Bayesian Emulation of Complex Multi-Output and Dynamic Computer Models

Stefano Conti[*] *Anthony O'Hagan*

Department of Probability and Statistics, The Hicks Building,
University of Sheffield, Sheffield S3 7RH, U.K.

Abstract: Computer models are widely used in scientific research to study and predict the behaviour of complex systems. The run times of computer-intensive simulators are often such that it is impractical to make the thousands of model runs that are conventionally required for sensitivity analysis, uncertainty analysis or calibration. In response to this problem, highly efficient techniques have recently been developed based on a statistical model (the *emulator*) that is built to approximate the computer model. The approach, however, is less straightforward for dynamic simulators, designed to represent time-evolving systems. A generalisation of the established methodology to allow for dynamic emulation is here proposed. Advantages and difficulties are discussed and illustrated with an application to the Sheffield Dynamic Global Vegetation Model, developed within the UK Centre for Terrestrial Carbon Dynamics.

Keywords: Bayesian inference, computer experiments, dynamic models, hierarchical models

1. INTRODUCTION

Large computer codes, implementing sophisticated mathematical models, are widely used in all fields of science and technology to describe and understand complex systems. We refer to any such program as a *simulator*. The size and complexity of a simulator can become a problem when it is necessary to make very many runs at different input values. For example, the model user may wish to study the sensitivity of model outputs to variations in its inputs, which entails many model evaluations when the number of inputs is large (as is very often the case). In particular, standard Monte Carlo-based methods of sensitivity analysis (extensively reviewed by Saltelli et al., 2000) typically require thousands of model runs. Another example is the practice of calibrating model parameters by varying them to fit a set of physical observations. Such explorations can become infeasible even for moderately large computer models requiring just a few seconds per run.

Following Sacks et al. (1989), a two-stage approach based on meta-modelling (*emulation*) of the simulator's response has been developed (see Haylock and O'Hagan, 1996; Kennedy and O'Hagan, 2001; Oakley and O'Hagan, 2002), offering substantial efficiency gains over standard Monte Carlo-based methods. These authors represent the simulator as a function $f(\cdot)$ which takes as input a vector $\boldsymbol{x}$ of inputs and produces an output $y = f(\boldsymbol{x})$. A Bayesian formulation assumes a Gaussian process prior distribution for the function $f(\cdot)$, conditional on various hyperparameters. This prior distribution is updated using as data a preliminary *training sample* $\big\{y_1 = f(\boldsymbol{x}_1), \ldots, y_n = f(\boldsymbol{x}_n)\big\}$ of n selected simulator runs. Formally, the posterior distribution of $f(\cdot)$ is regarded as the emulator. This posterior distribution is also a Gaussian process conditional on the hyperparameters; here conditioning upon the training set forces realisations from the emulator to interpolate the observed data points and induces posterior distributions for the hyperparameters.

The first stage of the two-stage approach is to build the emulator. Problems such as sensitivity analysis or calibration are then tackled in the second stage using the emulator. Since the emulator runs almost instantaneously, the computational cost of this approach lies in obtaining the training runs. Gains in efficiency arise because in practice it is usually possible to emulate the code output to a high degree of precision using only a few hundreds of training runs.

Research hitherto has almost exclusively dealt with emulating a single output of the simulator. However, it is common for simulators to produce many outputs. Often, the collection of outputs has a spatial and/or temporal structure. For instance, the oilfield simulator studied by Craig et al. (1996) outputs

[*] E-mail: s.conti@sheffield.ac.uk

its predictions of the pressure at a given well over time, so that we can view these outputs as a time series. Similarly, the atmospheric dispersion model used by Kennedy et al. (2002) predicts deposition of radioactive particles at points on a spatial grid. Although existing theory for single-output emulation may be used to emulate each output individually, this can be a laborious process and may lose important information about correlations between outputs. The purpose of the present article is to propose a *multi-output* emulator, and to compare it with two other approaches based on single-output emulation.

The single-output Bayesian methodology elaborated by O'Hagan (1992); Oakley and O'Hagan (2002) is extended in Section 2 to enable the simultaneous emulation of a vector of outputs. In particular our interest lies in emulating *dynamic* simulators that model a system evolving over time, thereby producing a time series of outputs. One such model is the Sheffield Dynamic Global Vegetation Model (henceforth SDGVM), which is used to simulate the carbon dynamics of forests and other kinds of vegetation. In Section 3 we present two alternative approaches to modelling the output of a dynamic simulator based on single-output emulation, and contrast the assumptions of these methods with those of the multi-output emulator. The methods are contrasted on the practical grounds of SDGVM in Section 4, then Section 5 presents some concluding remarks.

2. EMULATING MULTIPLE OUTPUTS

We consider a deterministic simulator returning outputs $\boldsymbol{y} \in \mathbb{R}^q$ from inputs $\boldsymbol{x}$ lying in some (often high-dimensional) input space $\mathcal{X} \subseteq \mathbb{R}^p$. The simulator is essentially a function $\boldsymbol{f}\colon \mathcal{X} \longmapsto \mathbb{R}^q$, and due to its deterministic nature it returns the same output if repeatedly executed on the same set of inputs. Despite $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ being in principle known for any $\boldsymbol{x}$, in practice the complexity of the simulator requires to execute the computer code in order to determine $\boldsymbol{y}$. From a Bayesian perspective, we thus regard $\boldsymbol{f}(\cdot)$ as an unknown function, and in line with e.g. O'Hagan et al. (1999) we represent the uncertainty surrounding it by means of the q-dimensional Gaussian process

$$\boldsymbol{f}(\cdot) \,|\, B, \Sigma, \boldsymbol{r} \sim \mathcal{N}_q\big(\boldsymbol{m}(\cdot), c(\cdot,\cdot)\Sigma\big) \tag{1}$$

conditional on hyperparameters B, Σ and $\boldsymbol{r}$. The notation here means that $\mathbb{E}\big[\boldsymbol{f}(\boldsymbol{x}_1) \,|\, B, \Sigma, \boldsymbol{r}\big] = \boldsymbol{m}(\boldsymbol{x}_1)$ and $\mathbb{C}\mathrm{ov}\big[\boldsymbol{f}(\boldsymbol{x}_1), \boldsymbol{f}(\boldsymbol{x}_2) \,|\, B, \Sigma, \boldsymbol{r}\big] = c(\boldsymbol{x}_1, \boldsymbol{x}_2)\Sigma \;\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$, where $c(\cdot,\cdot)$ is a positive-definite function such that $c(\boldsymbol{x}, \boldsymbol{x}) = 1 \;\forall \boldsymbol{x}$. Thus we assume a stationary, separable covariance structure, with covariance between the outputs at any given inputs given by the positive-definite matrix $\Sigma \in \mathbb{R}^+_{q,q}$ and with $c(\cdot,\cdot)$ providing spatial correlation across $\mathcal{X}$. Again as in previous work, we model the mean and correlation functions respectively as

$$\begin{aligned} \boldsymbol{m}(\boldsymbol{x}_1) &= B^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}_1) \\ c(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \exp\big\{-(\boldsymbol{x}_1 - \boldsymbol{x}_2)^{\mathrm{T}} R (\boldsymbol{x}_1 - \boldsymbol{x}_2)\big\} \quad . \end{aligned} \tag{2}$$

Here $\boldsymbol{h}\colon \mathcal{X} \longmapsto \mathbb{R}^m$ is an arbitrary vector of m regression functions $h_1(\boldsymbol{x}), \dots, h_m(\boldsymbol{x})$ shared by each individual function $f_j(\cdot),\; j = 1 \dots, q$; $B = [\boldsymbol{\beta}_1 \cdots \boldsymbol{\beta}_q] \in \mathbb{R}_{m,q}$ is a matrix of regression coefficients; and R is a diagonal matrix of p positive roughness parameters $\boldsymbol{r} = (r_1, \dots, r_p)$.

Gaussian processes have been widely used to model unknown functions in Bayesian statistics. Choice of the prior mean structure should be driven by both experience and simplicity: a linear specification $\boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x}) = (1, \boldsymbol{x}^{\mathrm{T}})$ has been found to be adequate in most applications, in which case $m = p + 1$. The form of $c(\cdot,\cdot)$ assumed in (2) implies that the $f_j(\cdot)$s are analytical functions, which may not be realistic for some simulators and can also lead to numerical instabilities in practice. Nevertheless, the mathematical tractability of the Gaussian covariance structure is convenient for the ensuing theory. Detailed guidance as to alternative classes of correlation functions can be found in Schlather (1997); Stein (1999), but Kennedy and O'Hagan (2001) found emulation in practice to be robust to the form of the covariance function. The model specification is completed by selecting a prior distribution for the hyperparameters. Although the methods of Oakley (2002) for eliciting prior beliefs about hyperparameters of a univariate Gaussian process could in principle be generalised to a multi-output setting, for simplicity we assume that only weak prior information is available about B and Σ. Accordingly the conventional 'non-informative' prior

$$\pi(B, \Sigma \,|\, \boldsymbol{r}) \propto |\Sigma|^{-\frac{q+1}{2}} \tag{3}$$

may be appropriate, whereas $\boldsymbol{r}$ is equipped with an arbitrary prior $\pi_{\boldsymbol{R}}(\cdot)$.

Running the computer code on a pre-selected design set $\mathcal{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\} \subset \mathcal{X}$ yields simulations organised in the output matrix $D = \big[f_j(\boldsymbol{s}_r)\big] \in \mathbb{R}_{n,q}$. These are the training data which are used to build the emulator. A good design set $\mathcal{S}$ for building efficient emulators can be characterised as having points well spaced out over portions of $\mathcal{X}$ which are thought of as relevant. The literature specialised in the field proposes space-filling design criteria, such as maxi-min Latin hypercubes (see for instance Sacks et al., 1989; Morris and Mitchell, 1995; Koehler and Owen, 1996), for simulators believed to operate homogeneously across the input space.

Formally, we identify the *multi-output emulator* as the posterior distribution of $\boldsymbol{f}(\cdot)$ given data D. To derive this posterior distribution, we first note that, from (1) and (2), the joint distribution of D conditional on hyperparameters B, Σ and $\boldsymbol{r}$ is the matrix-Normal distribution

$$D \,|\, B, \Sigma, \boldsymbol{r} \sim \mathcal{N}_{n,q}(HB, A, \Sigma) \quad , \tag{4}$$

where $H^{\mathrm{T}} = \big[\boldsymbol{h}(\boldsymbol{s}_1) \cdots \boldsymbol{h}(\boldsymbol{s}_n)\big] \in \mathbb{R}_{m,n}$ and $A = \big[c(\boldsymbol{s}_r, \boldsymbol{s}_l)\big] \in \mathbb{R}^{+}_{n,n}$. Standard Normal theory in addition to some matrix calculus manipulations hence leads to the following conditional posterior distribution for the computer simulator:

$$\boldsymbol{f}(\cdot) \,|\, B, \Sigma, \boldsymbol{r}, D \sim \mathcal{N}_q\big(\boldsymbol{m}^{\star}(\cdot), c^{\star}(\cdot,\cdot)\Sigma\big) \quad , \tag{5}$$

where for $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$

$$\begin{aligned} \boldsymbol{m}^{\star}(\boldsymbol{x}_1) &= B^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}_1) + (D - HB)^{\mathrm{T}} A^{-1}\boldsymbol{t}(\boldsymbol{x}_1) \\ c^{\star}(\boldsymbol{x}_1, \boldsymbol{x}_2) &= c(\boldsymbol{x}_1, \boldsymbol{x}_2) - \boldsymbol{t}^{\mathrm{T}}(\boldsymbol{x}_1) A^{-1}\boldsymbol{t}(\boldsymbol{x}_2) \end{aligned}$$

with $\boldsymbol{t}^{\mathrm{T}}(\cdot) = \big[c(\cdot, \boldsymbol{s}_1), \cdots, c(\cdot, \boldsymbol{s}_n)\big] \in \mathbb{R}^n$.

The posterior distribution of $\boldsymbol{f}(\cdot)$ conditional on $\boldsymbol{r}$ alone is found by integrating (5) with respect to the posterior distribution of B and Σ. First integrating B out of the combination of (5), (4) and (3) yields

$$\boldsymbol{f}(\cdot) \,|\, \Sigma, \boldsymbol{r}, D \sim \mathcal{N}_q\big(\boldsymbol{m}^{\star\star}(\cdot), c^{\star\star}(\cdot,\cdot)\Sigma\big) \quad ,$$

with

$$\begin{aligned} \boldsymbol{m}^{\star\star}(\boldsymbol{x}_1) &= \hat{B}_{\mathrm{GLS}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}_1) + (D - H\hat{B}_{\mathrm{GLS}})^{\mathrm{T}} A^{-1}\boldsymbol{t}(\boldsymbol{x}_1) \\ c^{\star\star}(\boldsymbol{x}_1, \boldsymbol{x}_2) &= c^{\star}(\boldsymbol{x}_1, \boldsymbol{x}_2) + \big[\boldsymbol{h}(\boldsymbol{x}_1) - H^{\mathrm{T}}A^{-1}\boldsymbol{t}(\boldsymbol{x}_1)\big]^{\mathrm{T}} \\ &\quad \cdot \big(H^{\mathrm{T}}A^{-1}H\big)^{-1}\big[\boldsymbol{h}(\boldsymbol{x}_2) - H^{\mathrm{T}}A^{-1}\boldsymbol{t}(\boldsymbol{x}_2)\big] \quad , \end{aligned}$$

where $\hat{B}_{\mathrm{GLS}} = \big(H^{\mathrm{T}}A^{-1}H\big)^{-1}H^{\mathrm{T}}A^{-1}D$ is the generalised least squares (GLS) estimator of B. Provided now that $n \geq m + q$, so that all ensuing posteriors are proper, the conditional posterior distribution of $\boldsymbol{f}(\cdot)$ given $\boldsymbol{r}$ then is the q-variate Student's process

$$\boldsymbol{f}(\cdot) \,|\, \boldsymbol{r}, D \sim \mathcal{T}_q\big(\boldsymbol{m}^{\star\star}(\cdot), c^{\star\star}(\cdot,\cdot)\hat{\Sigma}_{\mathrm{GLS}}; n - m\big) \tag{6}$$

with $n - m$ d.f., in which $\hat{\Sigma}_{\mathrm{GLS}} = (n-m)^{-1}(D - H\hat{B}_{\mathrm{GLS}})^{\mathrm{T}}A^{-1}(D - H\hat{B}_{\mathrm{GLS}})$ denotes the GLS estimator of Σ.

The final step to producing the emulator is to integrate (6) with respect to the posterior distribution of $\boldsymbol{r}$. Unfortunately this cannot be done analytically, and a full MCMC-based marginalisation of (6) with respect to the unknown roughnesses in R is computationally cumbersome. A more viable alternative lies in plugging-in some posterior estimates of $(r_1, \ldots, r_p)$: see Appendix A for additional details.

Given the estimated roughnesses, the posterior process (6) is the emulator of the simulator $\boldsymbol{f}(\cdot)$. Its mean function $\boldsymbol{m}^{\star\star}(\cdot)$ interpolates the training data exactly, and provides an approximation to $\boldsymbol{f}(\cdot)$ that can be used as a fast surrogate for the simulator. It should be stressed, though, that the emulator is more than just a code alias: the estimated covariance structure $c^{\star\star}(\cdot,\cdot)\hat{\Sigma}_{\mathrm{GLS}}$ provides a measure of the accuracy of the approximation, which can be arbitrarily improved across the input space by selecting an appropriately large training set $\mathcal{S}$. Once (6) has been fitted it becomes straightforward to generalise the Bayesian theory of uncertainty and sensitivity analysis given in e.g. Haylock and O'Hagan (1996); Oakley and O'Hagan (2002, 2004) to the multi-output case[†].

3. EMULATING A DYNAMIC SIMULATOR

Suppose that the dynamic model produces a vector of outputs $\boldsymbol{y} = (y_1, \ldots, y_T)$ spanning the simulation time period $t = 1, 2, \ldots, T$ and that a data matrix D as in Section 2 is obtained from some set of training runs. Here we introduce three procedures for emulating such a dynamic simulator.

Multi-output (MO) emulator The first method consists of just using the multi-output emulator (6), where now the dimension of the output space is $q = T$.

Ensemble of single-output (MS) emulators The second approach is to emulate the T outputs separately, each via a single-output emulator. Data for the t-th emulator would be then provided by the corresponding column of D.

Time input (TI) emulator The third approach involves building just one single-output emulator, following an idea originally outlined by Kennedy and O'Hagan (2001) to account for spatial outputs. The model is regarded as including time as an extra input, so that the output $y_t = f_t(\boldsymbol{x})$ is now represented as $f^*(\boldsymbol{x}, t)$, the extra input t taking values in the set $\{1, \ldots, T\}$. Emulation of $\boldsymbol{f}(\cdot)$ is then accomplished by emulating $f^*(\cdot, \cdot)$. Thus the training set for building this emulator comprises the nT outputs generated by inputs in the grid $\mathcal{S} \times \{1, \ldots, T\}$.

Each of the proposed methods, MO, MS and TI, has some advantages over the others, either in terms of flexibility or computational efficiency. From a computational perspective, the MO emulator is the simplest. When fitting general Gaussian processes, computational constraints typically arise from the size n of the design set $\mathcal{S}$. In particular an $n \times n$ matrix inversion is needed, which corresponds to inverting the correlation matrix A in Section 2. Repeated inversions of this matrix, which is typically ill-conditioned, are required when estimating, or accounting for uncertainty in, the roughness parameters. The computational load of the MO emulator is thereby comparable to that of building a single-emulator from the same number of runs, whereas the MS method will require a T-fold increase in CPU-time. The TI method may seem to be dramatically worse, since now the number of runs is no longer n but nT. However the fact that the training set of points is a grid means that (in case the same sort of correlation structure as in (2) holds) the inversion can be done as the Kronecker product of an $n \times n$ inversion and a $T \times T$ inversion. Still, this is computationally more demanding than the MO emulator.

We can compare the flexibility of the different methods in terms of the assumptions that they make about the simulator. All three approaches model the code output $\boldsymbol{y}$ as a Gaussian process, but they may differ in the structures they assume for its mean and covariance functions. As to the former, the simple linear form $\boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x}) = (1, \boldsymbol{x}^{\mathrm{T}})$ produces the same mean functions for the MO and MS methods, but the TI method is more restrictive because it would impose a linear form in t. However, an equivalent structure can be created in the TI mean function (e.g. by treating time as a factor and including its interactions with the other inputs), and in general all three models can always define the regression functions $\boldsymbol{h}(\cdot)$ so as to create the same mean functions.

More substantial differences should be noted in the covariance structures for the three strategies. Considering first the covariance between $f_t(\boldsymbol{x}_1)$ and $f_t(\boldsymbol{x}_2)$, that is between output values at different inputs but at the same time, in all three cases we generally assume the structure (2) with a diagonal R. However, because the MS method fits separate emulators for each output the roughness parameters in R can be estimated differently for each output, whereas both MO and TI will in general assume the same $\boldsymbol{r}$ irrespective of t (otherwise the analysis can become considerably more difficult). So in this respect the MS method is more flexible. The extra generality can be important, since in general there is no reason to believe that the smoothness of response to changes in a given input will be the same for all outputs. However this more restrictive form may still be reasonable in some cases: this is to say that in certain

[†] Ensuing formulae are available from the authors upon request.

applied settings an input to a given dynamic simulator can have a similar kind of effect on the output at different time points.

Now consider the covariance between $f_{t_1}(\boldsymbol{x}_1)$ and $f_{t_2}(\boldsymbol{x}_2)$, that is between the values of different outputs at different input settings. For the MO emulator approach, this is measured by (2) times the (t_1, t_2) entry of Σ, whereas for the TI emulator it will generally equal the Gaussian process variance multiplied by $\exp\{-r_T(t_1 - t_2)^2\}$. Although in principle it is possible to allow for alternative correlation structures between different time points in the TI method (see Section 5), the complete generality of the MO emulator in this respect is appealing, as it can be appreciated in the example of Section 4. The MS method does not model the outputs jointly, therefore this correlation is not defined. In practice, however, we would make joint predictions by assuming independence, which obviously entails a much stronger restriction.

The lack of a correlation structure over time in the MS method may be a serious drawback. In particular, one use of the emulator may be to estimate what the simulator would produce for time points $T_0 + 1$ to T, based on having run it only for time 1 to $T_0 \leq T$. The MS emulator would not be able to use the outputs at the earlier time points to help estimate simulator outcomes at later times. This in practice would not matter if enough training data had been available to build highly accurate emulators for all those later time points, but when dealing with large and complex dynamic simulators this will rarely be possible. Therefore, despite the extra flexibility in its covariance structure (due to allowing a different $\boldsymbol{r}$ for each output), we do not regard the MS method to be viable for emulating dynamic models. Hence in the example of the following section we compare the practical performance of the two remaining methods, that is MO and TI emulation.

4. APPLICATION: A PROCESS-MODEL FOR ECOSYSTEM CARBON

The Centre for Terrestrial Carbon Dynamics is a consortium of British academic and governmental institutions, established to improve scientific understanding of the role played by terrestrial ecosystems in the carbon cycle, with particular emphasis on forest ecosystems. The vegetation model SDGVM plays a central role in this research and we will consider emulating its daily version (Lomas et al., 2002). Vegetation models of its kind can be used to predict possible long-term responses of ecosystem processes to atmospheric CO_2 concentration and climate changes by modelling interactions at a regional to global scale between ecosystem carbon, water fluxes and vegetation. Reproduction of the biochemical processes of photosynthesis and phenology, which drive the growth and decay of generic vegetation categories (so-called plant functional types) at any given pixel, is pursued by encoding in SDGVM the knowledge base available about carbon fixation and transportation within the atmosphere-soil-vegetation system. Inputs to SDGVM comprise broad soil, vegetation and climate data, while outputs it returns include various measures of a site's carbon budget and miscellaneous environmental quantities. Of these outputs, we are especially interested in Net Biome Productivity (NBP, expressed in $g\, m^{-2} y^{-1}$), which indicates the amount of carbon sequestered by a site's vegetation after discounting for losses to the atmosphere due to both autotrophic and heterotrophic respiration and external disturbances (e.g. harvest, forest clearance, fire, disease).

For the purpose of testing the MO and TI emulators proposed in Section 3 we consider running SDGVM for ten years at a monthly resolution on a Deciduous Broadleaf forest area at Harwood in the UK. Feedback from the developers of SDGVM and some confirmatory analysis allowed us to identify as primary determinants for NBP ten input variables, which in the notation of Section 3 translates into $p = 10$ and $T = 120$. The occurrence of numerical singularities at the fitting stage of the TI emulator was alleviated by retaining from the simulated time span only the odd months, so that the emulation interval reduced to $\mathcal{T} = \{1, 3, \ldots, 117, 119\}$. To ensure fair comparison between the performances of the two emulators the MO emulator was also fitted to these 60 time steps only. SDGVM's input space was spanned with a maxi-min Latin hypercube design (Johnson et al., 1990; Morris and Mitchell, 1995; Koehler and Owen, 1996) of size $n = 400$, which we found to attain a good compromise between the accuracy and fitting time for both emulators. Design bounds were suggested by the modellers' uncertainty about true values for these inputs. Roughness parameters were then estimated for both MO and TI by their marginal posterior medians from an MCMC sample of size 10^5 based upon i.i.d. Log-Logistic priors,

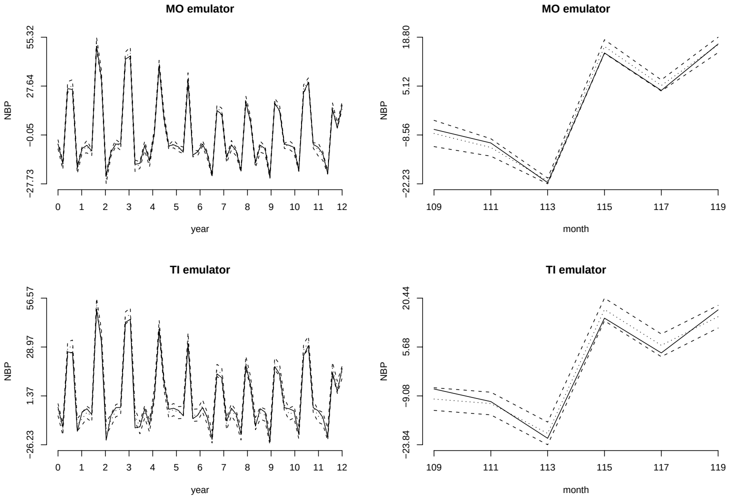

Figure 1. True (—) and emulated $(\cdots)$ *NBP* for an untested input configuration over 10 years (left panels) and for the 10^{th} year only (right panels), with 95% credibility bounds (– –) from the MO (upper panels) and TI (lower panels) emulators

as detailed in Appendix A. Inspection of the estimated sets of roughnesses obtained from both emulation strategies suggested that SDGVM's *NBP* output is mildly sensitive to most inputs.

In order to evaluate the performance of the two emulators, we selected 100 points in the input space that were not part of the original design set $\mathcal{S}$. For each of these inputs, we (a) ran SDGVM to obtain the true model outputs at the 60 time steps, and (b) computed the predictions of those sequences (posterior means, variances and covariances) for the two emulators. Figure 1 contrasts *NBP* values obtained by running SDGVM at one of those input points against corresponding predictions produced by both the MO and TI emulators. For both emulators, the estimated (posterior mean) output lies close to the true SDGVM output, showing that they are emulating the true model quite accurately. To some extent this is predictable, because the principal dynamics of the model output for any given configuration of the 10 uncertain inputs are driven by the weather at this site, which is assumed known. However, we have found that the detail of the *NBP* paths changes across the 100 different test inputs, yet both emulators were able to reproduce those changing details well.

The 95% credibility bounds around the MO emulator estimates are somewhat narrower than those computed with the TI method, which seems to indicate a better predictive performance of the former strategy over the latter. Specifically, the proportion of true *NBP* values falling within the respective 95% bounds, averaged across the 100 test inputs, was 90.6% for the MO emulator and 80.1% for the TI emulator. Therefore, not only does the MO emulator produce narrower prediction bounds, but these are also validated much better by the test runs. In fact, the TI emulator's bounds are wider but still not wide enough. Note that the performance of the MO emulator may be better than the TI emulator's but it is not perfect, in the sense that the coverage proportion of 90.6% was lower than the nominal 95%. Real-world simulators are rarely as smooth or as homogeneous in their behaviour across the input space as is assumed in the Gaussian process model, and this is typically reflected in prediction intervals that are a little too small. Whereas the behaviour of the MO emulator in this example is consistent with experience in other real applications, the TI emulator's performance would not be acceptable in practice.

Clearer insights into each emulator's abilities and weaknesses are provided by conventional diagnostic checks based on distributions of residuals in the 100 test output sequences. Figure 2 shows the residuals, standardised by dividing by their posterior predictive standard deviations, from both emulators for the 100 test sequences at each of three evenly spaced time points. According to the theory, such residuals should approximately follow standard Normal distributions, shown by the solid diagonal lines on the quantile plots. Both emulators exhibit some heavy-tailed characteristics. This behaviour is also commonly

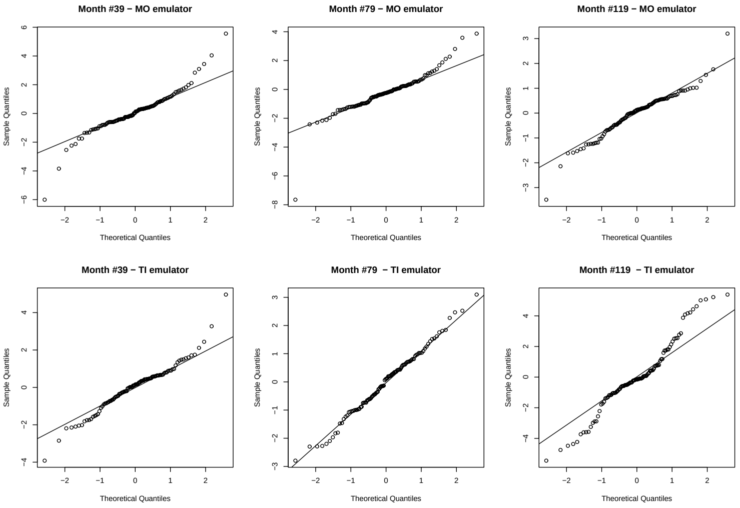

Figure 2. QQ-plots of standardised residuals for selected months from MO (top panel) and TI (bottom panel) emulators of 100 SDGVM test outputs

observed in validation of single-output emulators, and is again due to failure of the assumed smoothness and homogeneity of the simulator output. Except perhaps for month number 119, these graphs do not show the TI emulator performing appreciably worse than the MO emulator, despite the poorer coverage of intervals remarked upon above and, as discussed in Section 3, its more rigid correlation structure. However, a difference emerges when we compare the average root mean square prediction error (RMSPE), based upon the previously examined standardised residuals, across all 100 sequences and 60 time points. Whereas in case of good model fit the RMSPE should fluctuate around 1, its value for the TI emulator is 1.685 compared to that of 1.256 for the MO emulator. The latter 1.256 figure is consistent with experience with single-output emulators in other applications, but the 1.685 RMSPE for the TI emulator would again give cause for concern in practice.

Whereas RMSPE is a good test for the individual predictions, a stronger test for the 60 correlated

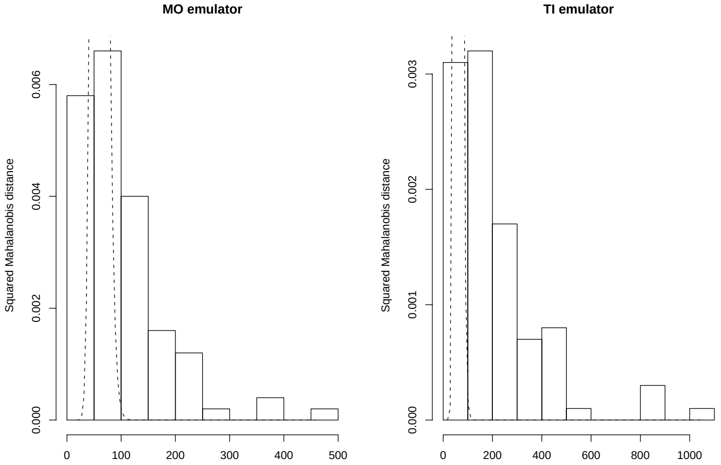

Figure 3. Histograms of squared Mahalanobis distances from MO (left) and TI (right) emulators of 100 SDGVM test outputs, with superimposed theoretical χ^2_{60} densities $(\cdots)$

outputs in a sequence is to compute the squared Mahalanobis distance for each sequence, which according to the theory should be approximately distributed as a χ^2_{60} in this context. The distribution of such statistic for each proposed emulator is contrasted in Figure 3 with its hypothetical χ^2_{60} density. While still favouring the MO model, the discrepancy between the computed and theoretical Mahalanobis distances indicates that neither emulation technique adequately captured the correlation structure featured by SDGVM through time. In particular, the squared Mahalanobis distances for the MO and TI emulators averaged over the 100 test points to 99.25 and 213.6 respectively, against an expected value of 60; corresponding variances, to be compared with a theoretical value of 120, instead amounted to 6416.308 and 34656.27. Despite squared Mahalanobis distances overall show that both emulators suffer from significant modelling shortcomings at a multivariate level, it is worthwhile noticing how predictive variances from the MO model still validate better than those from its TI counterpart on a per-point basis. Indeed reliable covariance estimation proved to be difficult under either strategy (especially the TI). We used a larger training set (400 points) than we would have needed for single-output emulation of SDGVM, specifically because of the difficulty of estimating covariances, yet larger training samples tend to cause other numerical instabilities by inducing ill-conditioned correlation matrixes.

Additional insights as to each model's predictive performance were obtained by computing average auto-correlations from the 100 test SDGVM outputs, which in Figure 4 are contrasted with corresponding estimates from the MO and TI emulators. It can be seen that correlations through time generated by the MO model alternate significant positive and negative values at half-year lags, whereas for the TI emulator they only exhibit an abrupt exponential decay. As discussed in Section 3, such discrepant patterns clearly arise from the different assumptions around which each emulator is built. Nonetheless neither emulation strategy seems to adequately accommodate the dynamics of SDGVM's sample paths. The TI model entertains an invariably rigid auto-correlation function, which drops to negligible values at lags greater than two months. On the other hand, the MO emulator can be seen to better reflect SDGVM's underlying correlation structure. Examination of the posterior predictive distributions of each emulator for months $\{61, \ldots, 119\}$, conditional on observing the true *NBP* values for the first 30 months, reflected the above findings: average RMSPEs over the 100 input points (and corresponding average squared Mahalanobis distances, which should now be approximately χ^2_{30}) were 1.291 (26.64) and 1.501 (80.85) for the MO and TI emulators respectively. The impact of the different auto-correlation structure on the performance of the compared models is further emphasised by the reduction in their predictive variance, obtained when each is conditioned to *NBP* outputs for months $\{1, \ldots, 59\}$. Due to the significant (albeit often inaccurate) auto-correlations it features through time, *NBP* estimates for months $t = 61, \ldots, 119$

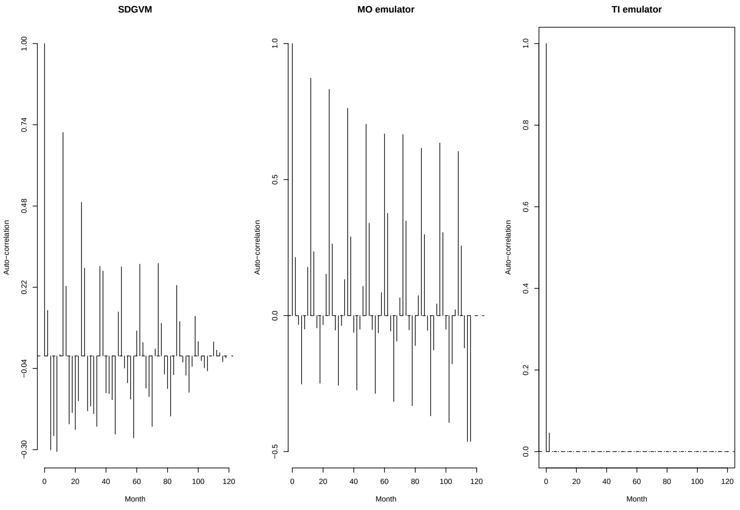

Figure 4. Plots of average auto-correlations from 100 true (left panel) and estimated (centre and right panels for the MO and TI emulators respectively) SDGVM outputs

produced by the conditional MO emulator show on average 39.08% less variance than those obtained from its unconstrained version. Conversely the TI emulator gaines no average predictive precision at all, in that as highlighted by Figure 4 it only corroborates its predictions with information about SDGVM outputs dating back 2 months at most[‡].

In the light of the above findings we thus argue in favour of adopting the MO emulator, since it overall reproduced and validated SDGVM evaluations appreciably better than its TI counterpart. However we also recognise that the inability (especially pathological for the TI model) to fully account for the dynamics of SDGVM sample paths underpins structural model deficiencies which in turn call for further, probably application-specific, research.

5. CONCLUSIONS

The paper focuses on two intertwined problems. First, we develop the multi-output emulator as an extension of theoretical results already established in the field of Bayesian emulation of a single output. Second, we consider the use of the multi-output emulator to model the time series output from a dynamic simulator, contrasting it with two alternatives approaches: one using multiple single-output emulators and the other based on treating time as an auxiliary input. A discussion of the modelling restrictions implied by the three alternative emulators suggested that the multiple single-output emulators approach was unsatisfactory because of its failure to address correlations through time. A subsequent empirical exploration based on emulating a time series of *NBP* outputs from SDGVM showed clearly better performance by the multi-output emulator.

We believe that the multi-output emulator can form the basis for successful emulation of dynamic simulators, which in practice is an important stepping stone for tackling problems such as uncertainty analysis, sensitivity analysis and calibration of dynamic models. However, there remain some directions for further research. The key to the multi-output emulator's improved performance relative to the time-input emulator lies in its more flexible modelling of correlations over time, but this is also a source of extra computational load through the need for larger training samples. It may be that a more restrictive structure, in which Σ is constrained to follow a standard time series form, would allow even better emulation with smaller training samples. Analogously the time-input emulator's correlation function could be assigned a time series structure in place of the usual $\exp\big\{-r_T(t_1-t_2)^2\big\}$, but it seems likely that the multi-output emulator would still be easier to use. Where there is good information to suggest an appropriate time-series structure, this approach seems worth investigating.

It would also be desirable to improve upon the flexibility of the multi-output emulator by relaxing its assumption of a set of roughness parameters common to all outputs; the generality of the many single-output emulators approach in this regard is its principal benefit. Unfortunately, it seems to be very difficult to combine different roughness parameters for each input in $\boldsymbol{x}$ with some covariance structure on the output space (in such a way as to create a valid positive-definite correlation function).

Acknowledgements

This research was supported by the Natural Environment Research Council through its funding for the Centre for Terrestrial Carbon Dynamics. The authors also wish to ackowledge Dr Marc C. Kennedy for providing the data utilised in the application.

APPENDIX A. INFERENCES ON ROUGHNESS PARAMETERS

Bayesian emulation of multi-response simulators can be implemented once roughness coefficients in (6) are properly dealt with. The prior distribution proposed in Section 2 for the hyperparameters has the form

$$\pi(B,\Sigma,\boldsymbol{r}) \propto \pi_{\boldsymbol{R}}(\boldsymbol{r})|\Sigma|^{-\frac{q+1}{2}} \quad . \tag{7}$$

Specification of $\pi_{\boldsymbol{r}}(\cdot)$ via elicitation of genuine prior beliefs is a difficult task, and alternative choices of 'default' priors are subject of ongoing research (see among the others Berger et al., 2001; Paulo, 2003),

[‡] Constraining the TI emulator to previous SDGVM evaluations leads to a variance reduction of 1.002% for *NBP* predictions at month 61 only.

mainly motivated by difficulties in avoiding improper posteriors for the r_i's. In the example of Section 4, we have used on an input/output normalised scale the product of i.i.d. vague (albeit proper) Log-Logistic priors, that is $\pi_{\boldsymbol{R}}(\boldsymbol{r}) = \prod_{i=1}^{p}(1+r_i^2)^{-1}$.

Combining (7) with (4) yields, via Bayes' theorem, the full posterior

$$\pi(B, \Sigma, \boldsymbol{r} \mid D) \propto \pi_{\boldsymbol{R}}(\boldsymbol{r})|A|^{-\frac{q}{2}}|\Sigma|^{-\frac{n-m+q+1}{2}} \exp\Big\{-\frac{1}{2}\Big[\mathrm{Tr}\{D^{\mathrm{T}}GD\Sigma^{-1}\} + \mathrm{Tr}\big\{(B-\hat{B}_{\mathrm{GLS}})^{\mathrm{T}}H^{\mathrm{T}}A^{-1}H(B-\hat{B}_{\mathrm{GLS}})\Sigma^{-1}\big\}\Big]\Big\} \quad .$$

It is then easy to integrate out the hyperparameter matrices B and Σ to obtain

$$\pi_{\boldsymbol{R}}(\boldsymbol{r} \mid D) \propto \pi_{\boldsymbol{R}}(\boldsymbol{r})|A|^{-\frac{q}{2}}|H^{\mathrm{T}}A^{-1}H|^{-\frac{q}{2}}|D^{\mathrm{T}}GD|^{-\frac{n-m}{2}} \quad . \tag{8}$$

A fully Bayesian approach would then proceed by sampling from this distribution, for example by MCMC, in order to average the conditional posterior (6) with respect to $\boldsymbol{r}$. In practice, it is simpler and adequate just to plug estimates of the r_is into (6). These estimates may be obtained by maximising (8) with respect to the r_is, or by taking mean or median values from an MCMC run.

References

J. O. Berger, V. De Oliveira, and B. Sansó. Objective Bayesian analysis of spatially correlated data. *J. Amer. Statist. Assoc.*, 96(456):1361–1374, 2001.

P. S. Craig, M. Goldstein, A. H. Seheult, and J. A. Smith. Bayes linear strategies for matching hydrocarbon reservoir history. In *Bayesian statistics, 5 (Alicante, 1994)*, Oxford Sci. Publ., pages 69–95. Oxford Univ. Press, New York, 1996.

R. G. Haylock and A. O'Hagan. On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. In *Bayesian statistics, 5 (Alicante, 1994)*, Oxford Sci. Publ., pages 629–637. Oxford Univ. Press, New York, 1996.

M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin designs. *Journal of Statistical Planning and Inference*, 26:131–148, 1990.

M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 63(3):425–464, 2001.

M. C. Kennedy, A. O'Hagan, and N. Higgins. Bayesian analysis of computer code outputs. In *Quantitative methods for current environmental issues*, pages 227–243. Springer, London, 2002.

J. R. Koehler and A. B. Owen. Computer experiments. In *Design and Analysis of Experiments*, volume 13 of *Handbook of Statist.*, pages 261–308. North-Holland, Amsterdam, 1996.

M. R. Lomas, F. I. Woodward, and S. Quegan. The role of dynamic vegetation models. Technical report, University of Sheffield, Sheffield UK, 2002.

M. D. Morris and T. J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43:381–402, 1995.

J. E. Oakley. Eliciting Gaussian process priors for complex computer codes. *The Statistician*, 51(1): 81–97, 2002.

J. E. Oakley and A. O'Hagan. Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, 89(4):769–784, 2002.

J. E. Oakley and A. O'Hagan. Probabilistic sensitivity analysis of complex models: a Bayesian approach. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 66(3):751–769, 2004.

A. O'Hagan. Some Bayesian numerical analysis. In *Bayesian statistics, 4 (Peñíscola, 1991)*, pages 345–363. Oxford Univ. Press, New York, 1992.

A. O'Hagan, M. C. Kennedy, and J. E. Oakley. Uncertainty analysis and other inference tools for complex computer codes. In *Bayesian statistics, 6 (Alcoceber, 1998)*, pages 503–524. Oxford Univ. Press, New York, 1999.

R. Paulo. Default priors for Gaussian processes. Technical report, National Institute of Statistical Sciences and Statistical and Applied Mathematical Sciences Institute, Durham, North Carolina, USA, 2003.

J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statist. Sci.*, 4(4):409–435, 1989. With comments and a rejoinder by the authors.

A. Saltelli, K. Chan, and E. M. Scott, editors. *Sensitivity Analysis.* Wiley Series in Probability and Statistics. John Wiley & Sons Ltd., Chichester, 2000.

M. Schlather. Introduction to positive definite functions and to unconditional simulation of random fiels. Technical report, Department of Mathematics and Statistics, FAculty of Applied Sciences, Lancaster University, 1997.

M. L. Stein. *Interpolation of Spatial Data*. Springer Series in Statistics. Springer-Verlag, New York, 1999. Some theory for Kriging.