Bayesian surrogate learning in dynamic simulator-based regression problems

Xi Chen [1]   **Mike Hobson** [1]

Abstract

The estimation of unknown values of parameters (or hidden variables, control variables) that characterise a physical system often relies on the comparison of measured data with synthetic data produced by some numerical simulator of the system as the parameter values are varied. This process often encounters two major difficulties: the generation of synthetic data for each considered set of parameter values can be computationally expensive if the system model is complicated; and the exploration of the parameter space can be inefficient and/or incomplete, a typical example being when the exploration becomes trapped in a local optimum of the objection function that characterises the mismatch between the measured and synthetic data. A method to address both these issues is presented, whereby: a surrogate model (or proxy), which emulates the computationally expensive system simulator, is constructed using deep recurrent networks (DRN); and a nested sampling (NS) algorithm is employed to perform efficient and robust exploration of the parameter space. The analysis is performed in a Bayesian context, in which the samples characterise the full joint posterior distribution of the parameters, from which parameter estimates and uncertainties are easily derived. The proposed approach is compared with conventional methods in some numerical examples, for which the results demonstrate that one can accelerate the parameter estimation process by at least an order of magnitude.

1. Introduction

Complicated dynamical systems are often modelled using computer-based simulators, which typically depend on a number of hidden or control variables $\boldsymbol{\theta}$ that can have a considerable influence on the simulator outputs. The true values $\boldsymbol{\theta}_*$ of these hidden variables are often unknown *a priori*, but are usually of considerable interest. These values must therefore be estimated, typically by comparing measured data (or observations) $\mathcal{D}_\mathrm{o}$ from the real physical system with deterministic (noise-free) synthetic data $\mathcal{D}_\mathrm{s}(\boldsymbol{\theta})$ produced by the simulator as the values of the control variables are varied. Nowadays, this approach is widely used across various sectors including geophysical history matching for reservoir modelling (Das et al., 2017), earth system modelling for climate forecasting (Palmer, 2012), and cancer modelling and simulation for medical diagnostics (Preziosi, 2003).

The approach does, however, suffer from two main drawbacks. First, the generation of the synthetic data by the simulator is often computationally intensive. For instance, an oil/gas reservoir simulator can require several days using High Performance Computing (HPC) resources to produce synthetic data for just a single set of parameter values $\boldsymbol{\theta}$ (Casciano et al., 2015). This clearly makes any exploration of the parameter space very slow, particularly for spaces of moderate to high dimensionality, for which one may require hundreds or thousands of simulator runs. Second, the method used to navigate the space of parameters $\boldsymbol{\theta}$ may be inefficient and/or lead to incomplete exploration of the space. The parameter estimates $\hat{\boldsymbol{\theta}}$ are usually obtained by considering some objective function $F(\boldsymbol{\theta})$ which often exhibit thin degeneracies and/or multiple modes (optima) in the parameter space. For systems where the simulator depends on more than just a few parameters, some standard iterative optimisation algorithm is usually employed (which may or may not require derivatives of the objective function). Such methods typically trace out some path in the parameter space as they converge to the nearest local optimum, rather than the global optimum, and often have difficulty navigating thin degeneracies. In addition, these method provide only a point estimate $\hat{\boldsymbol{\theta}}$ of the parameter values, without any quantification of their uncertainty.

We now outline briefly how the two main drawbacks described above may be circumvented, by the use of model surrogates in combination with a Bayesian approach in which samples are obtained from the posterior distribution of the parameters. We also discuss how such an approach can take advantage of modern parallel computing architectures.

[1]Cavendish Laboratory, University of Cambridge, Cambridge, UK. Correspondence to: Xi Chen <xc253@cam.ac.uk>.

1.1. Surrogate models

A system simulator with high computational cost often results in the fact that the process of estimating the hidden parameters, which requires many such evaluations, becomes infeasible in terms of the total runtime required. One potential approach to addressing this issue is simply to exploit large-scale parallel computing resources. Indeed, the rapid progress of GPU hardware development in recent years has made this an attractive prospect. Nonetheless, there still remains the unresolved issues of synchronisation and communication time between CPU/GPU cores, which becomes non-trivial when the number of cores is large. Moreover, parallel computing can only reduce runtime for repeated parallelable sub-tasks, but cannot accelerate individual sub-tasks, or sequential tasks. In particular, dynamical simulators cannot take full advantage of parallel computing because their inherently sequential execution behaviour.

We therefore address the issue of computationally intensive simulators by constructing a more efficient surrogate model (or emulator, proxy, meta-model, etc.), which mimics the behaviour of the simulator as closely as possible. This approach is also known as black-box modelling or behavioural modelling since the inner mechanism of the simulator is not assumed to be known. We will focus particularly on surrogate modelling for simulators of dynamical physical systems in regression problem, which typically requires the control variables $\boldsymbol{\theta}$ as input only at its initial time step, and then evolves to generate sequential outputs at subsequent time steps. The use of surrogates has the potential to reduce runtime by several orders of magnitude, and naturally accommodates sequential tasks. The main issue with this approach is that the accuracy of the emulator, which depends crucially on the training data and training algorithm. In particular, emulating a very complex simulator often requires a large number of training samples generated from the simulator. Thus, although the runtime for the trained emulator is short, this must be offset against the potentially long time required to prepare the training data and perform the training. Nonetheless, one can mitigate these training costs by taking advantages of parallel computing in some sub-tasks within both training processes.

Over the past two decades, a number of methods have been proposed for simulator-based surrogate construction. These include Gaussian processes (GP, also known as Kriging) (Conti et al., 2009; Conti & O'Hagan, 2010; Hung et al., 2015; Mohammadi et al., 2018), neural networks (NN) (van der Merwe et al., 2007; Melo et al., 2014; Tripathy & Bilionis, 2018; Schmitz & Embrechts, 2018), regression based methods and radial basis function (RBF) methods (Chen et al., 2006; Forrester & Keane, 2009). Of these methods, GP related approaches gain a growing popularity in recent years, but could be computationally infeasible for problems with high dimensionality or with large training data size. NN related methods have shown a good potential, most of its published work, however, targeted non-dynamical problems. Indeed, relatively little work has been performed in applying NN to the construction of surrogates for dynamical simulator-based regression problem. Nonetheless, the recurrent NN (RNN) (Goodfellow et al., 2016) approach has been explored in this context (van der Merwe et al., 2007; Schmitz & Embrechts, 2018). In particular, (van der Merwe et al., 2007), integrated PCA into the approach to reduce dimensionality across time steps and an RNN was used to predict states in PCA derived space.

Although RNN has proven powerful in many sequence modelling problems (e.g., speech recognition and natural language processing tasks), we have found that it performs poorly in (more general) multivariate dynamical regression problems. This is because such problems often contain a considerable number of hidden (or latent) variables, temporal correlated outputs, and the relationship between the variables and outputs is highly non-linear functions. It is then difficult to generate a corpus of training data that is sufficiently descriptive of the variation of the outputs as a function of the hidden variables. In this paper, we therefore adopt the alternative approach of designing a deep recurrent networks (DRN) under the Bayesian inverse problem framework to address the above issues, the details of which will be discussed in the next section.

1.2. Bayesian sampler for inverse problem

Bayesian inference (see e.g. MacKay 2003) seeks to determine the posterior probability distribution of a set of unknown parameters $\boldsymbol{\theta}$ in some model $\mathcal{M}$, given a set of observational data $\mathcal{D}_o$. This is performed by applying Bayes' theorem:

$$\Pr(\boldsymbol{\theta}|\mathcal{D}_o, \mathcal{M}) = \frac{\Pr(\mathcal{D}_o|\boldsymbol{\theta}, \mathcal{M})\Pr(\boldsymbol{\theta}|\mathcal{M})}{\Pr(\mathcal{D}_o|\mathcal{M})}, \quad (1)$$

where $\Pr(\boldsymbol{\theta}|\mathcal{D}_o, \mathcal{M}) \equiv \mathcal{P}(\boldsymbol{\theta})$ is the `posterior` probability density, $\Pr(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}) \equiv \mathcal{L}(\boldsymbol{\theta})$ is the `likelihood` probability density, $\Pr(\boldsymbol{\theta}|\mathcal{M}) \equiv \pi(\boldsymbol{\theta})$ is the `prior` probability density, and $\Pr(\mathcal{D}|\mathcal{M}) \equiv \mathcal{E}$ is the `evidence` (or marginal likelihood). Since $\mathcal{E}$ is independent of $\boldsymbol{\theta}$, one has

$$\mathcal{P}(\boldsymbol{\theta}) \propto \mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta}), \quad (2)$$

showing that the posterior is proportional to the product of likelihood and prior. In the context of estimating the parameters $\boldsymbol{\theta}$ of a simulator-based model $\mathcal{M}$ for some physical system, the likelihood provides a measure of the misfit between the observed data and the synthetic data produced by the simulator (or a proxy thereof), as a function of the parameters. This is used to update our prior belief for the parameter values to yield their posterior distribution. In

practical scenarios, this typically performed by obtaining samples from the posterior distribution using Monte Carlo numerical methods (Robert, 2004).

1.2.1. NESTED SAMPLING

This paper adopts Nested sampling (NS) (Skilling, 2006) as our primary Bayesian sampler, it is a sequential sampling approach that resolves many of the problems encountered by Markov Chain Monte Carlo (MCMC) by evolving a fixed number of points in the space to explore the posterior distribution in a different way. In addition, NS simultaneously provides posterior samples and an estimate of the evidence. Nested sampling algorithm is briefly described in pseudo-code given below.

Algorithm 1: Nested sampling algorithm

   // Initialisation

1 At iteration $i = 0$, draw N_{live} samples $\{\boldsymbol{\theta}_n\}_{n=1}^{N_{\text{live}}}$ from prior $\pi(\boldsymbol{\theta})$ within prior space Ψ.

2 Initialise evidence $Z = 0$ and prior volume $X_0 = 1$.

   // Sampling iterations

3 **for** $i = 1, 2, \cdots, I$ **do**

4   • Compute likelihood $\mathcal{L}(\boldsymbol{\theta}_n)$ for all N_{live} samples.

5   • Find lowest likelihood in live sample and save as $\mathcal{L}_i$.

6   • Calculate weight $w_i = \frac{1}{2}(X_{i-1} - X_{i+1})$, where the prior volume $X_i = \exp(-i/N_{\text{live}})$.

7   • Increment evidence Z by $\mathcal{L}_i w_i$.

8   • Replace the individual sample with likelihood $\mathcal{L}_i$ by a newly drawn sample from restricted prior space Ψ_i such that $\boldsymbol{\theta} \in \Psi_i$ satisfies $\mathcal{L}(\boldsymbol{\theta}) > \mathcal{L}_i$.

9   • If $\max\{\mathcal{L}(\boldsymbol{\theta}_n)\}X_i < \exp(\texttt{tol})Z$,then **stop.**

10 **end for**

11 Increment Z by $\sum_{n=1}^{N_{\text{live}}} \mathcal{L}(\boldsymbol{\theta}_n)X_I/N_{\text{live}}$.

12 Assign the sample replaced at iteration i the importance weight $p_i = L_i w_i / Z$.

Ψ denotes parameter space of $\boldsymbol{\theta}$. I represents total number of iterations, and its value depends on both the pre-defined convergence criteria and the complexity of the problem. X_0 is the prior volume at iteration 0, and X_i denotes the constrained prior volume at ith iteration.

Some widely used NS variants, such as MultiNest (Feroz et al., 2009; 2013) and PolyChord (Handley et al., 2015) have been developed in recent years. In MultiNest, new samples are drawn by rejection sampling at each iteration, from within a multi-ellipsoid bound to an iso-likelihood surface; the bound is computed by samples present at each iteration. It is worth noting that these NS methods can make use of parallel computing resources within the sampling

algorithm.

1.3. The proposed approach

We develop a deep recurrent network (DRN) technique to train surrogate models for multivariate dynamical simulator-based system, which is then used to perform a fast calculation of synthetic data $\mathcal{D}_{\text{s}}(\boldsymbol{\theta})$ from a given set of control parameters $\boldsymbol{\theta}$. $\mathcal{D}_{\text{s}}(\boldsymbol{\theta})$ are then used alongside measured data $\mathcal{D}_{\text{o}}$ from the real physical to provide a rapid evaluation of the likelihood function within a Bayesian analysis. Using this fast likelihood function, the parameter space $\boldsymbol{\theta}$ is explored using NS (via the MultiNest algorithm) to obtain samples from the posterior distribution and evidences.

The DRN employs a temporal cascading structure that is used to model temporal dynamics in simulator. This structure divides the system outputs into temporal features, and constructs a series of cascaded components to form a surrogate model. The proposed approach greatly facilitates the training process, and leads to a runtime at least an order of magnitude smaller than that of the simulator. Moreover, unlike traditional proxy building methods that use random samples for training, in our approach the surrogate model is trained using samples generated by the nested sampling algorithm. This assists in training by providing higher sample density in the high-likelihood region of interests, and thus improves the overall accuracy with which the parameters can be estimated.

The paper is organised as follows. Section 2 introduces and formulates the problems. Section 3 details the design of DRN. Section 4 depicts a complete pipeline of the proposed Bayesian surrogate modelling approach. Section 5 presents numerical results. Section 6 concludes the work and discusses some future directions.

2. Problem formulation

Consider a simulator S with J input and M output features. The simulator only takes inputs $\boldsymbol{\theta}$ at time $t = 0$, and evolves for T steps. The total output number is $T \times M$, and the noise-free simulator execution process can be written as:

$$\mathbf{Z}_{\boldsymbol{\theta}} = S(\boldsymbol{\theta}), \tag{3}$$

where input vector $\boldsymbol{\theta} \in \mathbb{R}^{J \times 1}$, with $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_j, \cdots, \theta_J]^\mathsf{T}$, and $\mathbf{Z}_{\boldsymbol{\theta}} \in \mathbb{R}^{T \times M}$ is the simulator output matrix corresponding to $\boldsymbol{\theta}$, with $\mathbf{Z}_{\boldsymbol{\theta}} = [\mathbf{z}_1, \cdots, \mathbf{z}_t, \cdots, \mathbf{z}_T]^\mathsf{T}$. The column vector $\mathbf{z}_t$ contains M output features at time t, and $\mathbf{z}_t = [z_{t,1}, \cdots, z_{t,m}, \cdots, z_{t,M}]^\mathsf{T}$. The sequential execution process is shown in Figure 1. S_t denotes an inner component of S producing $\mathbf{z}_t$ at time t.

In practice, the simulator output $\mathbf{Z}_{\boldsymbol{\theta}}$ may not correspond to noise-free synthetic data. We denote the synthetic data
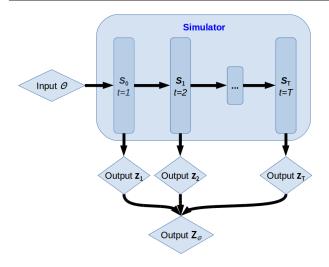
Figure 1. Sequential execution in a simulator. The rectangle blocks $\{S_1, S_2, \cdots, S_T\}$ represent simulator components at times t.

by the matrix $\mathbf{D}_{\boldsymbol{\theta}}$ (previously $\mathcal{D}_{\mathrm{s}}(\boldsymbol{\theta})$) with the same dimensions and structure as $\mathbf{Z}_{\boldsymbol{\theta}}$, then the process is expressed as $\mathbf{D}_{\boldsymbol{\theta}} = \mathbf{R}(\mathbf{Z}_{\boldsymbol{\theta}})$. $\mathbf{R}$ describes the response of the measurement system used, which may be a complicated non-linear function of its argument. In this paper we will assume, however, that the response function of the measurement process is simply the identity, so that $\mathbf{D}_{\boldsymbol{\theta}}$ is equal to $\mathbf{Z}_{\boldsymbol{\theta}}$.

The complete inference of a set of unknown parameters $\boldsymbol{\theta}$ from an observed data set $\mathbf{D}_o$ is contained in the posterior distribution $\mathrm{Pr}(\boldsymbol{\theta}|\mathbf{D}_o)$, which can be described by a set of discrete samples $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^I$ with their weights (posterior probability) computed by Equation (2):

$$\mathrm{Pr}(\boldsymbol{\theta}|\mathbf{D}_o) \sim \{\boldsymbol{\theta}^{(i)}; \mathrm{Pr}(\mathbf{D}_o|\boldsymbol{\theta}^{(i)})\,\mathrm{Pr}(\boldsymbol{\theta}^{(i)})\}_{i=1}^I. \quad (4)$$

The prior is assumed as a Gaussian with $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu_\theta}, \boldsymbol{\Sigma_\theta})$. We also assume a Gaussian additive noise (non-Gaussian noise model case will be presented in an extended version paper) $\boldsymbol{\xi}$ to describe non-informative part of the observation, such that $\xi_{t,m} \sim \mathcal{N}(0, (\sigma_{t,m}^\xi)^2)$ for the mth feature at time t. Sample likelihood $\mathrm{Pr}(\mathbf{D}_o|\boldsymbol{\theta}^{(i)})$ is given by:

$$\mathcal{L}(\boldsymbol{\theta}^{(i)}) = \prod_{t=1}^T \prod_{m=1}^M \left\{ \frac{1}{\sqrt{2\pi(\sigma_{t,m}^\xi)^2}} \exp\left[-\frac{(d_{t,o} - z_{t,m}^{(i)})^2}{2(\sigma_{t,m}^\xi)^2} \right] \right\}, \quad (5)$$

where $\mathbf{D}_o = [d_{1,o}, \cdots, d_{t,o}, \cdots, d_{T,o}]$ is the observed data, and $z_{t,m}^{(i)}$ is the simulator prediction for sample $\boldsymbol{\theta}^{(i)}$. The posterior $\mathrm{Pr}(\boldsymbol{\theta}|\mathbf{D}_o)$ can then be calculated by Equation (4).

3. DRN surrogate construction

Surrogate model H is used to emulate the simulator described in Equation (3), and is written as:

$$\hat{\mathbf{Z}}_{\boldsymbol{\theta}} = H(\boldsymbol{\theta}), \quad (6)$$

where $\hat{\mathbf{Z}}_{\boldsymbol{\theta}}$ is the surrogate output. The goal is to construct an H that minimises the misfit between $\hat{\mathbf{Z}}_{\boldsymbol{\theta}}^{(i)}$ and $\mathbf{Z}_{\boldsymbol{\theta}}^{(i)}$ given training set $\{\boldsymbol{\theta}^{(i)}, \mathbf{Z}_{\boldsymbol{\theta}}^{(i)}\}_{i=1}^I,$. The probability of a single training pair $\mathrm{Pr}(\mathbf{Z}_{\boldsymbol{\theta}}^{(i)}|\boldsymbol{\theta}^{(i)})$ can be decomposed as:

$$\log \mathrm{Pr}(\mathbf{Z}_{\boldsymbol{\theta}}^{(i)}|\boldsymbol{\theta}^{(i)}) = \sum_t \log(\mathbf{z}_t^{(i)}|\mathbf{z}_{t-1}^{(i)}, \boldsymbol{\theta}^{(i)}), \quad (7)$$

where $\mathbf{z}_t^{(i)}$ is a sequence element in $\mathbf{Z}_{\boldsymbol{\theta}}^{(i)}$ with

$$\mathbf{z}_t^{(i)} = \begin{cases} H_t(\boldsymbol{\theta}^{(i)}), & \text{if } t = 1, \\ H_t(\boldsymbol{\theta}^{(i)}, \mathbf{z}_{t-1}^{(i)}), & \text{otherwise.} \end{cases} \quad (8)$$

H is also composed by a set of functional sequence components $\{H_t\}_{t=1}^T$. We define model hyper-parameters as Φ, thus training a surrogate is equivalent to maximising the log-likelihood w.r.t. hyper-parameter set Φ:

$$\Phi^* = \arg\max_{\Phi} \sum_{i=1}^I \log \mathrm{Pr}(\mathbf{Z}_{\boldsymbol{\theta}}^{(i)}|\boldsymbol{\theta}^{(i)}; \Phi), \quad (9)$$

where Φ^* denotes the optimal hyper-parameter set.

3.1. Design of DRN

As illustrated in Figure 2, the proposed DRN is one of the typical designs (Goodfellow et al., 2016) whose only recurrence is from the prediction $\hat{\mathbf{z}}_t$ to its next hidden component H_{t+1}. H_t in Figure 2 is extended as an independent DNN (conditioned on its previous prediction) to learn features at each time step. A pre-configured DNN (detailed in the next section) is adopted as a component-wise ML training algorithm within the proposed DRN structure. This paper mainly focuses on introducing a complete Bayesian surrogate modelling approach, topics such as DNN structure design, hyper-parameter tuning, DRN design with LSTM, or scheduled sampling (Bengio et al., 2015) is beyond the scope of this paper.

3.2. DNN hidden component

As shown in Figure 3, we employ a DNN structure with 4 hidden layers and 2 Dropout layers. $H_t^{(b)}$ denotes bth hidden layer, and the number of nodes in each hidden layer depends on the number of inputs J, the number of features M, and complexity factor η. Particularly, node number in each DNN layer is determined as follows:
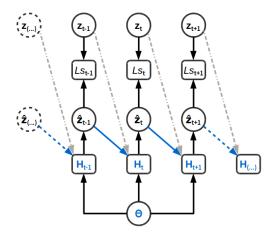
- Input layer: $J + M$.

Figure 2. A DRN illustrated in unfolded computational graph. Rectangle block represents function and round block represents variable. Model output is denoted by $\{\hat{\mathbf{z}}_t\}_{t=1}^T$. Loss function in DRN at time t is Ls_t, which computes the error between $\hat{\mathbf{Z}}_{\boldsymbol{\theta}}$ and the training sample $\mathbf{Z}_{\boldsymbol{\theta}}$. Features of the proposed DRN are highlighted in blue dashed line: (1) the input $\boldsymbol{\theta}$ is fixed in our problem. (2) the hidden component H_t contains an extended DNN structure in DRN. (3) predicted sample $\hat{\mathbf{z}}_t$ (solid blue line), rather than training sample $\mathbf{z}_t$ (dash-doted grey line) is used to feed its next hidden component H_{t+1}.

- $H_t^{(1)}$: $\eta \times (J + M)$.

- $H_t^{(2)}$: $4 \times \eta \times (J + M)$.

- $H_t^{(3)}$: $4 \times \eta \times (J + M)$.

- $H_t^{(4)}$: $\eta \times M$.

- Output layer: M.

Activation function used in the proposed DNN includes 2 linear functions, 2 Parametric Rectified Linear Unit (PReLU) functions, and 1 Hyperbolic tangent (Tanh) function. In the numerical examples, the proposed DNN structure and the majority of its hyper-parameters are fixed. Node number in the hidden layers can be adjusted through the complexity factor η. We adopt standard mean square error (MSE) as the loss function in DRN training, and use standard *Adam* algorithm (which is a first-order gradient-based stochastic optimisation method) as the optimiser. The Dropout ratio is set to 0.5 for the Dropout layers.

Although the proposed DRN structure can be generalised to different problems, constructing a good DNN architecture still highly depends on the complexity and dimensionality of the target problem and simulator. DNN structure needs to be carefully designed with specific purposes by examining a series of properties such as loss function, activation function, optimiser, layer number, node number, and evaluation metrics, etc.. The structure illustrated in Figure 3 is one of our design choices that fits well in the numerical examples.
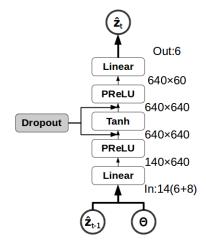

Figure 3. A example to illustrate component-wise DNN structure. The model complexity factor η is set to 10 in this example. Component H_t has 14 inputs and 6 outputs, and contains 4 forward hidden layers and 2 Dropout layers. The bottom two circle blocks depict input to the DNN, including $\boldsymbol{\theta}$ (with 8 input elements) and $\hat{\mathbf{z}}_{t-1}$ (with 6 predictions). The rectangle blocks denote activation functions between different layers. The numbers on the right of the rectangle blocks represent node number of different DNN layer.

4. Bayesian surrogate learning

The pipeline of a complete Bayesian surrogate learning process includes three phases: (1) sampling from simulator; (2) surrogate model construction; and (3) sampling from surrogate model. Details are described as follows and also illustrated in supplementary material Section Figure **??**.

4.1. Sampling from simulator

The goal in Phase 1 is to generate posterior samples for surrogate training through Bayesian sampling techniques performed in the original computationally intensive simulator S. MultiNest firstly draws $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^I$ random samples from prior $\pi(\boldsymbol{\theta})$. Simulator outputs $\mathbf{Z}_{\boldsymbol{\theta}}$ and its corresponding likelihood $\mathcal{L}(\boldsymbol{\theta})$ can then be computed through Equation (3) and (5), respectively. One can obtain posterior samples $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^I$ and their weights $\Pr(\boldsymbol{\theta}|\mathbf{D}_o)$ of $\boldsymbol{\theta}$ by Equation (4).

In practice, it is often time consuming to use posterior samples for training. Collecting Bayesian posterior samples requires sampling algorithms to perform a considerable (but unknown) number of likelihood evaluations. An alternative way is to simply draw random samples from the prior. Prior can be in uniform, Gaussian, or Latin-Hyper-Cube (LHC) (McKay et al., 1979) etc, and this method only requires exactly I likelihood evaluations. One key drawback of random prior drawn, however, relies on its flat sample densities in both the high and low likelihood regions, which is clearly less efficient than the posterior drawn approach when exploring high likelihood regions.

4.2. Surrogate model training

The obtained posterior samples $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^{I}$ are then fed as input to simulator S to generate $\mathbf{Z}_{\boldsymbol{\theta}}$. The training set $\{\boldsymbol{\theta}^{(i)}, \mathbf{Z}_{\boldsymbol{\theta}}^{(i)}\}_{i=1}^{I}$ can then be collected for surrogate training for the proposed DRN described in Section 3.

Bayesian posterior samples often concentrate in high likelihood regions, in which they can provide high resolution for surrogate learning. However, this also leads to its poor generalisation in the whole state space. On the contrary, random drawn samples can well generalised in the state space, but often require a relatively bigger sample size to gain enough resolution for the region of interests. This issue is further discussed and illustrated in a numerical example.

4.3. Sampling from surrogate model

Once the surrogate model H is obtained, it can be used for fast parameter calibration following a similar procedure as described in Phase 1. In fact, the surrogate acts as a fast forward mapping between $\boldsymbol{\theta}$ and $\hat{\mathbf{Z}}_{\boldsymbol{\theta}}$, and the DRN is used to interpolate the contour space based on the training data. One of the constrains for this approach is in training data collection step. Even with random drawn samples, the approach still requires simulator S to execute I times for training data collection. A small I can lead to poor generalisation, while a big I results in a long data preparation time. The choice of I varies case-by-case, and a trade-off should be carefully considered in different practical cases to balance estimation accuracy and total runtime.

5. Numerical examples

This section reports numerical performance of the proposed method. It is implemented with Keras (TensorFlow backend) (Chollet et al., 2015) in Python 3.5 environment. In particular, we compare: (a) the effect of using different training sample sources including Latin Hyper-Cube (LHC) samples (McKay et al., 1979), MultiNest posterior samples, and a mix of them. (b)Performance between the proposed DRN, a standard DNN, and a standard RNN. (c) Total runtime between simulator-based method and surrogate method using a same sampler. In the tests, training process is executed by an NVIDIA Quadro K2200 GPU with 640 CUDA cores and 4 GB graphical RAM. MultiNest estimation is performed in a computer equipped by an Intel Xeon E3-1246 v3 CPU with 4 cores (3.5 GHz) and 16 GB RAM.

5.1. Bivariate dynamic model

A synthetic 2 inputs and 10 outputs toy simulator is constructed to benchmark the algorithm performance. It has $J = 2$ unknown parameters, and $M = 1$ output feature for $T = 10$ time steps (thus 10 outputs in total). The simulator
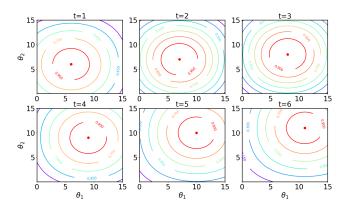

Figure 4. Illustration of temporal dynamics of simulator output in the bivariate example.The sub-figures depict contour from time $t = 1$ to 6. The value inline of contour denotes output $\mathbf{z}_t$ of that contour circle, and the peak is highlighted in red dot at centre.

function is defined as:

$$\mathbf{z}_t = \cos[\phi(\theta_1 - t - \eta)] \cos[\phi(\theta_2 - t - \eta)], \qquad (10)$$

where input $\boldsymbol{\theta} = [\theta_1, \theta_2]$, $\mathbf{z}_t$ denotes simulator output at time t, and $\mathbf{z}_t = z_t$ in this example. ϕ and η are known constant coefficients, with $\phi = 0.1$, and $\eta = 5$. Prior range is set to $\boldsymbol{\theta} \in [0, 15]$, and the ground truth $\boldsymbol{\theta}_* = [\theta_{*1}, \theta_{*2}]$ are $\theta_{*1} = \theta_{*2} = 10$. It is a uni-mode toy simulator, of which its highest output (the peak) gradually moves toward north-east direction in the 2D parameter space whereas the contour shape remains the same, as shown in Figure 4.

5.1.1. CHOICE OF TRAINING SAMPLE SOURCE

Industrial system simulator often contains a number of non-linear functions, which may result in highly complex contour in parameter space. A suitable training sample collection is then desired in this situation to improve surrogate model accuracy. Here we compare 3 different training sample sets:

1. Latin Hyper-Cube (LHC) samples.

2. Posterior samples from MultiNest algorithm.

3. An equal mix of MultiNest and LHC samples.

The number of training sample N is set to 2000, and for case (3) samples are composed by $N/2$ LHC samples and $N/2$ MultiNest posterior samples. Other settings are fixed: the Gaussian noise standard deviation $\sigma_{t,m}^{\xi}$ is set to 5% of the averaged noise-free simulator output $\mathbf{Z}_{\boldsymbol{\theta}}$, and the prior of $\boldsymbol{\theta}$ is a uniform distribution with range $[0, 15]$. DRN is trained with 300 epochs and its mini-batch size is 20. A 10-fold cross-validation (CV) method is used for training, i.e., the ratio between training and testing data is $9 : 1$.
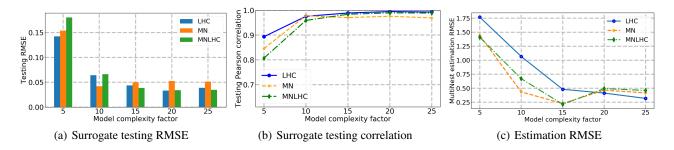
(a) Surrogate testing RMSE   (b) Surrogate testing correlation   (c) Estimation RMSE

Figure 5. Accuracy comparison of DRN surrogate with respect to model complexity η in different training sample sources. A bigger value of η denotes a more complex DNN structure in the proposed DRN. LHC: Latin Hyper Cube; MN: MultiNest posterior; MNLHC: an equal mix of MultiNest and LHC samples. Surrogate model accuracy is evaluated in RMSE (sub-figure (a)), and Pearson correlation (sub-figures (b)), both using out of sample testing data set. All results are evaluated with 10-folds cross-validation. Sub-figure (c) shows final estimation accuracy of $\boldsymbol{\theta}$ (in RMSE) using corresponding trained surrogate in MultiNest.

Figure 5 (a) and (b) shows accuracy comparison of different trained surrogate in test data. One sees that the surrogate accuracy (represented in root mean squared error (RMSE) and Pearson Correlation) is affected by different training data collection schemes. Both RMSE and correlation are calculated between the true training outputs and the surrogate predicted outputs. Surrogate accuracy stays at a robust level when $\eta \geq 15$. In sub-figure (b) all surrogate models score high correlation, i.e, $> 95\%$ when $\eta \geq 10$. The results demonstrate that the trained models can achieve comparable accuracy for $\eta \geq 15$ for the bivariate example.

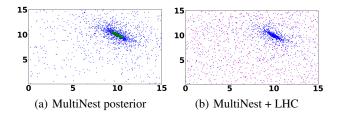
(a) MultiNest posterior   (b) MultiNest + LHC

Figure 6. Illustration of MN (sub-figure (a)) and MNLHC samples (sub-figure (b)). MultiNest samples are denoted in green and blue (darker in grey-scale). LHC samples are denoted in purple (lighter in grey scale).

Figure 5 (c) shows the corresponding MultiNest estimation performance. The best estimation accuracy is achieved by case (2) and (3) samples, both with model complexity $\eta = 15$. This is different from the observation in Figure 5 (a), where the best performance is achieved at $\eta = 20$ by case (1) and (3) samples. The observation suggests that highly accurate surrogate doesn't necessarily guarantee best posterior estimation performance. In fact, the two accuracy indicators (namely, surrogate RMSE and estimation RMSE) will give more consistent performance when the sampling resolution in parameter space is high enough.

The performance shows that training sample bias can af-

fect surrogate model performance. Figure 6 illustrates the difference between MN and LHC sampling schemes. In sub-figure (a), MultiNest initialises random samples (blue dots) across the parameter space, and samples intensively move toward the high likelihood region (green dots). Sub-figure (b) shows a mix of MN and LHC samples (purple dots), where LHC samples help to describe and generalise the whole parameter space for surrogate training. In our empirical tests, an equally mixed MCMC posterior and LHC samples (or uniform samples) also achieves similar results as those in case (3) MNLHC, which suggests that the proposed scheme is suitable for other Bayesian posterior sampling algorithms.

5.2. Industrial example: Black-Oil dynamic model

The second test example employs a widely recognised and studied industrial Black-Oil reservoir simulator in geophysical applications. The simulator contains a number of partial differential equations to model fluid dynamics in a petroleum reservoir. Please see (Odeh et al., 1981) and (Ramirez et al., 2017; Das et al., 2017) if one is interested in details of this simulator and its geophysical background. The tested Black-Oil simulator has 8 inputs in $\boldsymbol{\theta}$ that describes subsurface geophysical properties and 60 outputs (6 features per time step for 10 time steps). Prior of $\boldsymbol{\theta}$ follows uniform distribution, and its prior ranges and ground truths $\boldsymbol{\theta}_*$ are listed in Table 1. The MNLHC scheme is adopted for data preparation. DRN with $\eta = 10$ is used for training with 100 epochs 20 mini-batch size. Other settings are kept the same as those in the previous examples, and the reported accuracy is an averaged over 10 folds CV results. Regarding MultiNest settings, the number of live samples N_{live} is set to 300 with sampling efficiency $\text{erf} = 0.8$ and tolerance $\text{tol} = 0.5$.

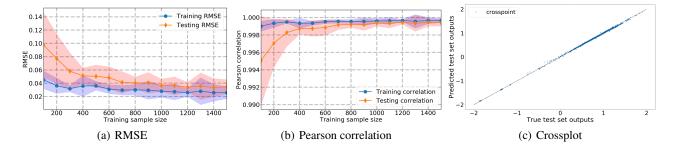| (a) RMSE | (b) Pearson correlation | (c) Crossplot |

Figure 7. Sub-figure (a) and (b) shows RMSE/Correlation learning curve, respectively, in a function of training sample size. The blue/orange (darker/lighter in grey scale) curves denote performance in training/testing data set, respectively. The corresponding shadowed areas show its fluctuation in 2 standard deviations range. Sub-figure (c) shows the cross-plot between surrogate and simulator outputs.

Table 1. List of ground truth $\boldsymbol{\theta}_*$ values. The last two columns indicate upper and lower limits of uniform distribution.

Parameter	Truth θ_*	Lower limit	Upper limit
1	1.21	0.2	5.0
2	0.3	0.2	5.0
3	3.0	0.2	5.0
4	0.26	0.1	1.0
5	0.64	0.1	1.0
6	1.0	0.75	1.25
7	0.8	0.75	1.25
8	1.2	0.75	1.25

5.2.1. DRN SURROGATE ACCURACY

Figure 7 (a) shows the surrogate accuracy in RMSE with training sample size varying between 100 and 1500. Training RMSE stays at the level around 0.032, while testing RMSE stays at around 0.045. Figure 7 (b) shows the corresponding correlation performance, and both training and testing correlations achieve high accuracy ($> 99.8\%$). A cross-plot of surrogate prediction versus simulator true output in out-of-sample test is shown in Figure 7 (c), the performance of which is consistent to the learning curves.

5.2.2. COMPARING WITH EXISTING NETWORK STRUCTURES

In addition to the proposed DRN, surrogate trained by other standard network structures are also implemented and evaluated. Specifically, a fully connected standard DNN with no cascading structure (ncDNN) and a standard RNN without extended hidden layers (stdRNN) are employed. ncDNN adopts a same DNN structure as the DNN component described previously, and its network complexity is set as $\eta = 50$. This non-cascading DNN takes $\boldsymbol{\theta}$ as its input, and generate all 60 outputs simultaneously. stdRNN re-uses the recurrent structure described in Figure 2, but without extended DNN hidden component.

As shown in Figure 8 (a), ncDNN performance is not robust for dynamic regression task, and the RMSE fluctuates with large variance. stdRNN shows a relatively stable RMSE performance, but its accuracy is much worse than that of DRN with 50 epochs. DRN's better performance is in a price of training time. As shown in Figure 8 (b), DRN training time is clearly higher than the other two networks. Nevertheless, one sees that the training time increase rate is similar in all three methods, which suggests that DRN is still the best balanced choice for the the Black-Oil problem.

5.2.3. ESTIMATION ACCURACY AND OVERALL RUNTIME

Figure 9 shows MultiNest estimation accuracy with 10% noise in triangle plot. One sees that the peaks of posterior distributions in the triangle plot are very close to the ground-truth black dashed lines, and the coloured contour (red and blue areas) are almost overlapping with each other. All these results suggest the DRN trained surrogate model is highly consistent with the original simulator, and MultiNest can perform accurate posterior estimation through the trained surrogate model. Table 2 shows a comparison of approximated runtime between surrogate model and simulator. The training time is collected based on DRN structure with $\eta = 15$, epoch $= 100$, and 1500 training data points. As shown in the second column of the table, the number of likelihood evaluations in both methods are comparable. The table clearly demonstrates that the proposed surrogate method can bring down MultiNest execution time for at least an order of magnitude.

Figure 8 (c) shows an approximation of total runtime in a function of repeated trail number. It demonstrates that the proposed method can save computational time (compare to the original simulator method) when the underlying example requires more than one repeated realisation. Note that this conclusion is based on specific settings and problems, the runtime can be affected by a number of factors including but not limit to: number of CPU/GPU cores, MultiNest hyper-parameter settings, and training sample size.
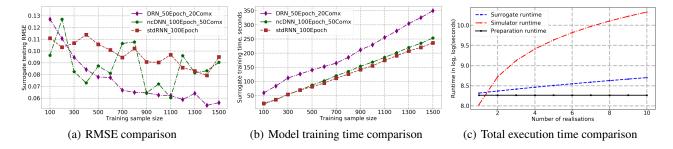
(a) RMSE comparison (b) Model training time comparison (c) Total execution time comparison

Figure 8. Sub-figure (a) and (b) shows comparison between different network structures in a function of training sample size. DRN_50Epoch_20Comx' denotes DRN (purple curve/diamond marker) with $\eta = 20$ and 50 epochs. 'ncDNN' denotes the non-cascading standard DNN structure, and 'stdRNN' represents standard RNN with single hidden layer. Sub-figure (a) shows surrogate accuracy in RMSE, and sub-figure (b) shows model training time. Sub-figure (c) shows the total execution time comparison between the proposed surrogate and original simulation in a function of repeated realisations number (i.e., the number of repeats needed to re-run a task). The runtime is in log-scale, and the red dash-doted / blue dashed curves denote MultiNest estimation time with simulator/surrogate, respectively. The black flat line denotes a summation of data preparation time and surrogate training time per realisation.
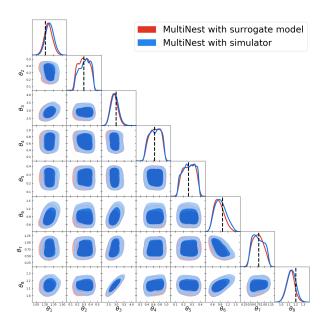

Figure 9. Triangle plot of posterior estimation in 10% noise. The diagonal blocks show posterior distributions of $\boldsymbol{\theta}$. Curves and regions in red (darker) and blue (lighter) denote estimated posterior with DRN surrogate model and original simulator, respectively. The black dashed lines denote ground-truth of $\boldsymbol{\theta}$.

6. Conclusions and discussions

This paper introduced a complete Bayesian surrogate learning methodology for fast parameter estimation in dynamic simulator-based regression problems. The proposed method has some limitations. Firstly, it is powerful for scenarios that require some, rather than one, simulator runs. Surrogate approach doesn't take advantage if only very few simulator runs are needed during a fairly long period of time (e.g. 3 times per year). Secondly, the proposed approach is not suit-

Table 2. Runtime comparison between MultiNest with surrogate model versus original simulator. 'Avg. numE': averaged likelihood evaluation number. 'prepT': posterior data collection time. 'trainT' and 'execT' are surrogate training time and MultiNest execution time, respectively, and 'sec.' denotes second.

Sampler	Avg. numE	prepT	trainT	execT (sec.)
Surrogate	12568	3062	819	213
Simulator	13345	0	0	3062

able for dynamic simulators that has large number of time steps, this is because each time step needs a separate DNN component training within the DRN. The proposed pipeline is flexible in changing its sampling and training algorithm components. For instance, the NS algorithm can be replaced by other sampling algorithms such as MCMC (For readers interested in a comparison of MCMC and MultiNest, please see (Chen et al., 2018) for more details), and the proposed DRN can also be replaced by other ML algorithms such as Gaussian Process, depending on specific needs. Since it is an emulator to a system simulator, there's no need to re-train a surrogate when new observation data becomes available.

Acknowledgements

References

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015. 4

Casciano, C., Cominelli, A., Bianchi, M., et al. Latest Advances In Simulation Technology For High-resolution Reservoir Models: Achievements And Opportunities For Improvement. In *SPE Reservoir Characterisation and Simulation Conference and Exhibition*, 2015. 1

Chen, V., Tsui, K., Barton, R., and Meckesheimer, M. A review on design, modeling and applications of computer experiments. *IIE transactions*, 38(4):273–291, 2006. 2

Chen, X., Hobson, M., Das, S., and Gelderblom, P. Improving the efficiency and robustness of nested sampling using posterior repartitioning. *arXiv preprint arXiv:1803.06387*, 2018. 9

Chollet, F. et al. Keras. https://github.com/fchollet/keras, 2015. 6

Conti, S. and O'Hagan, A. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of statistical planning and inference*, 140(3):640–651, 2010. 2

Conti, S., Gosling, J. P., Oakley, J. E., and O'Hagan, A. Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676, 2009. 2

Das, S., Chen, X., and Hobson, M. Fast gpu-based seismogram simulation from microseismic events in marine environments using heterogeneous velocity models. *IEEE Transactions on Computational Imaging*, 3(2):316–329, 2017. 1, 7

Feroz, F., Hobson, M., and Bridges, M. MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Monthly Notices of the Royal Astronomical Society*, 398(4):1601–1614, 2009. 3

Feroz, F., Hobson, M., Cameron, E., and Pettitt, A. Importance nested sampling and the MultiNest algorithm. *arXiv preprint arXiv:1306.2144*, 2013. 3

Forrester, A. and Keane, A. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45 (1-3):50–79, 2009. 2

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*, volume 1. MIT press Cambridge, 2016. 2, 4

Handley, W., Hobson, M., and Lasenby, A. POLYCHORD: next-generation nested sampling. *Monthly Notices of the Royal Astronomical Society*, 453(4):4384–4398, 2015. 3

Hung, Y., Joseph, V. R., and Melkote, S. N. Analysis of computer experiments with functional response. *Technometrics*, 57(1):35–44, 2015. 2

MacKay, D. *Information theory, inference and learning algorithms*. Cambridge university press, 2003. 2

McKay, M., Beckman, R., and Conover, W. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979. 5, 6

Melo, A., Cóstola, D., Lamberts, R., and Hensen, J. Development of surrogate models using artificial neural network for building shell energy labelling. *Energy Policy*, 69:457–466, 2014. 2

Mohammadi, H., Challenor, P., and Goodfellow, M. Emulating dynamic non-linear simulators using gaussian processes. *arXiv preprint arXiv:1802.07575*, 2018. 2

Odeh, A. S. et al. Comparison of solutions to a three-dimensional black-oil reservoir simulation problem (includes associated paper 9741). *Journal of Petroleum Technology*, 33(01):13–25, 1981. 7

Palmer, T. Towards the probabilistic Earth-system simulator: a vision for the future of climate and weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 138(665):841–861, 2012. 1

Preziosi, L. *Cancer modelling and simulation*. CRC Press, 2003. 1

Ramirez, B., Gelderblom, P., Eales, A., Chen, X., Hobson, M., Esler, K., et al. Sampling from the posterior in reservoir simulation. In *SPE Abu Dhabi International Petroleum Exhibition & Conference*, 2017. 7

Robert, C. *Monte Carlo methods*. Wiley Online Library, 2004. 3

Schmitz, T. and Embrechts, J.-J. Real time emulation of parametric guitar tube amplifier with long short term memory neural network. *arXiv preprint arXiv:1804.07145*, 2018. 2

Skilling, J. Nested sampling for general bayesian computation. *Bayesian Analysis*, 1(4):833–859, 2006. 3

Tripathy, R. and Bilionis, I. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *arXiv preprint arXiv:1802.00850*, 2018. 2

van der Merwe, R., Leen, T. K., Lu, Z., Frolov, S., and Baptista, A. M. Fast neural network surrogates for very high dimensional physics-based models in computational oceanography. *Neural Networks*, 20(4):462–478, 2007. 2