level MCMC [7]. The general approach generates proposals by running Markov subchains driven by an approximate model. In our case this approximation is constructed from a DNN of the forward map $\mathcal{F}(\boldsymbol{\theta})$ trained from offline samples of the prior distribution. Finally, we show how the approximate map can be corrected online, by adaptively learning a simple multi-variant Gaussian correction to the outputs of the neural network.

3.1. Modified delayed acceptance MCMC

Delayed Acceptance (DA) [15] is a technique that exploits a model hierarchy consisting of an expensive fine model and relatively inexpensive coarse approximation. The idea is simple: a proposal is first evaluated (pre-screened) by an approximate model and immediately discarded if it is rejected. Only if accepted, it is subjected to a second accept/reject step using the fine model. In this context, the likelihood of observations given a parameter set is henceforth denoted $\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta})$ when evaluated on the approximate model and remains $\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta})$ when evaluated on the fine model. This simple screening mechanism cheaply filters out poor proposals, wasting minimal time evaluating unlikely proposals on the expensive, fine model. Crucially, the coarse model need not evaluate every parameter, only a subset. The remaining fine parameters can then be sampled prior to the second accept/reject step. We denote the full parameter set $\boldsymbol{\theta}$, the coarse parameters $\hat{\boldsymbol{\theta}}$ and the fine parameters $\tilde{\boldsymbol{\theta}}$. so that $\boldsymbol{\theta} = [\hat{\boldsymbol{\theta}}, \tilde{\boldsymbol{\theta}}]$.

In this paper we extend this approach by not evaluating *every* accepted approximation proposal with the fine model. Instead, a proposal for the fine model is generated by running an approximate subchain until t approximate proposals have been accepted and only then evaluate using the fine model. We define the required number of accepted proposals in the approximate subchains as the *offset length*. This modified Delayed Acceptance MCMC algorithm is described in Algorithm 2 and an illustration of the process is given in Fig. 3.

This way, the autocorrelation of the fine chain is reduced, since proposals are 'more independent'. This approach is strongly related to a two-level version of multi-level MCMC. Since the fine model likelihood ratio is corrected by the inverse of the approximate likelihood ratio in step 6 of Algorithm 2, detailed balance is satisfied, the resulting Markov Chain is guaranteed to come from the true posterior and there is no loss of accuracy, even if the approximate model is severely wrong [15]. To demonstrate that this approach does indeed decrease the autocorrelation in our fine chain MCMC samples, we compute the Effective Sample Size N_{eff} of each MCMC simulation according to the procedures described in [42].

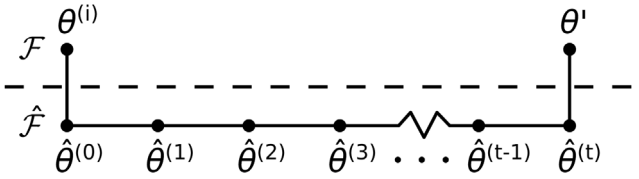


Fig. 3. Illustration of the principle used to offset fine level samples to reduce autocorrelation. The fine model $\mathcal{F}$ is only evaluated using the full set of proposed parameters $\boldsymbol{\theta}'$ after a prescribed number t (the *offset length*) of approximation parameter sets $\{\hat{\boldsymbol{\theta}}^{(1)}, \hat{\boldsymbol{\theta}}^{(2)}, \ldots, \hat{\boldsymbol{\theta}}^{(t)}\}$ have been evaluated on the approximate model $\hat{\mathcal{F}}$ and accepted into the coarse chain.

Algorithm 2: Modified Delayed Acceptance MCMC

1. Given a realisation of the approximation parameters $\hat{\boldsymbol{\theta}}^{(j)}$ and the transition kernel $q(\hat{\boldsymbol{\theta}}'|\hat{\boldsymbol{\theta}}^{(j)})$, generate a proposal for the approximation $\hat{\boldsymbol{\theta}}'$.
2. Compute the likelihood ratio on the approximate model between the proposal and the previous realisation:

$$\alpha_1 = \min\left\{1, \frac{\pi_0(\hat{\boldsymbol{\theta}}')\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\hat{\boldsymbol{\theta}}')}{\pi_0(\hat{\boldsymbol{\theta}}^{(j)})\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\hat{\boldsymbol{\theta}}^{(j)})}\right\} \qquad (AM)$$

$$\alpha_1 = \min\left\{1, \frac{\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\hat{\boldsymbol{\theta}}')}{\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\hat{\boldsymbol{\theta}}^{(j)})}\right\} \qquad (pCN)$$

3. If $u \sim U(0,1) > \alpha_1$ then set $\hat{\boldsymbol{\theta}}^{(j+1)} = \hat{\boldsymbol{\theta}}^{(j)}$ and return to (1); otherwise set $\hat{\boldsymbol{\theta}}^{(j+1)} = \hat{\boldsymbol{\theta}}'$ and continue to (4).
4. If t proposals have been accepted in the approximation subchain, continue to (5), otherwise return to (1).
5. Given the latest realisation of the entire parameter set $\boldsymbol{\theta}^{(i)} = [\hat{\boldsymbol{\theta}}^{(i)}, \tilde{\boldsymbol{\theta}}^{(i)}]$ with fine parameters $\tilde{\boldsymbol{\theta}}^{(i)}$ and the transition kernel $q(\tilde{\boldsymbol{\theta}}'|\tilde{\boldsymbol{\theta}}^{(i)})$, generate a proposal for the fine parameters $\tilde{\boldsymbol{\theta}}'$ and set $\boldsymbol{\theta}' := [\hat{\boldsymbol{\theta}}', \tilde{\boldsymbol{\theta}}']$.
6. Compute the likelihood ratio on the fine model between the proposal and the previous realisation:

$$\alpha_2 = \min\left\{1, \frac{\pi_0(\boldsymbol{\theta}')\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}')}{\pi_0(\boldsymbol{\theta}^{(i)})\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}^{(i)})}\frac{\pi_0(\hat{\boldsymbol{\theta}}^{(i)})\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\hat{\boldsymbol{\theta}}^{(i)})}{\pi_0(\hat{\boldsymbol{\theta}}')\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\hat{\boldsymbol{\theta}}')}\right\} \qquad (AM)$$

$$\alpha_2 = \min\left\{1, \frac{\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}')}{\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}^{(i)})}\frac{\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\hat{\boldsymbol{\theta}}^{(i)})}{\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\hat{\boldsymbol{\theta}}')}\right\} \qquad (pCN)$$

7. If $u \sim U(0,1) > \alpha_2$ then set $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$, otherwise set $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}'$.

3.2. Adaptive correction of the approximate posterior

Whilst in theory the modified delayed acceptance proposal described in Section 3.1 will provide a convergent Metropolis–Hastings algorithm, there are cases in which the rate of convergence will be extremely slow. To demonstrate this, the left-hand contour plot in Fig. 4 shows an artificially bad example. In this case the approximate model (red isolines) poorly captures the target likelihood distribution (blue density); there is a clear offset in the distributions, and the scale, shape and orientation of the approximate likelihood is incorrect. If using the modified delayed acceptance algorithm without alteration, it is easy to see that the proposal mechanism would struggle to traverse the whole of the target distribution, since much of it lies in the tails of the approximate likelihood

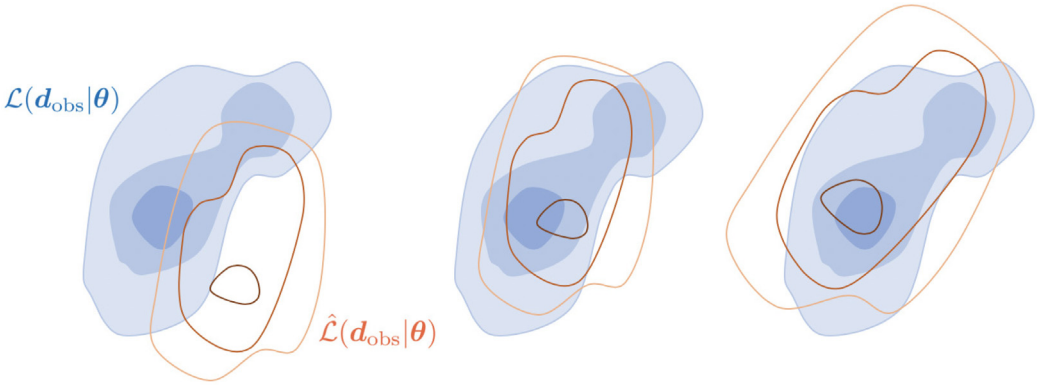


Fig. 4. Fine/target likelihood (blue) and approximate likelihood (red). (Left) Original likelihood before correction, (middle) corrected likelihood by a constant shift $\boldsymbol{\mu}_{\text{bias}}$ and (right) corrected approximate likelihood by multivariate Gaussian. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

distribution. As a result, in practice, we would observe extremely slow convergence to the true posterior; in practice – at finite computational times – results would contain a significant bias.

An ad hoc way to overcome this is to apply so-called *tempering* on the statistical model which drives the subchain. In this technique, the variance of the misfit $\boldsymbol{\Sigma}_\epsilon$ on the subchain is artificially inflated to capture the uncertainty in the approximate model. The issue in adopting this approach is the difficulty in selecting a robust inflation factor for tempering, particularly in higher dimensions. Furthermore, an isotropic inflation of the approximate posterior will in general be sub-optimal.

In this paper we instead implement an adaptive enhanced error model (EEM), which overcomes many of these challenges. Moreover, it is easy to implement and has negligible additional computational cost. Let $\hat{\mathcal{F}}$ denote the approximate forward map of the fine/target model $\mathcal{F}$. Then, following [43,18], we apply a trick to the statistical model (8) where we add and subtract the coarse map $\hat{\mathcal{F}}$. With some rearrangement we obtain the expression

$$\boldsymbol{d}_{\text{obs}} = \mathcal{F}(\boldsymbol{\theta}) + \boldsymbol{\epsilon} = \mathcal{F}(\boldsymbol{\theta}) + \hat{\mathcal{F}}(\boldsymbol{\theta}) - \hat{\mathcal{F}}(\boldsymbol{\theta}) + \boldsymbol{\epsilon} = \hat{\mathcal{F}}(\boldsymbol{\theta}) + \underbrace{\left(\mathcal{F}(\boldsymbol{\theta}) - \hat{\mathcal{F}}(\boldsymbol{\theta})\right)}_{:=\mathcal{B}(\boldsymbol{\theta})} + \boldsymbol{\epsilon}. \tag{19}$$

Here $\mathcal{B}(\boldsymbol{\theta}) = \mathcal{F}(\boldsymbol{\theta}) - \hat{\mathcal{F}}(\boldsymbol{\theta})$ is the bias associated with the approximation at given parameter values $\boldsymbol{\theta}$. We approximate this bias using a multivariate Gaussian distribution, i.e. $\mathcal{B} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{bias}}, \boldsymbol{\Sigma}_{\text{bias}})$, and therefore the likelihood function (12) can be rewritten as

$$\hat{\mathcal{L}}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta}) = \exp\left(-\frac{1}{2}(\hat{\mathcal{F}}(\boldsymbol{\theta}) + \boldsymbol{\mu}_{\text{bias}} - \boldsymbol{d}_{\text{obs}})^\intercal(\boldsymbol{\Sigma}_{\text{bias}} + \boldsymbol{\Sigma}_e)^{-1}(\hat{\mathcal{F}}(\boldsymbol{\theta}) + \boldsymbol{\mu}_{\text{bias}} - \boldsymbol{d}_{\text{obs}})\right). \tag{20}$$

The influence of redefining the likelihood is best demonstrated geometrically, as shown in Fig. 4 (middle and right). Firstly, as shown in Fig. 4 (middle) we can make a better approximation by simply adding a shift of the mean bias $\boldsymbol{\mu}_{\text{bias}}$ to the original approximate model $\hat{\mathcal{F}}(\boldsymbol{\theta})$. This has the effect of aligning the 'centre of mass' of each of the distributions. Secondly, we can learn the covariance structure of the bias. This has the effect of stretching and rotating the approximate distribution to give an even better overall approximation, as shown in Fig. 4 (right). The final mismatch between the approximate and target distribution will be driven by the assumption that bias can be represented by a multivariate Gaussian, although more complex distributions could be constructed using, for example, Gaussian process regression. Whilst this is an avenue to explore in the future, any such approach would surrender the simplicity of this approach, which from the results appears particularly effective.

The idea of using an EEM when dealing with model hierarchies originates from [43], who suggested to use samples from the prior distribution of parameters to construct the EEM prior to Bayesian inversion, so that

$$\boldsymbol{\mu}_{\text{bias}} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{B}(\boldsymbol{\theta}^{(i)}) \quad \text{and} \quad \boldsymbol{\Sigma}_{\text{bias}} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathcal{B}(\boldsymbol{\theta}^{(i)}) - \boldsymbol{\mu}_{\text{bias}})(\mathcal{B}(\boldsymbol{\theta}^{(i)}) - \boldsymbol{\mu}_{\text{bias}})^\intercal \tag{21}$$

The estimates for $\boldsymbol{\mu}_{\text{bias}}$ and $\boldsymbol{\Sigma}_{\text{bias}}$ could be obtained by sampling the prior distribution and comparing the approximate forward map against the target forward map. This approach has previously been successfully applied to a geophysical inverse problem by [44], who compared the modelling error for a large number of crosshole tomography models. However, since the model output generated by parameter sets drawn from the prior distribution may be biased significantly differently than samples drawn from a (relatively concentrated) posterior distribution, this approach may lead to an EEM that poorly represents the model bias associated with the posterior. If the approximate model is a good approximation *on average*, constructing the EEM from the prior distribution would lead to an underestimation of the mean and an overestimation of the covariance of the bias, compared to an EEM constructed from the posterior. Furthermore, in our example where the approximate model is built from samples from the prior, it is expected that such an approach would further underestimate both the mean *and* covariance of the bias, since the neural network has been explicitly trained to minimise the error with respect to samples from the prior.

Instead of estimating the bias using the prior, the posterior bias can be constructed on-line by iteratively updating its mean $\boldsymbol{\mu}_{\text{bias}}$ and covariance $\boldsymbol{\Sigma}_{\text{bias}}$ using coarse/fine solution pairs from the MCMC samples as suggested by [45]. Another similar approach was employed to a Bayesian geophysical problem by [46], who collected model bias estimates while sampling, and used the bias estimates of the k-nearest-neighbours of each new coarse sample to construct a bias. In this case we select

$$\boldsymbol{\mu}_{\text{bias},i+1} = \frac{1}{i+1}\left(i\,\boldsymbol{\mu}_{\text{bias},i} + \mathcal{B}(\boldsymbol{\theta}^{(i+1)})\right) \quad \text{and} \tag{22}$$

$$\boldsymbol{\Sigma}_{\text{bias},i+1} = \frac{i-1}{i}\boldsymbol{\Sigma}_{\text{bias},i} + \frac{1}{i}(\mathcal{B}(\boldsymbol{\theta}^{(i+1)})\,\mathcal{B}(\boldsymbol{\theta}^{(i+1)})^{\intercal} - \boldsymbol{\mu}_{\text{bias},i+1}\,\boldsymbol{\mu}_{\text{bias},i+1}^{\intercal}) \tag{23}$$

While this approach does not in theory guarantee ergodicity of the chain (as is also the case with the Adaptive Metropolis proposal), the bias distribution will converge as the chain progresses and adaptation diminishes, resulting in a *de facto* ergodic process after an initial period of high adaptivity. This is a common feature of adaptive MCMC algorithms, as discussed in the classic paper on Adaptive Metropolis [17]. Our experiments showed that the bias distribution did indeed converge for every simulation, and that repeated experiments converged towards the same posterior bias distribution. Admitting a bias term in the inverse problem further introduces an issue of *identifiability*, as highlighted in [47]. Since observations are now modelled as a sum of coarse model output and multiple stochastic terms, the stochastic terms $\mathcal{B} \sim N(\boldsymbol{\mu}_{\text{bias}}, \boldsymbol{\Sigma}_{\text{bias}})$ and $\boldsymbol{\epsilon} \sim N(0, \sigma^2\mathbb{I}_n)$ are generally unidentifiable in the coarse model formulation, meaning that the bias $\mathcal{B}$ and the data modelling noise $\boldsymbol{\epsilon}$ are observationally equivalent, and not well-defined.

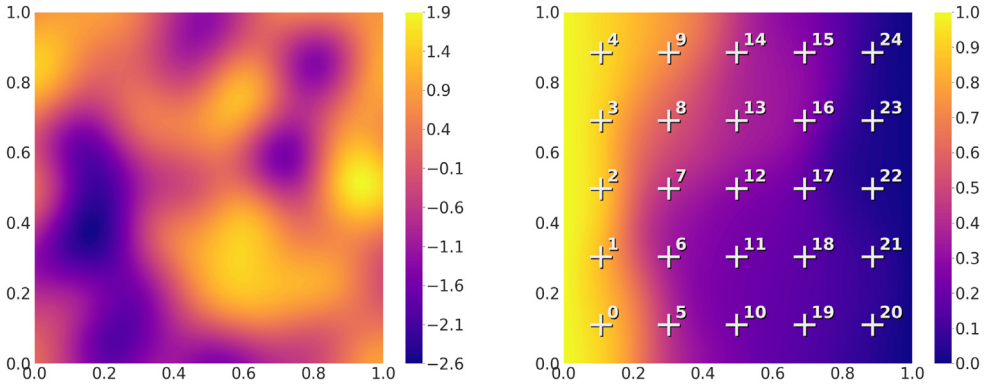
4. Results

In this section, we examine the effectiveness of our proposed strategy on two synthetic groundwater flow problems: a two-dimensional problem with an isotropic covariance kernel and a three-dimensional problem with an anisotropic covariance kernel. For both examples, we begin by outlining the model setup, for which we select a 'true' transmissivity field and a number of fixed observation points. For the first example, the influence of training size for the DNNs is examined, and the total cost of uncertainty quantification using a selection of DNNs is computed. For the second example we use a single DNN setup and analyse the resulting posterior marginal distributions and the quantity of interest. The first example was completed on commodity hardware — an HP Elitebook 840 G5 with an Intel Xeon E3-1200 quad-core processor, while the second example was completed on a TYAN Thunder FT48T-B7105 GPU server with two Intel Xeon Gold 6252 processors and an NVIDIA RTX 2080Ti GPU.

4.1. Example 1: 2D unit square

4.1.1. Model setup

This example was conducted on a unit square domain $\Omega = [0, 1]^2$, meshed using an unstructured triangular grid comprising 2,601 degrees of freedom. Dirichlet boundary conditions were imposed on the left and right boundaries with hydraulic heads of 1 and 0, respectively. The top and bottom edges impose homogeneous no-flow Neumann boundary conditions. To avoid committing an inverse crime, the covariance length scales of the ARD squared exponential kernel was set to $\boldsymbol{l} = (0.11, 0.11)^{\intercal}$ for data generation and $\boldsymbol{l} = (0.1, 0.1)^{\intercal}$ for the forward model used in sampling. The chosen length scales effectively resulted in an isotropic covariance kernel, equal to the 'classic'



(a) Log-Transmissivity. (b) Hydraulic head.

Fig. 5. "True" transmissivity field, its corresponding solution and sampling points.

Table 1
Neural network layers and activation functions in the model approximation DNNs.

Layer	# Nodes	Activation functions			
		DNN1	DNN2	DNN3	DNN4
Input	k KL coefficients	–	–	–	–
1	$4k$	Sigmoid	Sigmoid	Sigmoid	Sigmoid
2	$8k$	ReLU	ReLU	–	–
3	$4k$	ReLU	ReLU	ReLU	ReLU
Output	m datapoints	Exponential	Linear	Exponential	Linear

square exponential kernel with $l = 0.1$. This resulted in a KL decomposition with $> 80\%$ of total signal energy in the 32 largest eigenvalues and $> 95\%$ of signal energy in the 64 largest eigenvalues. Hence, 32 modes were included in the approximate model whilst 64 modes where included in the fine model.

Fig. 5(a) shows the 'true' transmissivity field that we attempt to recover through our MCMC methodology and the modelled, corresponding hydraulic head. Synthetic samples for the likelihood function were extracted at 25 points on a regular grid with a horizontal and vertical spacing of 0.2 m (Fig. 5(b)), and these data were perturbated with white noise with covariance $\boldsymbol{\Sigma}_e = 0.001\,\mathbb{I}_m$.

4.1.2. Deep neural network design, training and evaluation

We evaluated a selection of different DNNs to investigate the impact of various network depths and activation functions on the DNN performance. Table 1 shows the layers of the employed DNNs, the number of nodes in each layer and their corresponding activation functions. DNN1 and DNN2 had three hidden layers, while DNN3 and DNN4 had only two, as the ReLU layer with $8k$ nodes was not included in these networks. The output layer of DNN1 and DNN3 consisted of nodes with an exponential activation function $E(x) = e^x$, resulting in a strictly positive output. The DNNs with an exponential activation function in the final layer tended overall to lead to the best performance.

Each DNN was trained on a set of samples from the prior distribution of parameters $\pi_0(\boldsymbol{\theta}) = \mathcal{N}(0, \mathbb{I}_k)$, in advance of running the MCMC. Hence, the DNN samples were drawn from a Latin Hypercube [48] in the interval [0, 1] and transformed to the standard normal distribution using the *probit*-function, such that $\boldsymbol{\theta}_{train} \sim \mathcal{N}(0, \mathbb{I}_k)$. The coarse, 32-mode FEM model was then run for every parameter sample, obtaining for each a vector of model outputs at sampling points given parameters. We trained and tested each DNN on a range of different sample sizes, namely $N_{\text{DNN}} = \{2000, 4000, 8000, 16000, 32000, 64000\}$, where $N_{\text{DNN}} = N_{train} + N_{test}$, with a 9:1 training/test splitting ratio. Each DNN was then trained for 200 epochs with a batch size of 50 using the `rmsprop` optimiser [41].

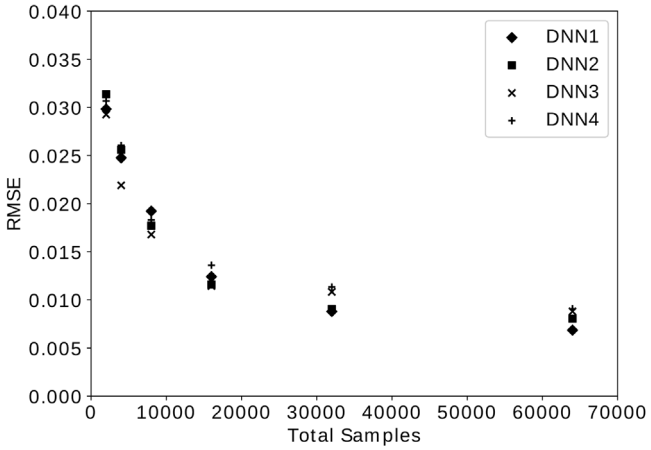


Fig. 6. Testing performance (RMSE) of each DNN against the total sample size ($N_{\text{DNN}} = N_{train} + N_{test}$). Please refer to Table 1 for details in the structure of each DNN.

Deep Neural Networks performance was compared using the RMSE of their respective testing dataset

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(h_i - \hat{h}_i)^2} \tag{24}$$

The residual RMSE (24) of each DNN was computed to compare the network designs described in Table 1 and to investigate the influence of training dataset size on the DNN performance (Fig. 6). As expected, each DNN performed better as the number of samples in the training dataset were increased. In comparison, the DNN design had much less influence on the testing performance, suggesting that the main driver for constructing an accurate surrogate model, within the bounds of the examined DNN designs, was the number of training samples. For the remaining analysis, we chose the network design resulting in the overall lowest RMSE at $N_{\text{DNN}} = 64000$, namely DNN1, and the sample sizes $N_{\text{DNN}} = \{4000, 16000, 64000\}$.

Further performance analysis consisted of analysing the DNN error $e = h_{\text{true}} - h_{\text{pred}}$ for true and predicted heads (h_{true} and h_{pred}, respectively) for datapoints 0, 8, 16 and 24. (Fig. 7). All error distributions were approximately Gaussian, with the errors for the DNN with $N_{\text{DNN}} = 4000$ exhibiting some right skew at sampling point 24. For all DNNs, the sampling points closer to the boundaries (at sampling points 0 and 24) had lower errors than those further away, since the heads close to the boundaries were more constrained by the model.

4.1.3. Uncertainty quantification

For inversion and uncertainty quantification, we chose a multivariate standard normal distribution as the prior parameter distribution, $\pi_0(\boldsymbol{\theta}) = \mathcal{N}(0, \mathbb{I}_k)$ and set the error covariance to $\boldsymbol{\Sigma}_e = 0.001\,\mathbb{I}_m$. While computationally convenient, the zero-centred prior in practice favours transmissivity field realisations capable of reproducing the observed heads with as little variation as possible. In total, eight different sampling strategies were investigated, namely single level 'Vanilla' MCMC, with no delayed acceptance, no adaptivity, and using only the 64-mode fine model; DA using three different DNNs trained and tested on $N_{\text{DNN}} = \{4000, 16000, 64000\}$ samples as the coarse model and the 64-mode model as the fine; and DA with an enhanced error model (DA/EEM) using the same three DNNs. The offset length t for the DA strategies was manually tuned to achieve an acceptance rate of $a \in [0.2, 0.4]$. To investigate the effect of the offset length t independently of other factors, an additional simulation with $N_{\text{DNN}} = 64000$ and $t = 1$ was also completed. In this first example, every simulation was completed using the pCN transition kernel, with $\beta = 0.15$. Each MCMC sampling strategy was repeated ($n = 32$) using randomly generated random seeds, to ensure that every starting point would converge towards the same stationary distribution and to allow for cross-chain statistics to be computed. Results given in this section pertain to these multi-chain samples rather than individual MCMC realisations, unless otherwise stated.

Our sampling strategies recovered the ground truth with good accuracy. Fig. 8 shows the mean and variance of the recovered field from the DA/EEM MCMC using the DNN with $N_{\text{DNN}} = 64000$. All recovered fields exhibit

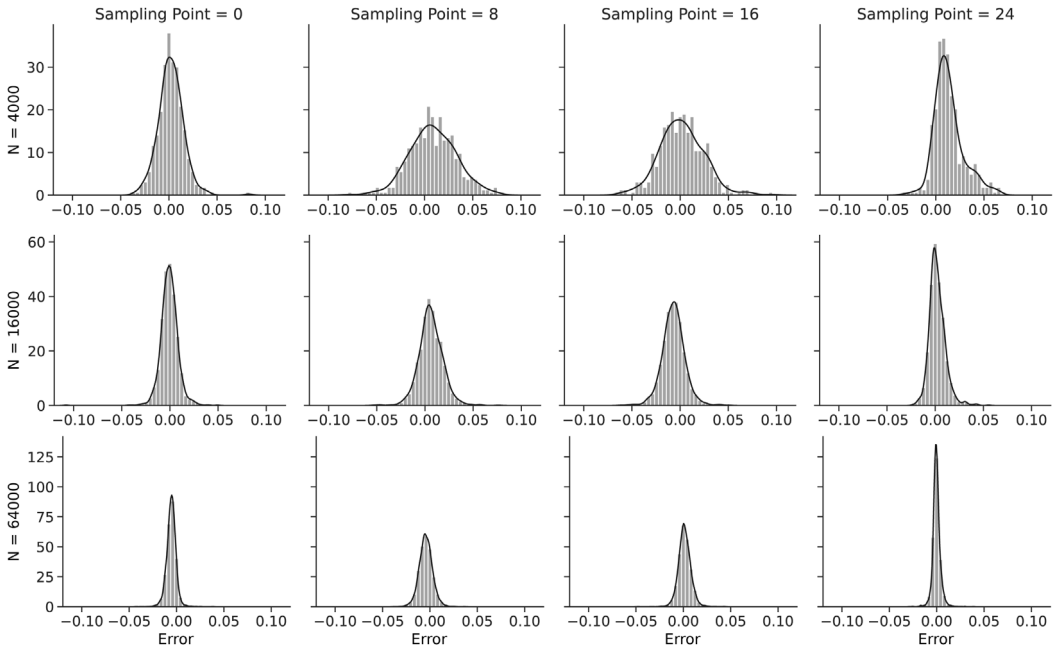


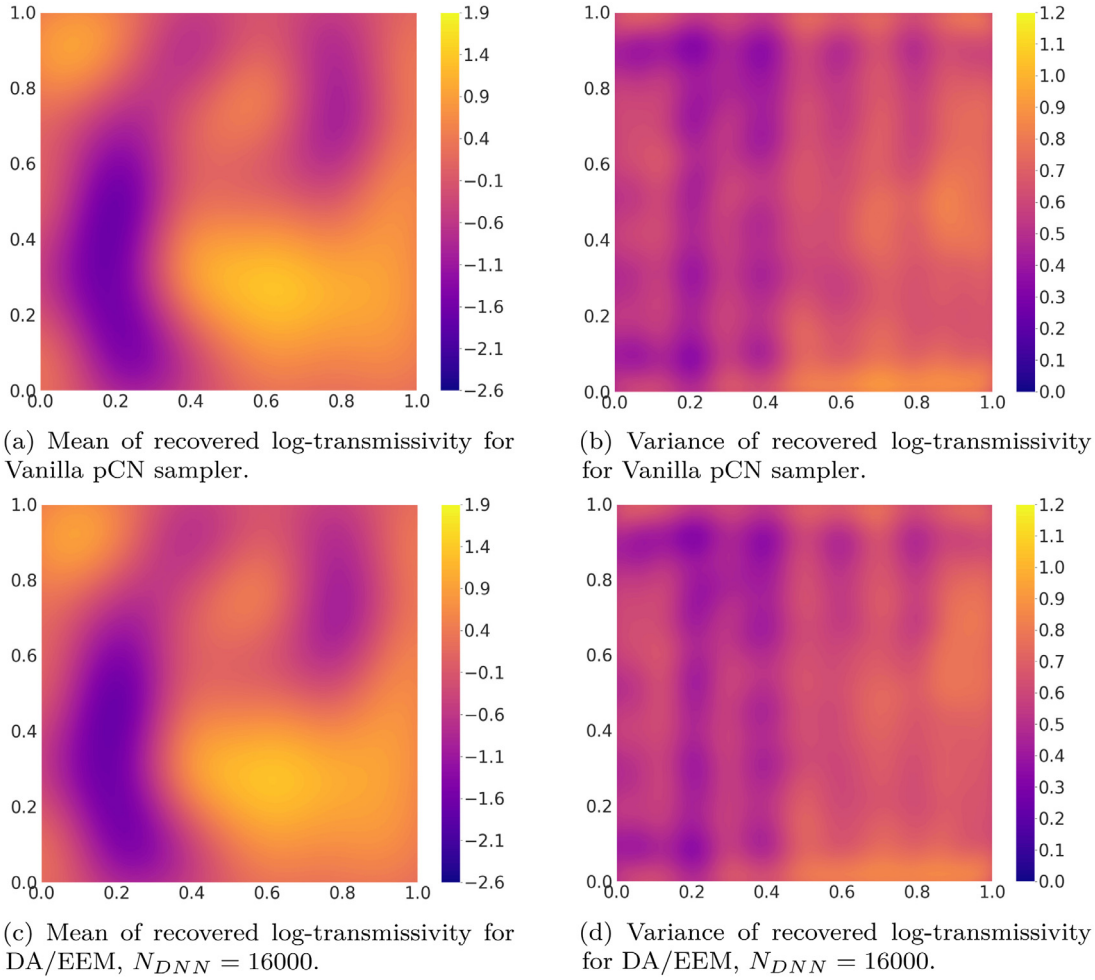
Fig. 7. Density plot of the error ($e = h_{\text{true}} - h_{\text{pred}}$) of the testing dataset for DNN1 trained and tested on $N_{\text{DNN}} = \{4000, 16000, 64000\}$ samples, for sampling points 0, 8, 16 and 24. Bars show density of each bin, while the curve shows Gaussian kernel density estimate.

Table 2
Results for various MCMC sampling strategies, means of multiple chains with $n = 32$. N_{DNN} is the number of total samples used to construct the DNN. t is the improved DA offset length. N_C/N_F is the final length of the coarse and fine chain, respectively, after subtracting burnin. *Acc. rate* is the fine chain acceptance rate. *Time* (min) is the total running time of the simulation in minutes. N_{eff} is the Effective Sample Size.

Strategy	N_{DNN}	t	N_C/N_F	Acc. Rate	Time (min)	N_{eff}
Vanilla	–	–	—/40000	0.33	32.1	85.6
DA	4000	2	85461.9/20000	0.27	16.2	64.5
DA/EEM	4000	2	78853.4/20000	0.31	15.2	79.0
DA	16000	4	172383.1/20000	0.27	18.2	116.3
DA/EEM	16000	4	178978.4/20000	0.30	18.4	143.6
DA	64000	8	336447.5/20000	0.24	30.1	196.5
DA/EEM	64000	8	377524.4/20000	0.30	29.9	235.7
DA/EEM	64000	1	56824.3/20000	0.57	15.3	68.6

higher smoothness than the ground truth, which can be attributed to the relatively low number of sampling points and their regular distribution on the domain, in combination with the regularisation introduced by the prior. Since the KL decomposition incorporated $> 95\%$ of the signal energy, the truncation would have contributed only marginally to the smoothing. None of the chains recovered the local peak in transmissivity on the right side of the domain, since there was too little data to discover this particular feature. However, this peak is clearly encapsulated by the posterior variance, as shown in Figs. 8(b) and 8(d).

While the recovered fields indicate that every MCMC sampling strategy converged towards the desired stationary distribution, they do not reveal the relative efficiency of each strategy. Hence, the Effective Sample Size (N_{eff}) was computed for each MCMC realisation. Every DA sampling strategy produced higher N_{eff} than the Vanilla pCN sampler, relative to the simulation time, with a clear correlation between DNN testing performance and N_{eff}. This was mainly because the better performing DNNs allowed for a longer coarse chain offset without diverging. Moreover, utilising the EEM produced even higher N_{eff} for every DA chain (Table 2).



(a) Mean of recovered log-transmissivity for Vanilla pCN sampler.

(b) Variance of recovered log-transmissivity for Vanilla pCN sampler.

(c) Mean of recovered log-transmissivity for DA/EEM, $N_{DNN} = 16000$.

(d) Variance of recovered log-transmissivity for DA/EEM, $N_{DNN} = 16000$.

Fig. 8. Mean and variance ($n = 32$) of recovered log-transmissivity fields using Vanilla pCN sampler (top) and DA/EEM MCMC with $N_{\text{DNN}} = 16000$ (bottom). Corresponding plots of every sampling strategy are shown in Figs. B.15–B.21 in Appendix B.

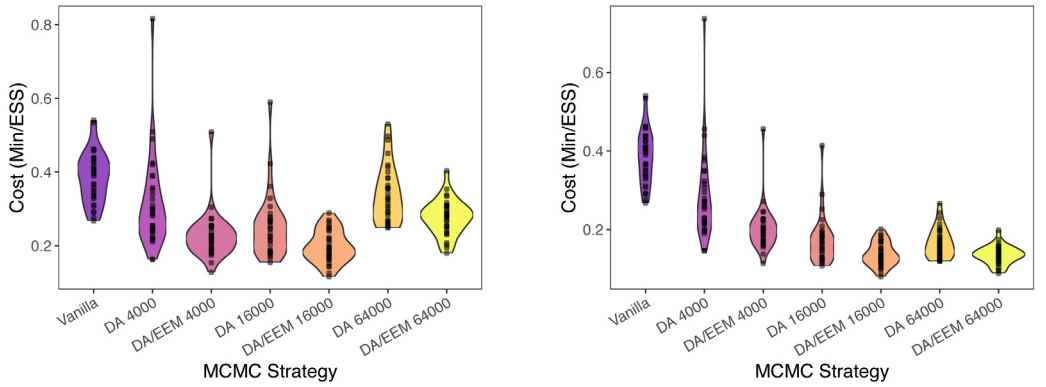
4.1.4. Total cost

Since the DA chains required computation of a significant number of fine model solutions and training of a DNN in advance of running the chain, the total cost C_{total} of each strategy was computed as

$$\text{C}_{\text{total}} = \frac{t_{\text{fine}} + t_{\text{train}} + t_{\text{run}}}{N_{eff}} \tag{25}$$

where t_{fine} was the time spent on precomputing fine model solution, t_{train} was the time spent on training the respective DNN, t_{run} was the time taken to run the chain and N_{eff} was the resulting effective sample size (Fig. 9).

The mean cost of every DA chain was lower than that of the Vanilla pCN chain, with the chains using the EEM consistently cheaper than their non-EEM counterparts. Moreover, using the EEM reduced the variance of the cost in repeated experiments, allowing each repetition to produce a consistently high N_{eff}. The overall cheapest inversion was completed using the DNN trained on 16,000 samples using the EEM, reducing the total cost, relative to the Vanilla pCN MCMC, with 50%. Notice that these results are extremely conservative in the sense that the entire cost of evaluating every DNN training sample and training the DNN in serial on a CPU was factored into the cost of every repetition, even though the same DNN was used for all the repetitions within each sampling strategy. The precomputation cost can be dramatically reduced by evaluating the DNN samples in parallel and utilising high-performance hardware, such as GPUs, for training the DNN.



(a) Total cost (conservative) with the full cost of constructing the DNN factored into all independent DA chains.

(b) Total cost (normalised) with the cost of constructing the DNN distributed between independent DA chains.

Fig. 9. Violinplots showing the total cost C_{total} of each MCMC strategy with $n = 32$. Points show independent Markov Chains.

4.2. Example 2: 3D rectangular cuboid

4.2.1. Model setup

This example was conducted on a rectangular cuboid domain $\Omega = [0, 2] \times [0, 1] \times [0, 0.5]$ meshed using an unstructured tetrahedral grid with 10,416 degrees of freedom (Fig. 10). Dirichlet boundary conditions of $h = 1$ and $h = 0$ were imposed at $x_1 = 0$ and $x_1 = 2$, respectively. No-flow Neumann conditions were imposed on all remaining boundaries.

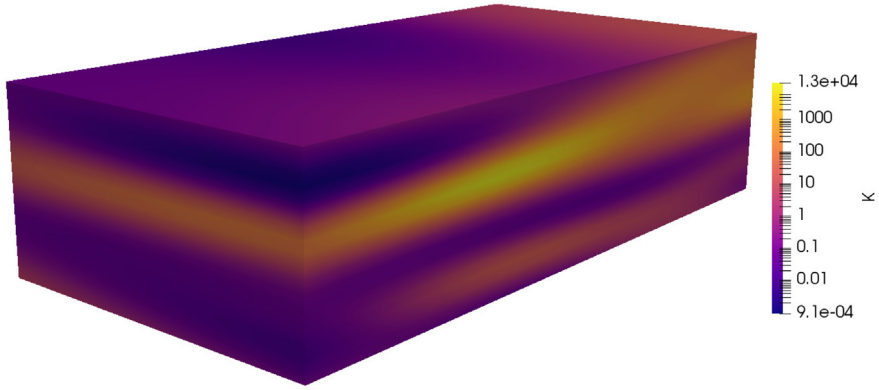
The covariance lengths scales for ARD squared exponential covariance kernel were set to $\boldsymbol{l} = (0.55, 0.95, 0.06)^\intercal$ for data generation and $\boldsymbol{l} = (0.5, 1.0, 0.05)^\intercal$ for the forward model used in sampling, resulting in a highly anisotropic random process with high variation in the x_3 direction to simulate geological stratification, some variation in the x_1 direction and little variation in the x_2 direction (Fig. 10(a)). Like in the first model, the random process was truncated at 64 KL eigenmodes for the fine model and 32 KL eigenmodes for the coarse model, embodying $> 97\%$ and $> 90\%$ of the total signal energy, respectively.

We drew $w = 50$ sampling well locations randomly using the Maximin Latin Hypercube Design [49], and samples of hydraulic head were extracted at each well at datums $x_3 = \{0.05, 0.15, 0.25, 0.35, 0.45\}$, measured from the bottom of the domain, resulting in $m = 250$ datapoints in total (Fig. 10(b)). These data were perturbated with white noise with covariance $\boldsymbol{\Sigma}_e = 0.001\, \mathbb{I}_m$.

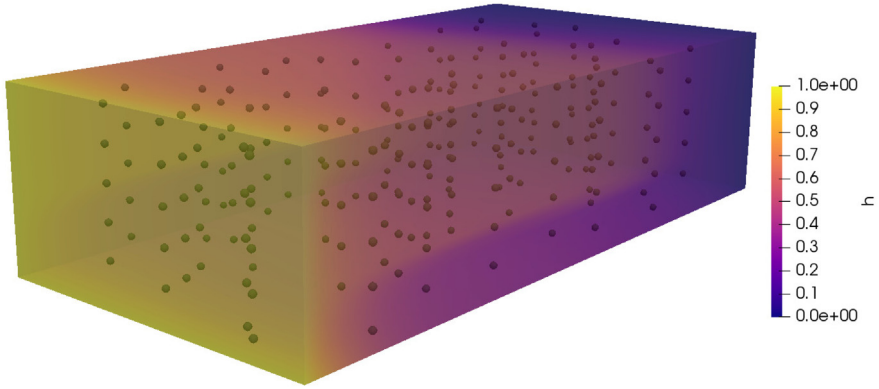
For this example, we first converged the conductivity parameters to the Maximum a posteriori (MAP) estimate $\boldsymbol{\theta}_{MAP} = \arg\max_{\theta} \pi_0(\boldsymbol{\theta})\mathcal{L}(\boldsymbol{d}_{\text{obs}}|\boldsymbol{\theta})$ using gradient descent, since initial MCMC experiments struggled to converge to the posterior distribution for random initial parameter sets.

4.2.2. Deep neural network design, training and evaluation

Training a DNN to accurately emulate the model response for this setup was challenging, and we found no single combination of neural network layers and activation functions that would predict the head at every datapoint with sufficient accuracy. We hypothesise that this limitation could be caused by a strong ill-posedness of the DNN — for a single neural network, the output dimension greatly exceeded the input dimension, i.e. $m \gg k$ where $m = 250$ was the number of datapoints, and $k = 32$ was the coarse model KL modes. When we instead predicted the heads at each datapoint datum using a separate DNN, we found that we could utilise largely the same DNN design as had been employed in the first example. Hence, to predict the head at all datapoints, we utilised five identically designed but independent DNNs (Fig. 11), each with four hidden layers and activation functions as indicated in Table 3. Each DNN was trained and tested on a dataset of $N_{DNN} = 16000$ samples with KL coefficients drawn from a Latin Hypercube [48] in the interval $[0, 1]$ and transformed to a normal distribution centred on the MAP estimate of the parameters $\boldsymbol{\theta}_{MAP}$, i.e. $\boldsymbol{\theta}_{train} \sim \mathcal{N}(\boldsymbol{\theta}_{MAP}, \mathbb{I}_k)$. This was done to increase the density of samples and



(a) Log-Conductivity of ground truth.



(b) Hydraulic head of ground truth and location of sampling points.

Fig. 10. "True" conductivity field, its corresponding solution and sampling points.

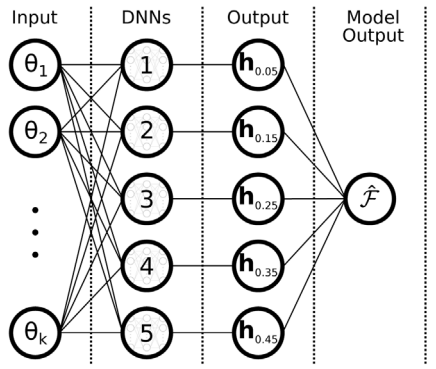


Fig. 11. Layout of the multi-DNN design. Each DNN outputs a vector $\boldsymbol{h}_{x_3}$ vector of w head predictions at datum x_3.

thus improve the DNN prediction at and around the MAP point, which ideally equals the mode of the posterior distribution. The DNNs were then trained for 200 epochs using a batch size of 50, the MSE loss function and the `rmsprop` optimiser [41]. Fig. 12 shows performance plots of each DNN for both the training (top) and the testing (bottom) datasets. While every DNN is clearly moderately biased by the training data, they all performed adequately with respect to the testing data.

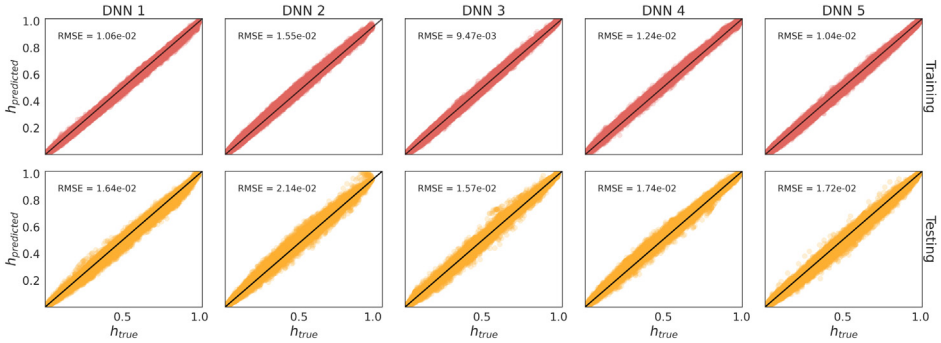


Fig. 12. Performance of the five DNNs used in the multi-DNN approach, as shown in Fig. 11, with respect to the training dataset (top) and the testing dataset (bottom).

Table 3
Layers and activation functions in the four DNNs. Each DNN takes all k KL coefficients as input and predicts the head $\boldsymbol{h}_{x_3}$ at w wells for a given datum.

Layer	# Nodes	Activation functions
Input	k KL coefficients	–
1	$4k$	Sigmoid
2	$8k$	ReLU
3	$8k$	ReLU
3	$4k$	ReLU
Output	w wells	Exponential

4.2.3. Uncertainty quantification

Similarly to the first example, we chose a multivariate standard normal distribution $\pi_0(\boldsymbol{\theta}) = \mathcal{N}(0, \mathbb{I}_k)$ as the prior distribution of parameters, and set the error covariance to $\boldsymbol{\Sigma}_e = 0.001\,\mathbb{I}_m$. Hence, the synthetic head data from the wells were perturbated with white noise with covariance $\boldsymbol{\Sigma}_e$. In this example, we utilised the Adaptive Metropolis (AM) transition kernel for generating proposals. We completed $n = 8$ independent simulations, each initialised from a random initial point close to the MAP point $\boldsymbol{\theta}_{MAP}$, with a burnin of 1000 and a final sample size of $N = 10{,}000$. The subchains were run with an acceptance delay of $t = 2$, since longer subchains tended to diverge, leading to sub-optimal acceptance rates on the fine level. The simulations had a mean acceptance rate of 0.26, a mean effective sample size (N_{eff}) of 55.2 and a mean autocorrelation length $\tau = N/N_{eff}$ of 181.0. The samples of each independent simulation were pruned according to the respective autocorrelation length, and the remaining samples were pooled together to yield 443 statistically independent samples that were then analysed further.

Fig. 13 shows the marginal distributions of the six coarsest KL coefficients along with a scatterplot matrix of all the samples remaining after pruning. All the marginal distributions are approximately Gaussian, and the two-parameter marginal distributions are mostly elliptical. It is evident that some of these parameters are correlated, namely parameters $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_5)$, $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$, $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_3)$, $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_4)$ and $(\boldsymbol{\theta}_2, \boldsymbol{\theta}_4)$. It is worth mentioning that in every independent simulation, the AM proposal kernel managed to capture these correlations.

Moreover, we analysed the hydraulic head as a function of datum $h(x_3)$ along a line in the centre of the domain $\boldsymbol{x} = (1.0, 0.5, x_3)^\intercal$. Fig. 14 shows $h(x_3)$ of the ground truth, MAP point $\boldsymbol{\theta}_{MAP}$, the mean of the $n = 8$ independent simulations, and all the samples remaining after pruning. We observe that both the MAP point and the sample mean are fairly close to the ground truth, albeit exhibiting higher smoothness, particularly between the observation depths, where the head is essentially allowed to vary freely. It is also clear that the individual samples encapsulate the ground truth, indicating that the ground truth is indeed contained by posterior distribution.

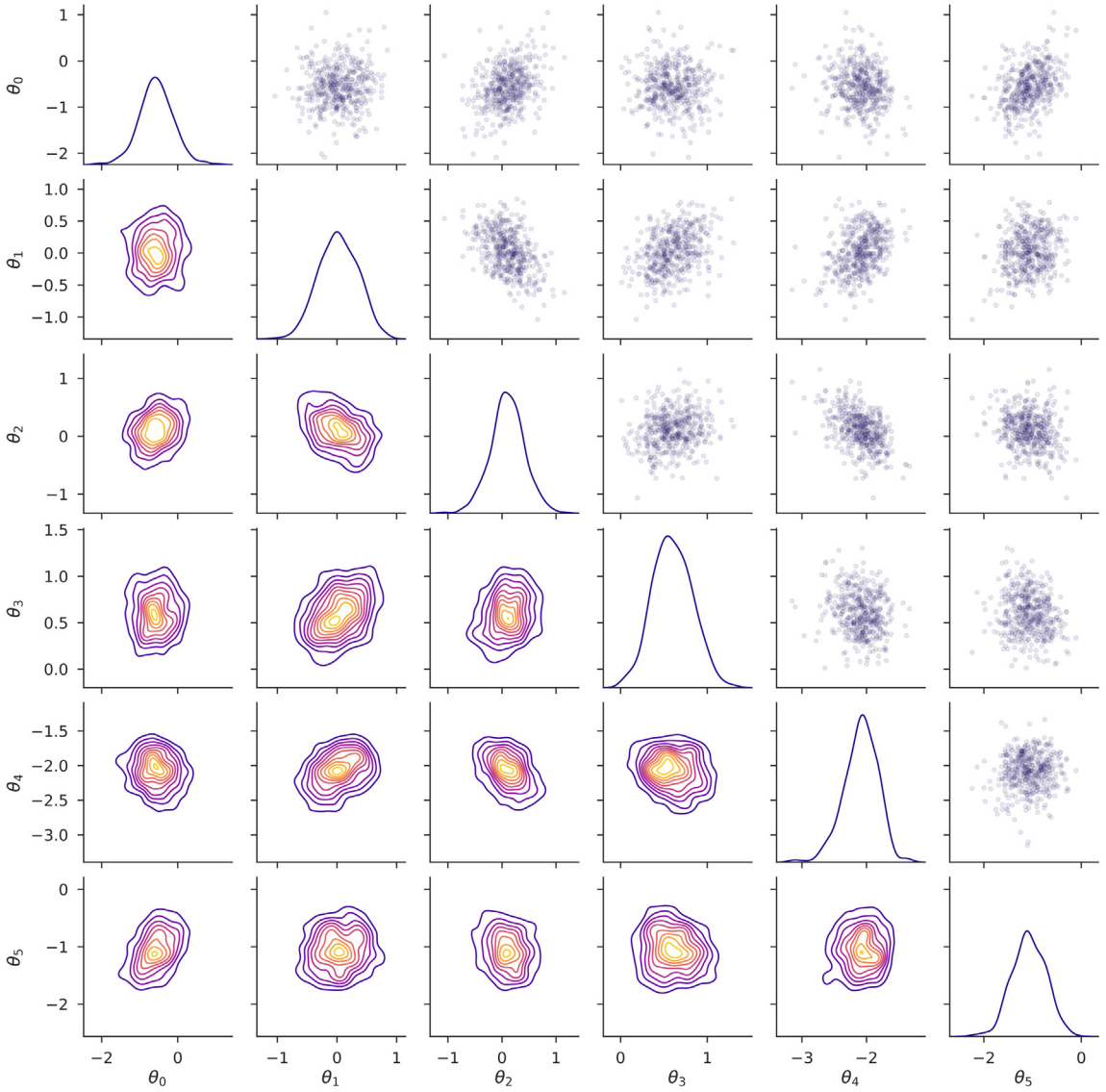


Fig. 13. One and two-dimensional posterior marginal distributions (diagonal and lower triangle) and scatterplots (upper triangle) of posterior samples pruned according to the autocorrelation length of each chain for the largest 5 KL eigenmodes. Please note that the axis scales of are not equal.

5. Discussion

In this paper, we have demonstrated the use of a novel Markov Chain Monte Carlo methodology which employs a delayed acceptance (DA) model hierarchy with a deep neural network (DNN) as an approximate model and a FEM solver as a fine model, and generates proposals using the pCN and AM transition kernels. Results from the first example clearly indicate that the use of a carefully designed DNN as a model approximation can significantly reduce the cost of uncertainty quantification, even for DNNs trained on relatively small sample sizes. We have established that offsetting fine model evaluations in the DA algorithm reduces the autocorrelation of the fine chain, resulting in a higher effective sample size which, in turn, improves the statistical validity of the results. In this context, the performance of the DNN is a critical driver when determining a feasible offset length to avoid divergence of the coarse chain. Hence, if a high effective sample size is required, it may be desirable to invest in

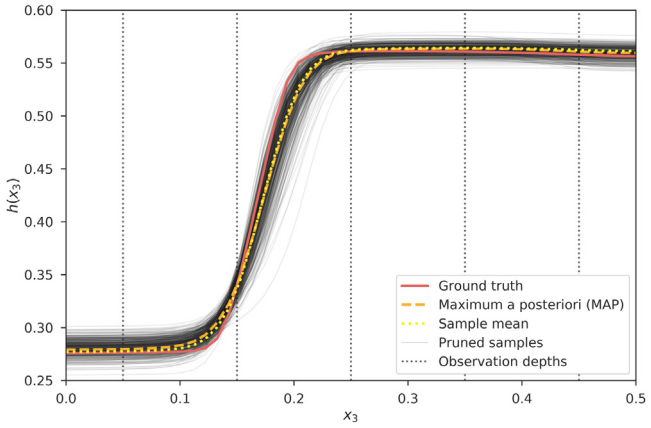


Fig. 14. Hydraulic head as a function of datum $h(x_3)$ at $\boldsymbol{x} = (1.0, 0.5, x_3)^\intercal$. The solid red line shows the hydraulic head of the ground truth, the dashed orange line shows the head of the Maximum a posteriori (MAP) point $\boldsymbol{\theta}_{MAP}$, the dotted yellow line shows the mean head of the independent simulations ($n = 8$) and the thin black lines show the head of 538 statistically independent samples, remaining after pruning according to the autocorrelation length of each chain, $n = 443$. The vertical dotted lines show the observation depths.

a well-performing DNN. Moreover, we have shown that an enhanced error model, which introduces an iteratively-constructed bias distribution in the coarse chain likelihood function, further increases the effective sample size and decreases the variance of the cost in repeated experiments. Finally, we observed that for the second example, even when employing a relatively well-performing model approximation, we had to constrain the offset length of the subchains rather strongly to achieve optimal acceptance rates. This can be attributed in part to an apparent non-spherical and correlated posterior distribution, causing the employed proposal kernels to struggle to discover areas of high posterior probability.

We have demonstrated that relatively simple inverse hydrogeological problems can be solved in reasonable time on a commonly available personal computer with no GPU-acceleration. This opens the opportunity to apply robust uncertainty quantification during fieldwork and as a decision-support tool for groundwater surveying campaigns. We have also demonstrated the applicability of our approach on a larger scale three-dimensional problem, utilising a GPU-accelerated high-performance computer (HPC). Aside from the benefit of using a HPC computer for accelerating the fine model evaluations, utilising the GPU allowed for rapidly training and testing multiple different DNN designs to efficiently establish a well performing model approximation. There are other obvious ways to further increase the efficiency of the proposed methodology. For example, construction of the DNNs used as coarse models comes with the cost of evaluating multiple models from the prior distribution, and, unlike the MCMC sampler, the prior models are independent and these fine model evaluations can thus be massively parallelised.

Our methodology was demonstrated in the context of two relatively simple groundwater flow problems with log-Gaussian transmissivity fields parametrised by Karhunen–Loève decompositions. While this model provides a convenient computational structure for our purposes, it may not reflect the full scale transmissivity of real-world aquifers, particularly in the presence of geological faults and other heterogeneities, as discussed in [24]. Future research could address this problem through geological layer stratification using the universal cokriging interpolation method suggested in [50], potentially utilising the open-source geological modelling tool GemPy [51], which allows for simple parametric representation of geological strata. Spatially heterogeneous parameters within each strata could then be modelled hierarchically using a low order log-Gaussian random field to account for within-stratum variation, as demonstrated in [12].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was funded as part of the Water Informatics Science and Engineering Centre for Doctoral Training (WISE CDT) under a grant from the Engineering and Physical Sciences Research Council (EPSRC), UK, grant number EP/L016214/1. TD was funded by a Turing AI Fellowship, UK (2TAFFP\100007). DM acknowledges support from the EPSRC Platform Grant PRISM, UK (EP/R029423/1). The authors have no competing interests. Data supporting the findings in this study are available in the Open Research Exeter (ORE, https://ore.exeter.ac.uk/repository/) data repository.

Appendix A. Preconditioned Crank–Nicolson

The preconditioned Crank–Nicolson (pCN) proposal was developed in [16] and is based on the following Stochastic Partial Differential Equation (SPDE):

$$\frac{du}{ds} = -\mathcal{K}\mathcal{L}u + \sqrt{2\mathcal{K}}\frac{db}{ds}$$

where $\mathcal{L} = \mathcal{C}^{-1}$ is the precision operator for the prior distribution μ_0, b is brownian motion with covariance operator I, and $\mathcal{K}$ is a positive operator. This equation can be discretised using the Crank–Nicolson approach to yield

$$v = u - \frac{1}{2}\delta\mathcal{K}\mathcal{L}(u+v) + \sqrt{2\mathcal{K}\delta}\xi_0$$

for white noise ξ_0 and a weight $\delta \in [0, 2]$. If we choose $\mathcal{K} = I$, we get the plain Crank–Nicolson (CN) proposal:

$$(2\mathcal{C} + \delta I)v = (2\mathcal{C} - \delta I)u + \sqrt{8\delta\mathcal{C}}\xi$$

where $\xi \sim \mathcal{N}(0, \mathcal{C})$. If we instead choose $\mathcal{K} = \mathcal{C}$, we get the pCN proposal:

$$v = \sqrt{1-\beta^2}u + \beta\xi, \quad \beta = \frac{\sqrt{8\delta}}{2+\delta}, \quad \beta \in [0, 1]$$

This is rewritten, conforming to our previous notation:

$$\boldsymbol{\theta}' = \sqrt{1-\beta^2}\boldsymbol{\theta}_i + \beta\boldsymbol{\xi}$$

Appendix B. Recovered conductivity fields

See Figs. B.15–B.21.

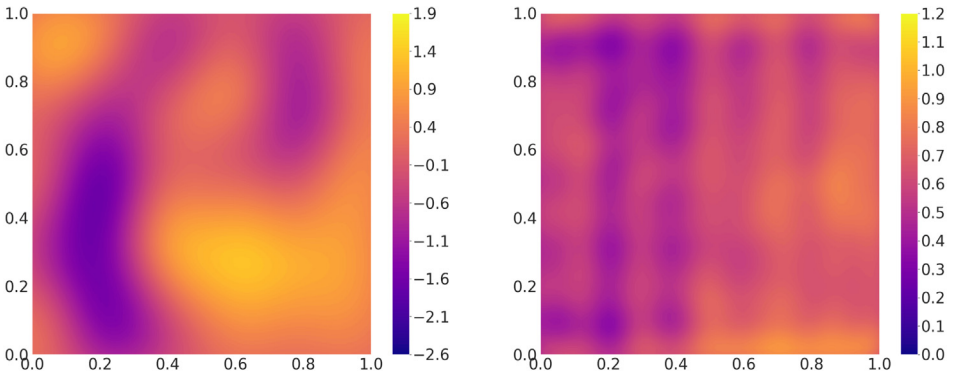


Fig. B.15. Mean (left) and variance (right) of recovered log-transmissivity for Vanilla pCN.

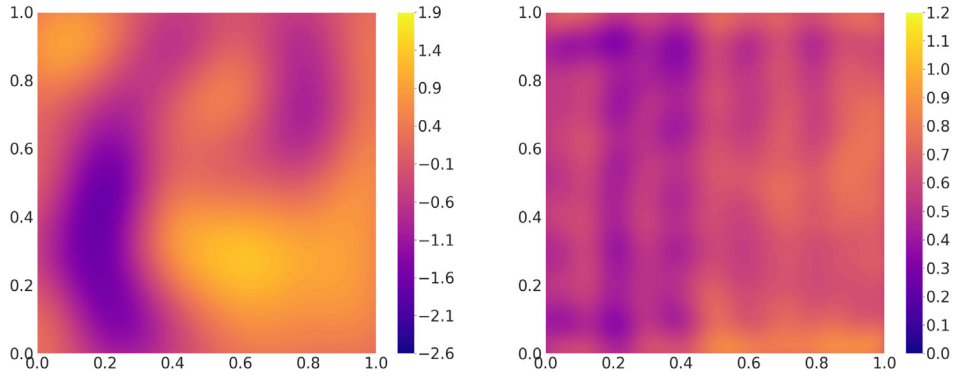


Fig. B.16. Mean (left) and variance (right) of recovered log-transmissivity for DA, $N_{DNN} = 4000$.

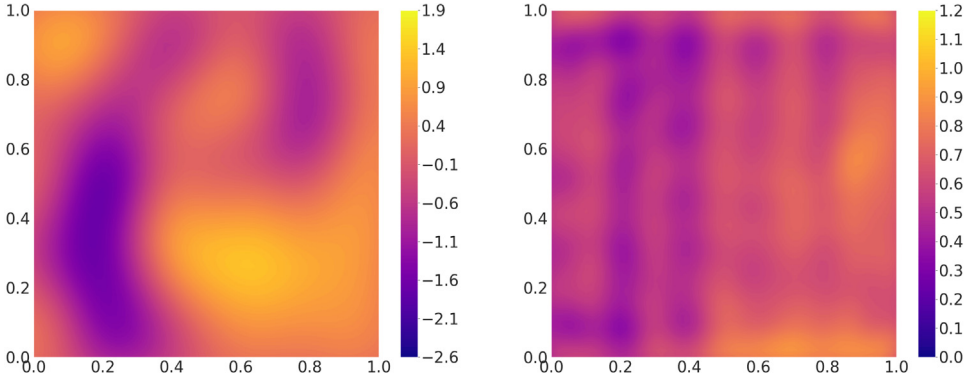


Fig. B.17. Mean (left) and variance (right) of recovered log-transmissivity for DA/EEM, $N_{DNN} = 4000$.

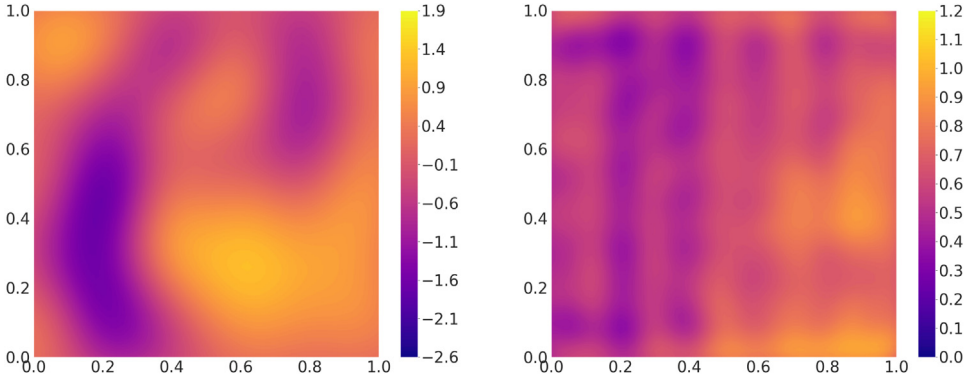


Fig. B.18. Mean (left) and variance (right) of recovered log-transmissivity for DA, $N_{DNN} = 16000$.

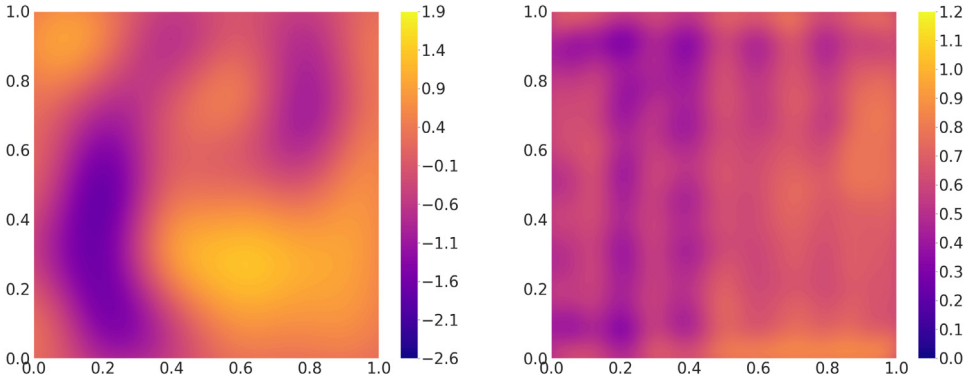


Fig. B.19. Mean (left) and variance (right) of recovered log-transmissivity for DA/EEM, $N_{DNN} = 16000$.

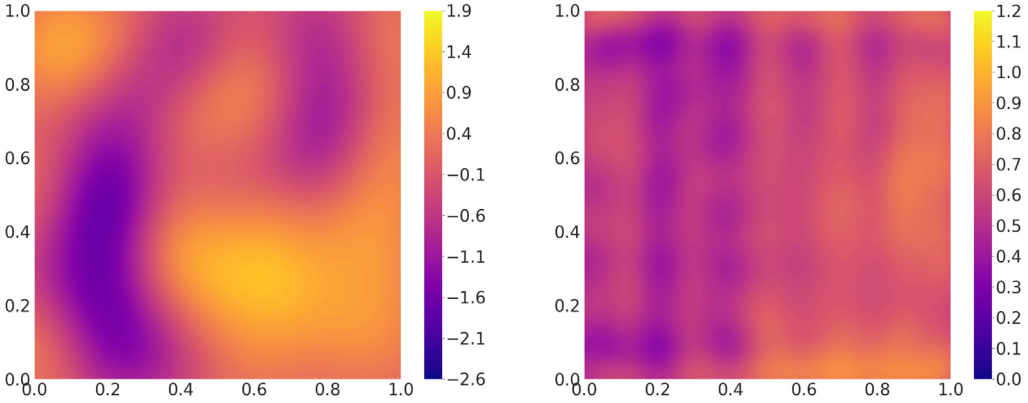


Fig. B.20. Mean (left) and variance (right) of recovered log-transmissivity for DA, $N_{DNN} = 64000$.

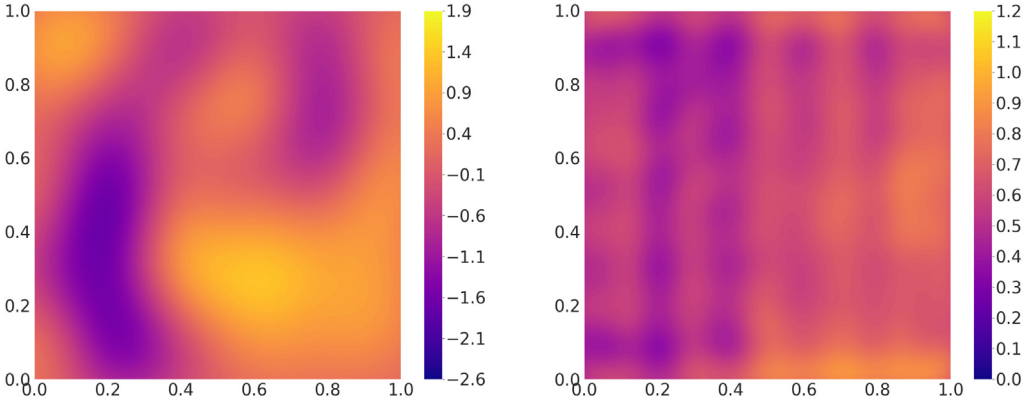


Fig. B.21. Mean (left) and variance (right) of recovered log-transmissivity for DA/EEM, $N_{DNN} = 64000$.

References

[1] M.P. Anderson, W.W. Woessner, R.J. Hunt, Applied Groundwater Modeling: simulation of Flow and Advective Transport, second ed., Academic Press, London; San Diego, CA, 2015, oCLC: ocn921253555.

[2] A.D. Woodbury, T.J. Ulrych, A full-Bayesian approach to the groundwater inverse problem for steady state flow, Water Resour. Res. 36 (8) (2000) 2081–2093, http://dx.doi.org/10.1029/2000WR900086, URL http://doi.wiley.com/10.1029/2000WR900086.
[3] G. Mariethoz, P. Renard, J. Caers, Bayesian inverse problem and optimization with iterative spatial resampling: ITERATIVE SPATIAL RESAMPLING, Water Resour. Res. 46 (11) (2010) http://dx.doi.org/10.1029/2010WR009274, URL http://doi.wiley.com/10.1029/2010WR009274.
[4] M. de la Varga, J.F. Wellmann, Structural geologic modeling as an inference problem: A Bayesian perspective, Interpretation 4 (3) (2016) SM1–SM16, http://dx.doi.org/10.1190/INT-2015-0188.1, URL http://library.seg.org/doi/10.1190/INT-2015-0188.1.
[5] C.P. Robert, G. Casella, Monte Carlo Statistical Methods, second ed., in: Springer Texts in Statistics, Springer, New York, NY, 2010, oCLC: 837651914.
[6] D. Higdon, H. Lee, C. Holloman, Markov chain Monte Carlo-based approaches for inference in computationally intensive inverse problems, in: Bayesian Statistics, Vol. 7, Oxford University Press., 2003, pp. 181–197.
[7] T.J. Dodwell, C. Ketelsen, R. Scheichl, A.L. Teckentrup, A hierarchical multilevel Markov chain Monte Carlo algorithm with applications to uncertainty quantification in subsurface flow, SIAM/ASA J. Uncertain. Quantif. 3 (1) (2015) 1075–1108, http://dx.doi.org/10.1137/130915005, URL http://epubs.siam.org/doi/10.1137/130915005.
[8] G. Detommaso, T. Dodwell, R. Scheichl, Continuous level Monte Carlo and sample-adaptive model hierarchies, 2018, arXiv:1802.07539 [math]. URL http://arxiv.org/abs/1802.07539.
[9] J. Doherty, Calibration and Uncertainty Analysis for Complex Environmental Models, 2015, oCLC: 991568728.
[10] B. Peherstorfer, K. Wilcox, M. Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization, SIAM Rev. 60 (3) (2018) 550–591.
[11] Y. Efendiev, A. Datta-Gupta, V. Ginting, X. Ma, B. Mallick, An efficient two-stage Markov chain Monte Carlo method for dynamic data integration, Water Resour. Res. 41 (12) (2005) http://dx.doi.org/10.1029/2004WR003764, URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2004WR003764.
[12] A. Mondal, Y. Efendiev, B. Mallick, A. Datta-Gupta, Bayesian uncertainty quantification for flows in heterogeneous porous media using reversible jump Markov chain Monte Carlo methods, Adv. Water Resour. 33 (3) (2010) 241–256, http://dx.doi.org/10.1016/j.advwatres.2009.10.010, URL https://linkinghub.elsevier.com/retrieve/pii/S0309170809001729.
[13] P. Dostert, Y. Efendiev, B. Mohanty, Efficient uncertainty quantification techniques in inverse problems for Richards' equation using coarse-scale simulation models, Adv. Water Resour. 32 (3) (2009) 329–339, http://dx.doi.org/10.1016/j.advwatres.2008.11.009, URL https://linkinghub.elsevier.com/retrieve/pii/S0309170808002121.
[14] E. Laloy, B. Rogiers, J.A. Vrugt, D. Mallants, D. Jacques, Efficient posterior exploration of a high-dimensional groundwater model from two-stage Markov chain Monte Carlo simulation and polynomial chaos expansion: Speeding up MCMC simulation of a groundwater model, Water Resour. Res. 49 (5) (2013) 2664–2682, http://dx.doi.org/10.1002/wrcr.20226, URL http://doi.wiley.com/10.1002/wrcr.20226.
[15] J.A. Christen, C. Fox, Markov chain Monte Carlo using an approximation, J. Comput. Graph. Statist. 14 (4) (2005) 795–810, http://dx.doi.org/10.1198/106186005X76983, URL http://www.tandfonline.com/doi/abs/10.1198/106186005X76983.
[16] S.L. Cotter, G.O. Roberts, A.M. Stuart, D. White, MCMC methods For functions: Modifying old algorithms to make them faster, Statist. Sci. 28 (3) (2013) 424–446, http://dx.doi.org/10.1214/13-STS421, URL http://arxiv.org/abs/1202.0709.
[17] H. Haario, E. Saksman, J. Tamminen, An adaptive metropolis algorithm, Bernoulli 7 (2) (2001) 223, http://dx.doi.org/10.2307/3318737, URL https://www.jstor.org/stable/3318737?origin=crossref.
[18] T. Cui, C. Fox, M.J. O'Sullivan, A eriori stochastic correction of reduced models in delayed acceptance MCMC, with application to multiphase subsurface inverse problems, 2018, arXiv:1809.03176 [stat]. URL http://arxiv.org/abs/1809.03176.
[19] H.-J.G. Diersch, FEFLOW: Finite Element Modeling of Flow, Mass and Heat Transport in Porous and Fractured Media, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, http://dx.doi.org/10.1007/978-3-642-38739-5, URL http://link.springer.com/10.1007/978-3-642-38739-5.
[20] H.P. Langtangen, A. Logg, Solving PDEs in Python – The FEniCS Tutorial Volume I, 2017.
[21] A.W. Harbaugh, MODFLOW-2005: The U.S. Geological Survey Modular Ground-Water Model–the Ground-Water Flow Process, Report, 2005, http://dx.doi.org/10.3133/tm6A16, URL http://pubs.er.usgs.gov/publication/tm6A16.
[22] R.A. Freeze, A stochastic-conceptual analysis of one-dimensional groundwater flow in nonuniform homogeneous media, Water Resour. Res. 11 (5) (1975) 725–741, http://dx.doi.org/10.1029/WR011i005p00725, URL http://doi.wiley.com/10.1029/WR011i005p00725.
[23] N.-O. Kitterrød, L. Gottschalk, Simulation of normal distributed smooth fields by Karhunen-Loéve expansion in combination with kriging, Stoch. Hydrol. Hydraul. 11 (6) (1997) 459–482, http://dx.doi.org/10.1007/BF02428429, URL http://link.springer.com/10.1007/BF02428429.
[24] J. Gómez-Hernández, X.-H. Wen, To be or not to be multi-Gaussian? A reflection on stochastic hydrogeology, Adv. Water Resour. 21 (1) (1998) 47–61, http://dx.doi.org/10.1016/S0309-1708(96)00031-0, URL http://linkinghub.elsevier.com/retrieve/pii/S0309170896000310.
[25] J. Kerrou, P. Renard, H.-J. Hendricks Franssen, I. Lunati, Issues in characterizing heterogeneity and connectivity in non-multiGaussian media, Adv. Water Resour. 31 (1) (2008) 147–159, http://dx.doi.org/10.1016/j.advwatres.2007.07.002, URL https://linkinghub.elsevier.com/retrieve/pii/S0309170807001236.
[26] D. Russo, M. Bouton, Statistical analysis of spatial variability in unsaturated flow parameters, Water Resour. Res. 28 (7) (1992) 1911–1925, http://dx.doi.org/10.1029/92WR00669, URL http://doi.wiley.com/10.1029/92WR00669.
[27] R.J. Hoeksema, P.K. Kitanidis, Analysis of the spatial structure of properties of selected aquifers, Water Resour. Res. 21 (4) (1985) 563–572, http://dx.doi.org/10.1029/WR021i004p00563, URL http://doi.wiley.com/10.1029/WR021i004p00563.
[28] P. Dostert, Y. Efendiev, T. Hou, W. Luo, Coarse-gradient Langevin algorithms for dynamic data integration and uncertainty quantification, J. Comput. Phys. 217 (1) (2006) 123–142, http://dx.doi.org/10.1016/j.jcp.2006.03.012, URL https://linkinghub.elsevier.com/retrieve/pii/S0021999106001380.

[29] Y.M. Marzouk, H.N. Najm, Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems, J. Comput. Phys. 228 (6) (2009) 1862–1902, http://dx.doi.org/10.1016/j.jcp.2008.11.024, URL https://linkinghub.elsevier.com/retrieve/pii/S0021999108006062.
[30] C. Scarth, S. Adhikari, P.H. Cabral, G.H. Silva, A.P.d. Prado, Random field simulation over curved surfaces: Applications to computational structural mechanics, Comput. Methods Appl. Mech. Engrg. 345 (2019) 283–301, http://dx.doi.org/10.1016/j.cma.2018.10.026, URL https://linkinghub.elsevier.com/retrieve/pii/S0045782518305309.
[31] A. Gelman (Ed.), Bayesian Data Analysis, second ed., in: Texts in Statistical Science, Chapman & Hall/CRC, Boca Raton, Fla, 2004.
[32] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (6) (1953) 1087–1092, http://dx.doi.org/10.1063/1.1699114, URL http://aip.scitation.org/doi/10.1063/1.1699114.
[33] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika (1970) 13.
[34] L. Katafygiotis, K. Zuev, Geometric insight into the challenges of solving high-dimensional reliability problems, Probab. Eng. Mech. 23 (2–3) (2008) 208–218, http://dx.doi.org/10.1016/j.probengmech.2007.12.026, URL https://linkinghub.elsevier.com/retrieve/pii/S0266892007000707.
[35] G.O. Roberts, J.S. Rosenthal, Examples of adaptive MCMC, J. Comput. Graph. Statist. 18 (2) (2009) 349–367, http://dx.doi.org/10.1198/jcgs.2009.06134, URL http://www.tandfonline.com/doi/abs/10.1198/jcgs.2009.06134.
[36] F. Chollet, et al., Keras, 2015, https://github.com/fchollet/keras.
[37] Theano Development Team, Theano: A python framework for fast computation of mathematical expressions, 2016, arXiv e-prints abs/1605.02688. URL http://arxiv.org/abs/1605.02688, 2016.
[38] T.M. Hansen, K.S. Cordua, Efficient Monte Carlo sampling of inverse problems using a neural network-based forward—applied to GPR crosshole traveltime inversion, Geophys. J. Int. 211 (2017) 10.
[39] D. Moghadas, A.A. Behroozmand, A.V. Christiansen, Soil electrical conductivity imaging using a neural network-based forward solver: Applied to large-scale Bayesian electromagnetic inversion, J. Appl. Geophys. 176 (2020) 104012, http://dx.doi.org/10.1016/j.jappgeo.2020.104012, URL https://linkinghub.elsevier.com/retrieve/pii/S0926985120300033.
[40] T. Hastie, R. Tibshirani, J.H. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, second ed., in: Springer Series in Statistics, Springer, New York, NY, 2009.
[41] G. Hinton, N. Srivastava, K. Swersky, Neural Networks for Machine Learning, Lecture 6a: Overview of Mini-Batch Gradient Descent, Coursera, University of Toronto, 2012.
[42] U. Wolff, Monte Carlo errors with less errors, Comput. Phys. Comm. 176 (5) (2007) 383, http://dx.doi.org/10.1016/j.cpc.2006.12.001, URL http://arxiv.org/abs/hep-lat/0306017.
[43] J. Kaipio, E. Somersalo, Statistical inverse problems: Discretization, model reduction and inverse crimes, J. Comput. Appl. Math. 198 (2) (2007) 493–504, http://dx.doi.org/10.1016/j.cam.2005.09.027, URL https://linkinghub.elsevier.com/retrieve/pii/S0377042705007296.
[44] T.M. Hansen, K.S. Cordua, B.H. Jacobsen, K. Mosegaard, Accounting for imperfect forward modeling in geophysical inverse problems — Exemplified for crosshole tomography, Geophysics 79 (3) (2014) 22.
[45] T. Cui, C. Fox, M.J. O'Sullivan, Bayesian calibration of a large-scale geothermal reservoir model by a new adaptive delayed acceptance Metropolis Hastings algorithm: ADAPTIVE DELAYED ACCEPTANCE METROPOLIS-HASTINGS ALGORITHM, Water Resour. Res. 47 (10) (2011) http://dx.doi.org/10.1029/2010WR010352, URL http://doi.wiley.com/10.1029/2010WR010352.
[46] C. Köpke, J. Irving, A.H. Elsheikh, Accounting for model error in Bayesian solutions to hydrogeophysical inverse problems using a local basis approach, Adv. Water Resour. 116 (2018) 195–207, http://dx.doi.org/10.1016/j.advwatres.2017.11.013, URL https://linkinghub.elsevier.com/retrieve/pii/S0309170817308308.
[47] J. Brynjarsdóttir, A. O'Hagan, Learning about physical parameters: The importance of model discrepancy, Inverse Problems 30 (11) (2014) 114007, http://dx.doi.org/10.1088/0266-5611/30/11/114007, URL http://stacks.iop.org/0266-5611/30/i=11/a=114007?key=crossref.7b886360dda7b385609c577ad82450aa.
[48] M.D. McKay, R.J. Beckman, W.J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 21 (2) (1979) 239–245, URL http://www.jstor.org/stable/1268522.
[49] M.D. Morris, T.J. Mitchell, Exploratory designs for computational experiments, J. Statist. Plann. Inference 43 (3) (1995) 381–402, http://dx.doi.org/10.1016/0378-3758(94)00035-T, URL https://linkinghub.elsevier.com/retrieve/pii/037837589400035T.
[50] C. Lajaunie, G. Courrioux, L. Manuel, Foliation fields and 3D cartography in geology: Principles of a method based on potential interpolation, Math. Geol. 29 (4) (1997) 571–584, http://dx.doi.org/10.1007/BF02775087, URL http://link.springer.com/10.1007/BF02775087.
[51] M. de la Varga, A. Schaaf, F. Wellmann, GemPy 1.0: Open-source stochastic geological modeling and inversion, Geosci. Model Dev. 12 (1) (2019) 1–32, http://dx.doi.org/10.5194/gmd-12-1-2019, URL https://www.geosci-model-dev.net/12/1/2019/.