A Bayesian Analysis of Complex Dynamic Computer Models

Stefano Conti, Clive W. Anderson, Marc C. Kennedy and Anthony O'Hagan

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

Abstract: In many fields of science, sophisticated mathematical models are devised and implemented within large computer codes in order to simulate and predict complex real-world phenomena. These models are known for being exposed to various sources of uncertainty taking place at their building and validation steps, so that they are routinely subject to reliability tests by means of uncertainty and/or sensitivity analysis. Since such diagnostics typically require a large number of training code runs, for CPU-intensive models an approach based around preliminary *emulation* of a code's response, followed by application of the aforementioned techniques to the emulator, can be more practical and efficient. This paper extends results already established within a Bayesian set-up for deterministic models (see e.g. [1]) to dynamic multi-response computer codes, for which some of the outputs at one stage of a simulation become inputs to the subsequent stage. Advantages and difficulties in the implementation are here discussed, and a test-bed application to the Sheffield Dynamic Global Vegetation Model, developed within the UK Centre for Terrestrial Carbon Dynamics, is also presented.

Keywords: Bayesian inference, computer experiments, hierarchical models, sensitivity analysis, uncertainty analysis

1. INTRODUCTION

A nowadays widespread practice in the scientific community is the utilisation of large computer codes embedding sophisticated mathematical models descriptive of complex aspects of reality. The exploratory and predictive ability of any computer simulator is often hampered by substantial model preparation and computational requirements. Whereas computational burden is not remarkably cumbersome, nonetheless various uncertainties can still significantly compromise the performance of a computer model. Among recognised sources of uncertainty affecting the processes of model building and validation are (see [1] for a thorough discussion on the subject): **parameter uncertainty**, originating from unknown quantities tuning the code; **model inadequacy**, due to necessarily imperfect fit to the observed data; **residual variability**, related either to intrinsic randomness or unrecognised features of the real-world phenomenon; **parametric variability**, arising from quantities conveniently left unspecified; **observation error**, caused by inaccuracies at the hard data recording stage; and **code uncertainty**, related to the complex nature of the simulator.

Further author information: (Send correspondence to Stefano Conti, e-mail: s.conti@sheffield.ac.uk)

Several methodologies aimed at ascertaining the reliability and effectiveness of a given computer model are available off-the-shelf from the classical statistical literature: an exhaustive reference is provided by [2]. Unfortunately standard uncertainty/sensitivity analysis tools often require a large number of training code runs, hence proving unsuitable for validation of computationally expensive models. In order to circumvent this problem, a strategy based around preliminary emulation of the code's outcome (*meta-modelling*) suggested by [3] has been widely adopted. This procedure would typically be followed by application of the aforementioned techniques to the emulator, which in fact is treated like a cheaper alias of the original code.

In this context interesting results were obtained via a Bayesian semi-parametric representation of deterministic single-response codes, that is models returning the same scalar output when repeatedly fed with the same input configuration. In previous works in the field (refer e.g. to [4]) a Gaussian process prior for the code's output was shown to be a convenient, flexible and reasonable tool, especially for tackling the problems of model calibration and rectification.

Sometimes special features of the phenomena of interest may translate into aspects of the computer model that could complicate standard emulation. This is usually the case with dynamic computer models, typically designed for time-evolving processes. In particular, such codes have the distinctive feature that some of the inputs required at each stage of a simulation are actually outputs from previous stages. This affects the structure of the input space, in that variables required for the code's operation can be classified as: *constants*, which describe enduring characteristics of the examined events; *time-varying*, related to aspects of the process's evolution over time; and *intermediate* or *final outputs*, which in turn may or may not be recycled by the model into subsequent simulations. In this context meta-modelling of the code's outputs needs to be suitably adjusted in order to accommodate any relationship featured by variables evolving over time.

The paper discusses such adjustments and is organised as follows. In Section 2 a generalisation of the emulator as developed in [4] to multi-response codes is detailed. Section 3 is devoted to adapting the emulator to encompass dynamic computer models as well. An application to a sub-module of an environmental model is outlined in Section 4, while Section 5 summarises concluding remarks.

2. MODELLING MULTI-RESPONSE COMPUTER MODELS

Consider a deterministic computer model which takes inputs $\boldsymbol{x}$, typically lying in some (possibly high-dimensional) input space $\mathscr{X}$, and returns outputs $\boldsymbol{y}$. The process of computing vectors $\boldsymbol{y}$ from $\boldsymbol{x}$ can be formalised via a function $\boldsymbol{f}\colon \mathscr{X} \longmapsto \mathbb{R}^q$, where the input space is usually a subset of the Euclidean space $\mathbb{R}^p$ with typically $p \geq q$. In light of the usually high degree of complexity of the empirical processes being modelled via $\boldsymbol{f}(\cdot)$, it is normally unaffordable to explore the whole input space $\mathscr{X}$ just by repeatedly running the computer program. As a consequence, although in principle deterministic the code is in fact prone to at least code uncertainty (see Section 1). Hence $\boldsymbol{f}(\cdot)$ is regarded as a stochastic function on $\mathscr{X}$, though still assumed to be reasonably smooth. Therefore, within the Bayesian framework an appropriate prior process needs to be assigned to the random function $\boldsymbol{f}(\cdot)$. In line with [4] we suppose that knowledge of the simulator can be summarised by the semi-parametric Gaussian process representation

$$\big[\boldsymbol{f}(\cdot) \mid B, \Sigma, R\big] \sim \mathscr{N}_q\big(\boldsymbol{m}(\cdot), c(\cdot,\cdot)\Sigma\big) \quad , \tag{1}$$

where $\forall\, \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathscr{X}$

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

Here $\boldsymbol{h}\colon \mathscr{X} \longmapsto \mathbb{R}^m$ is a vector of arbitrary regression functions $\boldsymbol{h}(\boldsymbol{x})$, common to every component $f_i(\cdot),\ i = 1\ldots, q$ of $\boldsymbol{f}(\cdot)$, $B = [\boldsymbol{\beta}_1 \cdots \boldsymbol{\beta}_q] \in \mathbb{R}_{m,q}$ is a matrix of regression coefficients, $\Sigma = [\sigma_{ij}] \in \mathbb{R}_{q,q}$ a dispersion matrix with generic entry $\sigma_{ij} = \mathbb{C}\mathrm{ov}\big[f_i(\cdot), f_j(\cdot)\big]$ and $R = \operatorname{diag}\{r_i\} \in \mathbb{R}^+_{q,q}$ a diagonal positive-definite roughness matrix. Gaussian processes constitute the natural counterpart of the Normal model usually invoked for finite-dimensional estimands, and enjoy the same flexibility and tractability when utilised for addressing problems related to functional inference. Separability of the covariance structure between the code inputs and its outputs is here assumed for simplicity. It is also worthwhile mentioning that diagonality of R, here imposed for parsimony, implies that the correlation structure between any pair $f(\boldsymbol{x}_1)$ and $f(\boldsymbol{x}_2)$ is insensitive to any interaction among inputs.

Running the computer code on a pre-selected *design set* $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\} \subset \mathscr{X}$ yields simulations organised in the data matrix $D = \big[f_i(\boldsymbol{s}_r)\big] \in \mathbb{R}_{n,q}$. The design set can be selected in accord to some space-filling experimental design criterion: see for instance [5, 6, 7] and annotated bibliography. Due to the learning mechanism intrinsic to the Bayesian paradigm, as more model runs become available the posterior distribution of $\boldsymbol{f}(\cdot)$ becomes more concentrated near the input configurations, which in turn are exactly interpolated.

In light of the assumptions listed above, the joint distribution of the code responses D conditional on nuisance parameters B, Σ, R is the matrix-Normal distribution

$$[D \mid B, \Sigma, R] \sim \mathscr{N}_{n,q}\big(HB,\ \Sigma \otimes A\big) \quad ,$$

where $H^{\mathrm{T}} = \big[\boldsymbol{h}(\boldsymbol{s}_1) \cdots \boldsymbol{h}(\boldsymbol{s}_n)\big] \in \mathbb{R}_{m,n}$, $A = [c(\boldsymbol{s}_r, \boldsymbol{s}_l)] \in \mathbb{R}_{n,n}$ and $\otimes$ denotes the Kronecker product. Letting now $\boldsymbol{t}^{\mathrm{T}}(\cdot) = \big[c(\cdot, \boldsymbol{s}_1) \cdots c(\cdot, \boldsymbol{s}_n)\big] \in \mathbb{R}^n$, standard Normal theory and some matrix calculus manipulations lead to the following conditional posterior distribution for the computer simulator:

$$\big[\boldsymbol{f}(\cdot) \mid B, \Sigma, R, D\big] \sim \mathscr{N}_q\big(\boldsymbol{m}^\star(\cdot), c^\star(\cdot,\cdot)\Sigma\big) \quad , \tag{2}$$

where

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

A possible way to obtain the posterior process of $\boldsymbol{f}(\cdot)$ conditional on the roughness matrix R alone is by integration of (2) with respect to the posterior distribution of the nuisance parameters B, Σ. Since any substantial information about such parameters will hardly ever be elicited from the code developers, a prior choice found to be both reasonable and manageable is the Jeffreys non-informative independence distribution $\pi^{\mathrm{J}}(B, \Sigma \mid R) \propto |\Sigma|^{-\frac{q+1}{2}}$. Standard Bayesian calculations from (2) and $\pi^{\mathrm{J}}(\cdot)$ yield

$$[\boldsymbol{f}(\cdot) \mid \Sigma, R, D] \sim \mathscr{N}_q\big(\boldsymbol{m}^{\star\star}(\cdot), c^{\star\star}(\cdot,\cdot)\Sigma\big) \quad , \tag{3}$$

where

$$\begin{aligned} \boldsymbol{m}^{\star\star}(\cdot) &= D^{\mathrm{T}}A^{-1}\boldsymbol{t}(\cdot) + \hat{B}^{\mathrm{T}}_{\mathrm{GLS}}\big[\boldsymbol{h}(\cdot) - H^{\mathrm{T}}A^{-1}\boldsymbol{t}(\cdot)\big] & \text{(4a)}\\ c^{\star\star}(\boldsymbol{x}_1, \boldsymbol{x}_2) &= c^{\star}(\boldsymbol{x}_1, \boldsymbol{x}_2) \\ &\quad + \big[\boldsymbol{h}(\boldsymbol{x}_1) - H^{\mathrm{T}}A^{-1}\boldsymbol{t}(\boldsymbol{x}_1)\big]^{\mathrm{T}}\big(H^{\mathrm{T}}A^{-1}H\big)^{-1}\big[\boldsymbol{h}(\boldsymbol{x}_2) - H^{\mathrm{T}}A^{-1}\boldsymbol{t}(\boldsymbol{x}_2)\big] & \text{(4b)} \end{aligned}$$

and $\hat{B}_{\mathrm{GLS}} = \big(H^{\mathrm{T}}A^{-1}H\big)^{-1}H^{\mathrm{T}}A^{-1}D$ is the GLS estimator of B. Provided that $n \geq m + q$ so that all ensuing posteriors are proper, the conditional posterior Student's $\mathscr{T}$ process

$$[\boldsymbol{f}(\cdot) \mid R, D] \sim \mathscr{T}_q\big(\boldsymbol{m}^{\star\star}(\cdot), c^{\star\star}(\cdot,\cdot)\hat{\Sigma}_{\mathrm{GLS}}; n - m\big) \tag{5}$$

is finally obtained, 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.

Direct utilisation of (5) and of precursory results for drawing inferences about the simulator $\boldsymbol{f}(\cdot)$ must still be preceded by estimation of the unknown roughness matrix R. A full Bayesian treatment of the roughness parameters, notoriously difficult to estimate (see in particular [8, 9] for insights), is here rejected on the grounds of computational tractability. Nonetheless a plug-in approach based upon the posterior mode of $(r_1, \ldots, r_p)$ arising from a diffuse, albeit proper, prior was found to yield satisfactory results. Once this task is accomplished expressions (4a)-(4b) furnish respectively a cheap code interpolator and, when combined with $\hat{\Sigma}_{\mathrm{GLS}}$, a measure of its accuracy. Furthermore, integration of the posterior $\mathscr{T}$ process (5) relative to appropriately selected distributions over (possibly portions of) the input space $\mathscr{X}$ constitutes the basis of customary uncertainty analysis techniques. For a complete review of such methodologies, and their adaptations and implications within a Bayesian set-up, see [2, 4, 10].

3. DYNAMIC META-MODELS

3.1. From Static to Dynamic Emulation

Dynamic computer models come into play when it is desired to reproduce and examine the evolutionary nature of a time-varying process. As mentioned in Section 1, in order to reproduce dynamic patterns computer models customarily utilise outputs from each stage of a simulation as inputs to subsequent stages. This is in essence achieved by computing the state vector $\boldsymbol{y}_t$ relative to a time step $t = 1, \ldots$ from inputs comprising both constant tuning values $\boldsymbol{x}$ and outputs $\boldsymbol{y}_{t-1}$ from the previous time period $t - 1$[*]. Taking into

[*] For many physical processes to impose a Markovian dependence of dynamic outputs over time, although not correct, may still produce an acceptable representation.

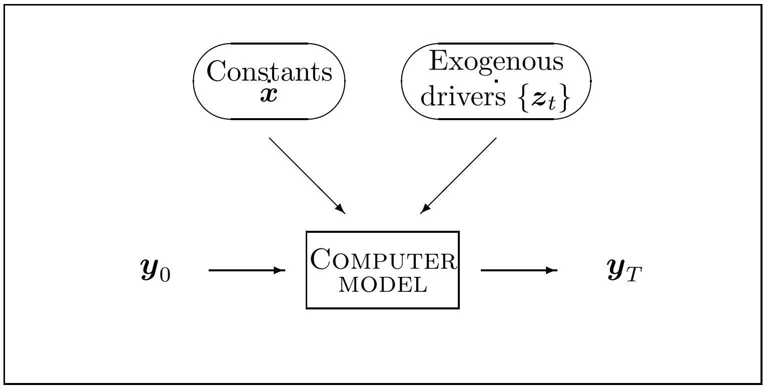

Figure 1. Single-step meta-modelling

account other time-evolving exogenous code drivers $\{\boldsymbol{z}_t\}$, the model's functioning can be then represented via the recursive relation

$$\begin{aligned}\boldsymbol{y}_t &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}_t, \boldsymbol{y}_{t-1}) \\ &= \boldsymbol{f}\big[\boldsymbol{x}, \boldsymbol{z}_t, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}_{t-1}, \boldsymbol{y}_{t-2})\big] \quad . \\ &= \cdots\end{aligned}$$

If the time span of interest is delimited by endpoints $t = 0$ and $t = T$, then in principle emulation of $\boldsymbol{f}(\cdot)$ can be attained over such interval just in a single-run fashion (see Figure 1): under this perspective the simulator is imagined to take a set of input values, comprising initial system descriptors $\boldsymbol{y}_0$, and to return a collection of outputs inclusive of $\boldsymbol{y}_T$. The main appeal of such procedure clearly lies in enabling straightforward application of standard statistical analysis tools already existing for static codes. An important disadvantage however is that in this set-up the input space $\mathscr{X}$ comprises constants $\boldsymbol{x}$, the

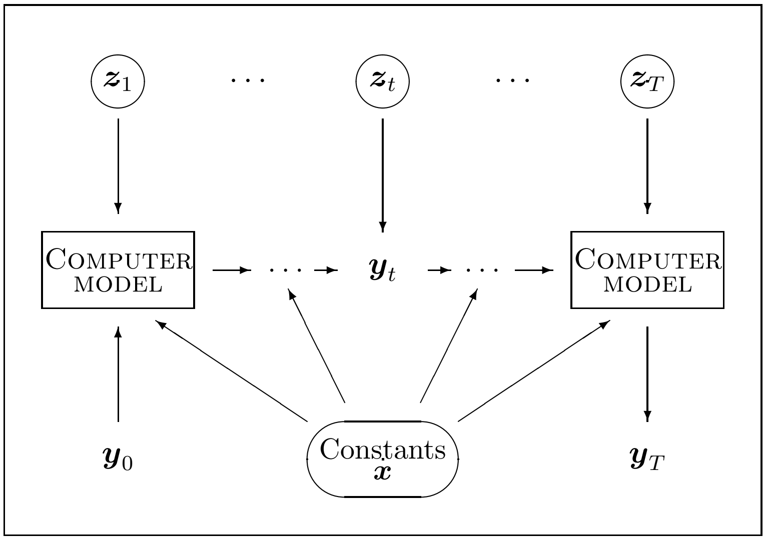

Figure 2. Recursive meta-modelling

initial state vector $\boldsymbol{y}_0$ and all drivers $\{\boldsymbol{z}_t\}$. On the other hand the output generated by $\boldsymbol{f}(\cdot)$ consists of the whole collection of runs $\{\boldsymbol{y}_t\}$. Therefore applications of this strategy is severely hindered by typically unwieldy dimensions for both the model input and output spaces. Additional relevant flaws undermining single-step emulation can be also recognised in the need to rebuild the posterior process (5) from scratch whenever the temporal window of interest changes and in deferment of any occasional model rectification exercise until the end of the simulation.

An approach which intuitively retains the code's evolutionary pattern is alternatively depicted in Figure 2. Here the proposed idea is to run a single-step emulator in a recursive fashion, until coverage of the time span $(0, T)$ is attained: given the state vector $\boldsymbol{y}_t$ acquired at any time $t = 0, \ldots, (T-1)$, subsequent emulation of $\boldsymbol{y}_{t+1}$ is then computed only on the basis of $\boldsymbol{y}_t$ and of current drivers $\boldsymbol{z}_{t+1}$. As a consequence input and output spaces become more tractable and interactive data assimilation is now feasible. Main drawbacks of the method in this case are: need to extend theoretical results from static to dynamic codes; tighter accuracy requirements at each time step for ensuring adequate overall meta-modelling; and gradual fading of computational advantages over direct Monte Carlo simulation for increasing values of T.

3.2. Dynamic Emulation Theory

Assume for simplicity that no exogenous drivers are required for running $\boldsymbol{f}(\cdot)$. In addition, suppose with no loss of generality that the variables which the code rolls forward over time are the last $q \le p$ input entries; that is, for any t and any suitable $\boldsymbol{x} \in \mathbb{R}^{p-q}$, $\boldsymbol{y}_t \in \mathbb{R}^q$ we have $\boldsymbol{y}_{t+1} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_t)$.

An issue that should promptly be highlighted is that under assumption (1) a recursive emulator of nonlinear codes can at best be approximately Gaussian: if $\boldsymbol{y}_t = \boldsymbol{f}(\bar{\boldsymbol{x}})$ for some $\bar{\boldsymbol{x}} \in \mathscr{X}$, then for any $\boldsymbol{x}$ such that $(\boldsymbol{x}, \boldsymbol{y}_t) \in \mathscr{X}$ the statement

$$\big[\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{Y}_t) \mid \Sigma, R\big] \sim \mathscr{N}_q(\cdot, \cdot)$$

will not strictly hold[†]. Keeping an assumption of approximate Normality for modelled outputs on practical grounds obviously entails careful assessment of its plausibility at each stage of the whole emulation process. Subject to this condition it then becomes feasible to explore the first and second order properties of the posterior distribution of $\boldsymbol{f}(\boldsymbol{y}_t)$, given previous-step outputs $\boldsymbol{y}_{t-1}$ and nuisance parameters Σ, R. In fact, recalling results (4) approximate expressions have been derived in closed form for

$$\mathbb{E}\big[\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{Y}_t) \mid \Sigma, R, D\big] = \mathbb{E}\big[\boldsymbol{m}^{\star\star}(\boldsymbol{x}, \boldsymbol{Y}_t) \mid \Sigma, R, D\big] \tag{6a}$$

$$\begin{aligned} \mathbb{C}\mathrm{ov}\big[\boldsymbol{f}(\boldsymbol{x}_1, \boldsymbol{Y}_{t_1}), &\boldsymbol{f}(\boldsymbol{x}_2, \boldsymbol{Y}_{t_2}) \mid \Sigma, R, D\big] \\ &= \mathbb{C}\mathrm{ov}\big[\boldsymbol{m}^{\star\star}(\boldsymbol{x}_1, \boldsymbol{Y}_{t_1}), \boldsymbol{m}^{\star\star}(\boldsymbol{x}_2, \boldsymbol{Y}_{t_2}) \mid \Sigma, R, D\big] \\ &\quad + \mathbb{E}\Big\{c^{\star\star}\big[(\boldsymbol{x}_1, \boldsymbol{Y}_{t_1}), (\boldsymbol{x}_2, \boldsymbol{Y}_{t_2})\big] \mid \Sigma, R, D\Big\}\Sigma \end{aligned} \tag{6b}$$

by applying the law of iterated expectations and relying upon properties of multi-Normal distributions. After integrating out the unknown parameters in Σ and R, expressions (6)

[†] An intuitive counterexample is provided by the case $f \colon \mathbb{R} \longmapsto \mathbb{R}^+,\ f(x) = x^2$.

in turn play an analogous role to their "static" counterparts (4) when attempting usual uncertainty or sensitivity analyses of $\boldsymbol{f}(\cdot)$ in a dynamic fashion. As regards the $(q\frac{q+1}{2})$-dimensional problem of marginalising Σ, in light of the dual origin of the Student's $\mathscr{T}_q$ distribution (see for instance [11], pg. 23) this was found to be efficiently dealt with by means of a simple univariate integration.

4. THE SHEFFIELD DYNAMIC GLOBAL VEGETATION MODEL: A TEST-BED

The Centre for Terrestrial Carbon Dynamics (*CTCD*) is a consortium of British academic and governmental institutions, established for the purpose of progressing scientific understanding of the role played by terrestrial ecosystems in the carbon cycle, with particular emphasis on forest ecosystems. The ultimate goals of the project are: to gauge carbon fluxes and their uncertainties at different space/time resolutions; to devise methodological, data and instrument advances for reducing these uncertainties; to deliver relevant findings in accessible formats to the scientific community and ultimately to policy makers. These tasks are pursued with the support of a variety of environmental models designed for simulating carbon patterns over different geographical and climatic scenarios. Unfortunately, such models suffer from coarse reproduction of some underlying physical processes and loose connections to driving data.

Bayesian statistical methods are being employed within the Centre for the assessment of relevant model (and data) developments required for reducing the uncertainty around them. In this setting, statistical challenges other than pure uncertainty and sensitivity analysis which presently require special care are: prediction, i.e. estimation of (possibly functionals of) model outputs at unavailable input configurations; screening, that is identification of which code inputs exert most significant influence on the outputs; and code verification, or detection of bugs in the actual implementation of the program.

Among the simulators devised and deployed within CTCD a central role is played by the Sheffield Dynamic Global Vegetation Model, daily version (henceforth *SDGVMd*). SDGVMd is aimed at illustrating possible 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. Inputs to SDGVMd comprise broad soil, vegetation and climate descriptors; outputs of the model include various measures of a site's carbon budget and miscellaneous environmental quantities. Additional challenges specifically offered by SDGVMd comprise a high-dimensional input space and the existence of embedded sub-modules operating at different time scales. A complete description of SDGVMd and the modules it incorporates can be found in [12].

For the purpose of illustrating the broad range of possibilities offered by Gaussian process-based meta-modelling, the soil module of SDGVMd (*CENTURY*: for details refer to [13]) was extracted and subject to emulation. In essence CENTURY manages the soil carbon (C) calculations within SDGVMd by recursively solving a set of independent PDEs, each being indexed by temperature, relative humidity and precipitation drivers and describing the monthly evolution of 8 different C pools. It should be stressed that

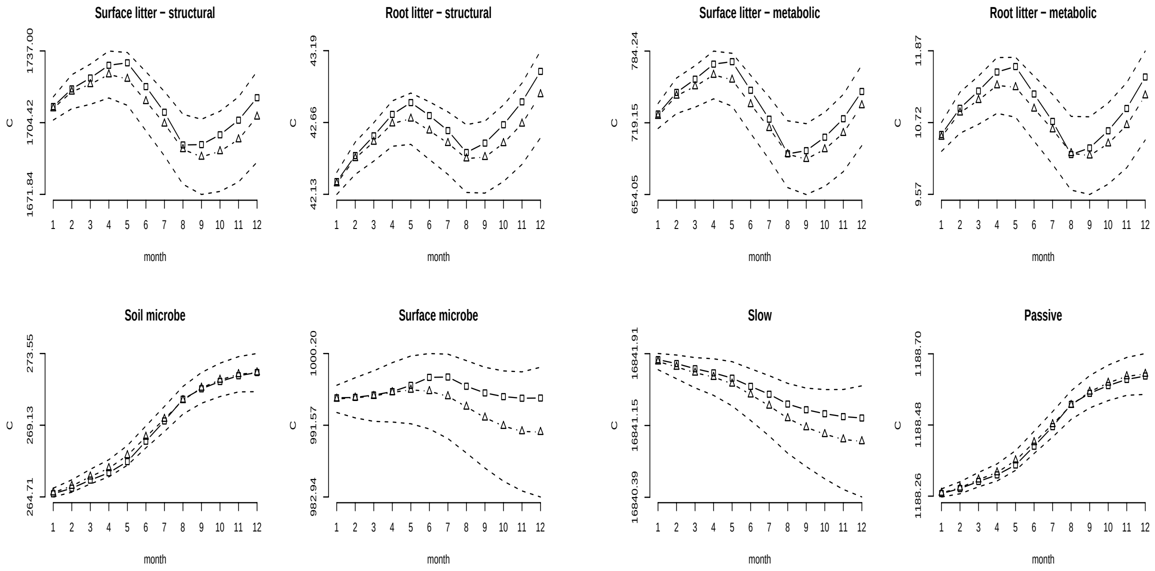

Figure 3. Simulated (—□—) vs. emulated ($\cdots\triangle\cdots$) C pools within 95% credible bounds ($- - -$).

in fact CENTURY is not a CPU-intensive model; nonetheless this feature conveniently enables straightforward evaluation of its emulator's performance.

Having thus recognised as an appropriate representation of CENTURY a function $\boldsymbol{f}: \mathbb{R}^{11} \longmapsto \mathbb{R}^{8}$, an interpolation exercise over a time period of 12 months was subsequently carried out. The input space was covered via a maximin Latin hypercube design of size $n = 200$; thereafter roughness parameters were estimated by the joint posterior mode based on vague i.i.d. Log-Logistic priors on $(r_1, \ldots, r_p)$. The prior mean was chosen to be linear, i.e. $\boldsymbol{h}(\boldsymbol{x}) = (1, x_1, \ldots, x_p)$, again for convenience. Figure 3 compares CENTURY's exact simulations for each C pool with their corresponding approximate posterior values from (6a), embedded within approximate 95% credible bounds.

A few comments are in order. It should be noticed how in most cases estimated interpolators appear to satisfactorily capture the underlying original outputs, apart from perhaps a couple of C pools (Surface microbe and Slow) where some drift can be observed to emerge over time. Additional effort should be placed into achieving somewhat narrower credible bounds for the interpolators, but provisional results look overall encouraging.

5. CONCLUSIONS

The paper focuses on two main goals. First, it deals with extensions to multi-response computer simulators of theoretical results already established for Bayesian meta-modelling of single-response codes. Second, it attempts to adapt the general methodology to encompass dynamic computer models within the same formal framework. Single-step and recursive emulation schemes were introduced and contrasted; preference towards the latter methodology was based on computational manageability. The proposed statistical machinery was then tested on the soil compartmental model embedded within SDGVMd: although the implemented exercise was limited to pure code interpolation, the analysis's outcome confirmed that interesting insights can be gained from applying the principle of Bayesian Gaussian process-based emulation to more sophisticated settings.

ACKNOWLEDGMENTS

The authors wish to thank Caitlin Buck for helpful suggestions. This research is financed by the UK Natural Environment Research Council, contract number F14/G6/105.

References

[1] 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.

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

[3] 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.

[4] 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.

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

[6] 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.

[7] J. A. Royle and D. Nychka. An algorithm for the construction of spatial coverage designs with implementation in SPLUS. Technical report, Department of Statistics, North Carolina State University, 1997.

[8] 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.

[9] 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.

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

[11] D. Gamerman, editor. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference.* Chapman & Hall. Chapman & Hall, London, 1997.

[12] M. R. Lomas, F. I. Woodward, and S. Quegan. The role of dynamic vegetation models. Technical report, Department of Animal and Plant Sciences, University of Sheffield, Sheffield UK, 2002.

[13] W. J. Parton, J. M. O. Scurlock, D.S. Ojima, T. G. Gilmanov, R. G. Scholes, D. S. Schimel, T. Kirchner, J.-C. Menaut, T. Seastedt, E. Garcia Moya, A. Kamnalrut, and J. I. Kinyamario. Observations and modelling of biomass and soil organic matter dynamics for the grassland biome worldwide. *Global Biogeochemical Cycles*, 7:785–809, 1993.