Number 4, pp. 783–816

A Simulation Approach to Bayesian Emulation of Complex Dynamic Computer Models

Sourabh Bhattacharya[*]

Abstract. In recent times, complex computer models have received wide attention in scientific research. However, in order to make conventional statistical statements regarding the scientific research, many expensive runs of the computer model are usually needed. New statistical theories, making their appearances, hold promise to alleviate the technical challenges. However, in cases where the underlying complex system is evolving with time, an effective theory for statistical analyses is lacking. In this paper, we propose a novel Bayesian methodology that extends the existing methodologies to the case of complex dynamic systems. The approach described in the paper exploits the recursive nature of dynamic simulation models to give a more efficient and accurate emulator.

The motivating example, although not a real model for any physical process, may be thought of as a proxy for a model representing climate change, where it is of interest to predict, over time t, the four-dimensional proxy time series $\mathbf{y}_t$ = (temperature, ice melting rate, barren land, CO_2 emission). Also available are proxy observations on deforestation, recorded over time; hence treated as known. The latter is known as *forcing input*, denoted by z_t. The computer model is treated as a *black box*.

Typically, Gaussian processes are used to model unknown computer models, which we adopt in our article. In order to exploit the recursive nature of dynamic computer models, we introduce a grid within the range of the unknown function where the entire dynamic sequence is *expected* to lie. This grid essentially defines a look-up table. Our proposed method then assumes that conditional on the response surface on the grid, and the available training data, the future responses are approximately independent. Exploiting the properties of Gaussian process, we justify our proposal theoretically and with ample simulation studies. We also apply our proposed methodology to the motivating example.

Key words: Gaussian process; Latin hypercube sampling; Markov property; matrix normal; parallel computing; spatial interpolation

1 Introduction

In scientific studies, mathematical models are widely used to describe complex physical processes. Actual implementation requires creating complex computer codes, which, given an input, produce an output. In other words, if x denotes input to a computer code denoted by $\eta(\cdot)$, then $y = \eta(x)$ is the corresponding output. The function $\eta(\cdot)$ is deterministic; that is, if $x_i = x_j$, then $\eta(x_i) = \eta(x_j)$. The underlying mathematical model may be highly complex; this may result in a code that is quite complicated, and very expensive in terms of computer run-time. In such cases, it becomes a challenging task, sometimes infeasible, to run the code at many different input values, which is often necessary for statistical analyses, such as sensitivity analysis (see Saltelli et al. (2000)) and calibration (see Kennedy and O'Hagan (2001)). One way around the technical challenge is to reduce the number of runs necessary by means of sophisticated Bayesian approaches. The essence of the idea, exploited in a series of papers, such as Haylock and O'Hagan (1996), Kennedy and O'Hagan (2001), Oakley and O'Hagan (2002), Oakley and O'Hagan (2004), is to regard the underlying code $\eta(\cdot)$ as unknown, modeled as a Gaussian process with a specified mean and covariance function, obtain training samples from the code by actually running the code at some pre-specified inputs points, and then obtain the posterior distribution of $\eta(\cdot)$ conditional on the training data set. This posterior distribution is called the *emulator* of the computer code. Note that, instead of running the actual computer code at all possible input values, it is only needed to run at some pre-specified input points to obtain the training data set. Outputs at other input values can be predicted using the posterior distribution, without running the expensive computer code. Thus, using an emulator of the expensive computer code in the form of the posterior distribution is many times more efficient than running the actual computer code at all possible input values.

C.1 Dynamic computer code

However, success of the Bayesian approach so far seems to have been limited to computer codes of the form $y = \eta(x)$, where for each given input x, there is a fixed output y. In reality, there are many examples of physical processes that evolve with time. In other words, the output from the code at time t may become an input to the code at time $t+1$; hence the computer model is dynamic in nature. Moreover, apart from the standard inputs, there may be "forcing inputs", describing external influences on the system and varying with time, and a "constant input", invariant with respect to time, and describing fundamental features of the underlying mathematical model. The constant input and the forcing inputs will be assumed to be recorded over time, or known. Denoting the forcing inputs at time t by w_t, the constant input by c and writing $z_t = (c, w_t)$, we note that in these situations, rather than the standard model $y = \eta(x)$, the dynamic model of the form $y_t = \eta(z_t, y_{t-1})$ is more appropriate. As an example, consider a dynamic vegetation model representing forest growth. The input for the vegetation model includes the biomass, height and leaf of the vegetation, the availability of water and nutrients, etc. These variables are assumed to be time varying and unknown, and it is of interest to predict the variables over time. The external forcing inputs at any time step comprise the weather conditions at that time, for example, temperature, precipitation, humidity, sunlight. Clearly, these influence the system, and are also time-varying. However, these are assumed recorded over time, unlike the variables of interest, biomass, height and leaf of the vegetation, which are not recorded and need to be predicted. Constants in this model include topographical data, soil characteristics and parameters of the model equations. One can treat these as invariant with respect to time. Here $\mathbf{y}_t = (\text{biomass, height, leaf, availability of water, nutrients})_t$

are the variables of interest that need to be predicted over time, the fixed constants $\mathbf{c}$ = (topographical data, soil characteristics, other parameters), and the forcing inputs, $\mathbf{w}_t$ = (temperature, precipitation, humidity, sunlight)$_t$, are assumed recorded over time. We write, $\mathbf{z}_t = (\mathbf{c}, \mathbf{w}_t)$. Based on the current input vector, the forcing inputs, and the constants, a single cycle of the code updates the input vector for the next cycle.

Apart from a suggestion in Kennedy and O'Hagan (2001) there seems to be no material available in the current statistical literature associated with dynamic computer codes. Moreover, the Kennedy-O'Hagan suggestion ignores forcing inputs and furthermore implies a restrictive correlation structure. In this paper we propose an approach that promises to deal appropriately and effectively with the problem of dynamic computer emulators. But before providing details of our theory for dynamic computer code emulation we briefly review the theory for emulation of non-dynamic computer models in Section 2.

2 Available Bayesian approach for non-dynamic computer code emulation

As indicated in Section 1, the computer code is treated as unknown (rather, as a black box), and statistically modeled as a Gaussian process with a specified mean and covariance structure. By assuming a model for the computer code, we are acknowledging the uncertainty about it, and the underlying Gaussian process model is expected to quantify the uncertainty accurately. It is usually assumed that $\eta(\cdot)$ is a Gaussian process with mean function $\mathbf{h}(\cdot)'\boldsymbol{\beta}$ (in particular, for any g, it is usually assumed that $\mathbf{h}(g) = (1, g)'$) and for any g_1, g_2 the covariance function is taken to be of the form $\sigma^2 c(g_1, g_2)$ where σ^2 is the process variance, $c(\cdot, \cdot)$ is the correlation function, usually taken as $c(g_1, g_2) = \exp\{-b(g_1 - g_2)^2\}$ (note that, for any g, $c(g, g) = 1$), and b is the smoothness parameter, the latter being associated with the smoothness of the process realizations. The above correaltion function implicitly assumes that the underlying model for the computer code is infinitely smooth. For details on issues related to smoothness of a random process and its connection to the associated smoothness parameter, see, for example, Stein (1999).

Suppose that the computer code is actually run only at input (grid) points $\mathbf{G} = (g_1, \ldots, g_n)'$ (n may be small for expensive codes) to obtain outputs $(\eta(g_1), \ldots, \eta(g_n))'$. This is the training data set, which we denote by $\mathbf{D}$. Here we note that the outputs in $\mathbf{D}$ are observed without error, since they arise from the true computer model, not from the Gaussian process representation of the computer model. However, the actual mathematical formulation of the problem treats even the training data set as a realization from the Gaussian process. It will be explained subsequently why this assumption makes good sense.

Now, if prediction of the output at any point $g^{(new)}$ is desired, then one simply computes the posterior $[\eta(g^{(new)}) \mid \mathbf{D}]$. If $\mathbf{A}_D$ is the n-dimensional correlation matrix corresponding to the training data set $\mathbf{D}$, where $c(g_i, g_j)$ is the (i, j)th element, and if the vector $(c(\cdot, g_1), \ldots, c(\cdot, g_n))'$ is denoted by $\mathbf{s}_D(\cdot)$ and $\mathbf{H}'_D = (\mathbf{h}(g_1), \ldots, \mathbf{h}(g_n))$, then since $(\eta(g^{(new)}), \mathbf{D})$ is assumed to be a realisation from the Gaussian process, it is jointly multivariate normal with mean vector $\begin{pmatrix} \mathbf{h}(g^{(new)})'\boldsymbol{\beta} \\ \mathbf{H}_D\boldsymbol{\beta} \end{pmatrix}$ and covariance matrix $\sigma^2 \left\{ \begin{pmatrix} 1 & \mathbf{s}_D(g^{(new)})' \\ \mathbf{s}_D(g^{(new)}) & \mathbf{A}_D \end{pmatrix} \right\}$. Hence, it follows that the conditional distribution $[\eta(g^{(new)}) \mid \mathbf{D}]$ is normal with mean $\mathbf{h}(g^{(new)})'\boldsymbol{\beta} + (\mathbf{D} - \mathbf{H}_D\boldsymbol{\beta})'\mathbf{A}_D^{-1}\mathbf{s}_D(g^{(new)})$ and variance $\sigma^2\{1 - \mathbf{s}_D(g^{(new)})'\mathbf{A}_D^{-1}\mathbf{s}_D(g^{(new)})\}$. This posterior is marginalised with respect to posteriors of $\boldsymbol{\beta}$ and σ^2 to arrive at a Student's t distribution. It is only needed to estimate the smoothness parameter b, which is usually done by maximum likelihood methods; we provide details in Section 5. In other words, the posterior predictive distribution is actually derived conditional on the maximum likelihood estimate of b.

Note that, conditioning on $\mathbf{D}$ forces the simulated function to pass through the points in $\mathbf{D}$. This is because, due to a well-known property of predictors in the case of Gaussian processes (for a simple exposition, see, for example, pages 56–57 or page 93 of Santner et al. (2003)), the conditional $[\eta(g^{(new)}) \mid \mathbf{D}]$ has zero variance if $g^{(new)}$ equals any input g_i of $\mathbf{D}$. So, since $\mathbf{D}$ is observed without error, forcing the simulated function to pass through points in $\mathbf{D}$ makes sense. Experiments have indicated that compared to the standard Monte Carlo approach, which requires many runs of the code, the Bayesian approach described above is many times more efficient.

The remaining part of our paper is structured as follows. In Section 3 we provide the motivation of our idea for dynamic computer code emulation using concepts related to non-random functions. Extension of the concepts described in Section 3 for posterior simulation of any one-dimensional random dynamic sequence, based on Gaussian process, is outlined in Section 4. Further development, and illustration of the proposed methodology with a one-dimensional function is detailed in Section 5. We provide formal and complete details of general, possibly multivariate dynamic computer code emulation in a Bayesian set up in Section 6. In Section 7 we assess the performance of our proposed methodology on a real four dimensional dynamic computer model. We conclude in Section 8.

3 Motivation of the proposed simulation idea for dynamic computer code emulation

To gain the necessary intuition, let us first consider a known function $f(\cdot)$. Here we use a notation different from $\eta(\cdot)$ to emphasize that unlike $\eta(\cdot)$, which is a random function, $f(\cdot)$ is a completely known function. In order to obtain a dynamic sequence of length T, which is of the form $x_1 = f(x_0), x_2 = f(x_1), \ldots, x_T = f(x_{T-1})$, starting with x_0 (here again we use a different notation x instead of y to denote non-random dynamic sequence), two methods can be envisaged –

(a) Simply use the formula $x_t = f(x_{t-1})$, and

(b) Evaluate first $x_1 = f(x_0)$; then evaluate $f(x)$ for all $x \neq x_0$. This yields a complete look-up table of the form $\{x, f(x)\}$. Then, since x_1 has been obtained, one can simply find x_1 in the first column of the look-up table. Then, the corresponding row of the second column of the look-up table is the required $x_2 = f(x_1)$. One simlarly obtains $x_3 = f(x_2)$. Thus, in principle, this procedure can be repeated to obtain the entire dynamic sequence. This can be achieved exactly in practice if the input-output space is discrete and finite. For continuous input-output spaces the function can not be evaluated at all possible points. However, it is possible to evaluate the function on a fine grid and interpolate within the resulting look-up table to obtain an approximate version of the dynamic sequence. The accuracy of the approximation depends upon how fine the grid is. Observe that in both cases (a) and (b), given x_{t-1} and the function f, x_t depends only upon x_{t-1}. Intuitively, this seems to suggest Markov property for corresponding dynamic emulators based on random processes.

In this paper, we are interested in situations where the true, deterministic function is unknown (treated as a black box), or too complicated to evaluate at many input values. As a result, uncertainty regarding the unknown function is quantified by a random process $\eta(\cdot)$; as assumed in the literature for modeling computer output, in this methodological paper we also assume that $\eta(\cdot)$ is a Gaussian process. Although method (a) described above does not seem to provide the mathematical intuition necessary for generating random dynamic sequences from $\eta(\cdot)$ (since $\eta(\cdot)$ is random, we can not use the formula $y_t = \eta(y_{t-1})$ analogous to $x_t = f(x_{t-1})$), it will be demonstrated that method (b) can be generalized in the case of random $\eta(\cdot)$ to generate random dynamic sequences of the form $y_1 = \eta(y_0), y_2 = \eta(y_1), \ldots, y_T = \eta(y_{T-1})$. In this case, since $\eta(\cdot)$ is modeled as continuous Gaussian process, a grid will be necessary. Although it may apparently seem that this will increase a lot of computational burden and may give rise to a lot of numerical problems, it will be demonstrated that our proposed methodology is remarkably stable numerically, and not much additional computational cost is involved. In addition, the methodology we propose is very much amenable to parallel computing.

4 Dynamic simulation in the random function case given the training data set D

As before, we assume that $\eta(\cdot)$ is a Gaussian process with mean function $\mathbf{h}(\cdot)'\boldsymbol{\beta}$ and covariance function $\sigma^2 c(g_1, g_2)$, where $c(g_1, g_2) = \exp\{-b(g_1 - g_2)^2\}$. Given a training data set $\mathbf{D}$ from the true, unknown function, and an initial input value y_0, it is necessary to simulate the dynamic sequence $y_1 = \eta(y_0), y_2 = \eta(y_1), y_3 = \eta(y_2), \ldots, y_T = \eta(y_{T-1})$. In other words, simulation from the joint posterior distribution $[y_1 = \eta(y_0), y_2 = \eta(y_1), y_3 = \eta(y_2), \ldots, y_T = \eta(y_{T-1}) \mid \mathbf{D}]$ is needed. In order to clarify the key idea we initially assume that values of all the parameters, namely, $\boldsymbol{\beta}, \sigma^2$ and b, are known.

Analogically, as in the case of method (b), we will first simulate the first random variable of the sequence $y_1 = \eta(y_0)$, given $\mathbf{D}$. The theory reviewed in Section 2 shows that the conditional distribution $[\eta(y_0) \mid \mathbf{D}]$ is normal with mean $\mathbf{h}(y_0)'\boldsymbol{\beta} + \mathbf{s}(y_0)'\mathbf{A}_D^{-1}(\mathbf{D} - \mathbf{H}_D\boldsymbol{\beta})$ and variance $\sigma^2\{1 - \mathbf{s}(y_0)'\mathbf{A}_D^{-1}\mathbf{s}(y_0)\}$, where the notation is as before. Hence, the distribution of the first random variable of the dynamic sequence is completely known.

To proceed with simulation of the remaining dynamic sequence, we first need to construct a grid $\mathbf{G}^* = (g_1^*, \ldots, g_N^*)'$ which is fine enough (the term 'fine' is arbitrary at this moment; we will return to this subsequently) to approximate the input space. On the grid $\mathbf{G}^*$ we define a vector of latent random variables $\mathbf{D}^* = (\eta(g_1^*), \ldots, \eta(g_N^*))'$. Evidently, due to the Gaussian process assumption, $\mathbf{D}^*$ is jointly distributed as multivariate normal.

Given $y_1 = \eta(y_0)$, we simulate $\mathbf{D}^*$ from $[\mathbf{D}^* \mid \eta(y_0) = y_1, \mathbf{D}]$. This is an N-variate normal distribution with mean vector

$$\mathbf{E}[\mathbf{D}^* \mid \eta(y_0) = y_1, \mathbf{D}] = \mathbf{H}_{D^*}\boldsymbol{\beta} + \mathbf{A}^{(12)}_{(D^*,D)} \left\{\mathbf{A}^{(22)}_{(D^*,D)}\right\}^{-1} \left\{\begin{pmatrix} \mathbf{D} \\ y_1 \end{pmatrix} - \begin{pmatrix} \mathbf{H}_D\boldsymbol{\beta} \\ \mathbf{h}(y_0)'\boldsymbol{\beta} \end{pmatrix}\right\} \tag{C.1}$$

and covariance matrix

$$\mathrm{var}[\mathbf{D}^* \mid \eta(y_0) = y_1, \mathbf{D}] = \sigma^2 \left\{\mathbf{A}^{(11)}_{(D^*,D)} - \mathbf{A}^{(12)}_{(D^*,D)} \left(\mathbf{A}^{(22)}_{(D^*,D)}\right)^{-1} \mathbf{A}^{(21)}_{(D^*,D)}\right\} \tag{C.2}$$

In the above,

$$\mathbf{A}^{(11)}_{(D^*,D)} = \mathbf{A}_{D^*}, \tag{C.3}$$

$$\mathbf{A}^{(22)}_{(D^*,D)} = \begin{pmatrix} \mathbf{A}_D & \mathbf{s}(y_0) \\ \mathbf{s}(y_0)' & 1 \end{pmatrix}, \tag{C.4}$$

$$\mathbf{A}^{(12)}_{(D^*,D)} = \begin{pmatrix} c(g_1^*, g_1) & c(g_1^*, g_2) & \ldots & c(g_1^*, g_n) & c(g_1^*, y_0) \\ c(g_2^*, g_1) & c(g_2^*, g_2) & \ldots & c(g_2^*, g_n) & c(g_2^*, y_0) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c(g_N^*, g_1) & c(g_N^*, g_2) & \ldots & c(g_N^*, g_n) & c(g_N^*, y_0) \end{pmatrix} \tag{C.5}$$

Note that the above conditional distribution follows from the fact that the joint distribution of $(\mathbf{D}^*, \mathbf{D}, \eta(y_0))$ is multivariate normal with mean vector $\begin{pmatrix} \mathbf{H}_{D^*}\boldsymbol{\beta} \\ \mathbf{H}_D\boldsymbol{\beta} \\ \mathbf{h}(y_0)'\boldsymbol{\beta} \end{pmatrix}$ and covariance matrix $\sigma^2\mathbf{A}_{(D^*,D)}$ where

$$\mathbf{A}_{(D^*,D)} = \begin{pmatrix} \mathbf{A}^{(11)}_{(D^*,D)} & \mathbf{A}^{(12)}_{(D^*,D)} \\ \mathbf{A}^{(21)}_{(D^*,D)} & \mathbf{A}^{(22)}_{(D^*,D)} \end{pmatrix} \tag{C.6}$$

Once a realization of $\mathbf{D}^*$ is obtained from the above conditional, we then simulate y_{t+1} from $[\eta(y_t) \mid \mathbf{D}^*, \mathbf{D}, y_t]$, which, for each $t = 1, \ldots, T-1$, is a univariate normal distribution with mean and variance given, respectively, by

$$\mu_t = \mathbf{h}(y_t)'\boldsymbol{\beta} + (\mathbf{D}_{n+N} - \mathbf{H}_{(D^*,D)}\boldsymbol{\beta})'\mathbf{A}^{-1}_{(D^*,D)}\mathbf{s}_{(D^*,D)}(y_t) \tag{C.7}$$

$$\sigma_t^2 = \sigma^2\{1 - \mathbf{s}_{(D^*,D)}(y_t)'\mathbf{A}^{-1}_{(D^*,D)}\mathbf{s}_{(D^*,D)}(y_t)\} \tag{C.8}$$

In the above, $\mathbf{D}_{n+N} = \begin{pmatrix} \mathbf{D} \\ \mathbf{D}^* \end{pmatrix}$ is the augmented matrix,

$$\mathbf{H}'_{(D^*,D)} = [\mathbf{h}(g_1), \ldots, \mathbf{h}(g_n), \mathbf{h}(g_1^*), \ldots, \mathbf{h}(g_N^*)],$$
$$\mathbf{s}_{(D^*,D)}(\cdot) = [c(\cdot, g_1), \ldots, c(\cdot, g_n), c(\cdot, g_1^*), \ldots, c(\cdot, g_N^*)]'$$

The resulting sequence is an approximate realization from the posterior distribution $[y_1 = \eta(y_0), y_2 = \eta(y_1), \ldots, y_T = \eta(y_{T-1}) \mid \mathbf{D}]$, as required. The procedure must be repeated many times to generate as many random sequences as desired to learn the posterior distribution of the dynamic sequence. Obviously, each random sequence is independent of each other. A very important computational property follows, thanks to this independence of the random sequences. The independence renders our dynamic emulator parallelisable, that is, the random sequences can be generated in independent parallel processors, although the original dynamic computer code may not be parallelisable. This completes the basic formulation of the problem and our methodology.

As already indicated in Section 2, note here that, if the grid $\mathbf{G}^*$ already contains y_t, then $\mathbf{D}^*$ contains $\eta(y_t)$ and $[\eta(y_t) \mid \mathbf{D}^*, y_t] = \delta_{\eta(y_t)}$, where δ_x denotes point mass at x. In other words, the conditional gives point mass to $\eta(y_t)$. However, since y_t is a continuous random variable and $\mathbf{G}^*$ is a set of discrete points, $y_t \notin \mathbf{G}^*$ with probability 1, and a simulation from $[\eta(y_t) \mid \mathbf{D}^*, y_t]$ can be viewed as spatial interpolation within the set $\mathbf{D}^*$; see, for example, Cressie (1993), Stein (1999). By making $\mathbf{G}^*$ more and more fine the above conditional can be made to approach $\delta_{\eta(y_t)}$.

In keeping with the motivation obtained from deterministic methods (a) and (b) of obtaining the true dynamic sequence, we assume that, given $\mathbf{D}^*$, simulation of the dynamic sequence exhibits Markov property. In other words, we assume that simulation of $y_{t+1} = \eta(y_t)$ given function values $\mathbf{D}^*$ and the training data set $\mathbf{D}$ depends only upon y_t. That is, $[\eta(y_t) \mid \mathbf{D}^*, \mathbf{D}, y_t = \eta(y_{t-1}), y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0)] = [\eta(y_t) \mid \mathbf{D}^*, \mathbf{D}, y_t]$. Although this is an assumption associated with our methodology, it will be shown that our proposal yields very accurate predictions. We remark that, $\mathbf{D}^*$ may also be interpreted as a set of latent/auxiliary variables brought in to facililate simulation. However, marginalization over $\mathbf{D}^*$ does not preserve this Markov property, and computation becomes difficult and unstable. Details are provided in the Appendix.

So far we have assumed that the parameters, $\boldsymbol{\beta}, \sigma^2$ and the smoothness parameter b, are known. Moreover, we have not yet described how the forcing inputs z_t are related to our proposed methodology, or the cases where the random function of interest is multidimensional. In Section 5 we illustrate our methodology, providing further necessary methodological details, on a simple one-dimensional function, with one-dimensional input space. In Section 6 we extend the methodology to the case of multidimensional functions, with multidimensional input space. Forcing inputs (which may also be multidimensional) are incorporated within the general methodology that we develop in Section 6.

5 Assessment of the performance of the proposed method using a one-dimensional function with a single input

In this section we assume that the true function is given by $f(x) = \cos(x + \sin(x))$, and that starting with an initial value, prediction of the dynamic sequence induced by the above function is needed. This function has been used by SenGupta and Ugwuowo

(2006) in the context of circular-linear regression modeling. Observe that this function is differentiable of all orders (analytic), and hence, infinitely smooth. This example thus fits our assumption, and with this we demonstrate our approach, clarifying further methodological details as we proceed.

C.1 Training data set

Using Latin hypercube sampling, we obtain $n = 6$ inputs required for creating the training data set $\mathbf{D}$ of size $n = 6$. These are $g_1 = -0.258, g_2 = 0.066, g_3 = 0.223, g_4 = 0.393, g_5 = 0.781, g_6 = 0.960$. Corresponding to these inputs, the outputs obtained by directly evaluating the true function $f(\cdot)$ at the input values are $\cos(g_1 + \sin(g_1)) = 0.871, \cos(g_2+\sin(g_2)) = 0.991, \cos(g_3+\sin(g_3)) = 0.903, \cos(g_4+\sin(g_4)) = 0.714, \cos(g_5+\sin(g_5)) = 0.085, \cos(g_6 + \sin(g_6)) = -0.207$. Hence, the training data set $\mathbf{D}$ is observed without error.

We now assume that, starting with the initial value y_0, we need to predict the dynamic sequence $y_t = f(y_{t-1})$, pretending that the function $f(\cdot)$ is unknown. As before, we denote by $\eta(\cdot)$ the Gaussian process model for the unknown function. The Gaussian process has mean function $\mathbf{h}(\cdot)'\boldsymbol{\beta}$ (where $\mathbf{h}(g) = (1, g)'$ for any g) and covariance function of the form $\sigma^2 c(g_i, g_j)$, where $c(g_i, g_j) = \exp\left\{-b(g_i - g_j)^2\right\}$ for any i, j, and b is the smoothness parameter.

C.2 Formulation of the problem

With the above set up, the likelihood of the unknown parameters $(\boldsymbol{\beta}, \sigma^2, b)$, given the training data set $\mathbf{D}$ corresponds to an n-variate Gaussian density, the logarithm of which is, up to a constant, given by

$$\ell(\boldsymbol{\beta}, \sigma^2, b) = -n\log(\sigma^2) - \log(|\mathbf{R}|) - (\mathbf{D} - \mathbf{H}_D\boldsymbol{\beta})'\mathbf{R}^{-1}(\mathbf{D} - \mathbf{H}_D\boldsymbol{\beta})/\sigma^2 \tag{C.9}$$

In the above equation, $\mathbf{R}$ depends upon the smoothness parameter b.

Since it is generally extremely difficult to elicit meaningful prior information about the unknown parameters $(\boldsymbol{\beta}, \sigma^2)$, the prior distribution is chosen to be non-informative. Specifically, we choose the prior as

$$\pi(\boldsymbol{\beta}, \sigma^2) \propto \frac{1}{\sigma^2} \tag{C.10}$$

Marginalizing the conditional $[\eta(y_0) \mid \mathbf{D}, \boldsymbol{\beta}, \sigma^2, b]$ with respect to posteriors of $\boldsymbol{\beta}$ and σ^2, we obtain, $[\eta(y_0) \mid \mathbf{D}, b] \sim \mathcal{T}_1(\mu_2(y_0), c_2(y_0, y_0)\hat{\sigma}^2_{GLS}, n-2)$, a univariate Student's t distribution where the parameters are given by

$$\begin{aligned} \mu_2(y_0) &= \mathbf{h}(y_0)'\hat{\boldsymbol{\beta}}_{GLS} + (\mathbf{D} - \mathbf{H}_D\hat{\boldsymbol{\beta}}_{GLS})'\mathbf{A}_D^{-1}\mathbf{s}_D(y_0) && \text{(C.11)} \\ c_2(y_0, y_0) &= 1 - \mathbf{s}_D(y_0)\mathbf{A}_D^{-1}\mathbf{s}(y_0) \\ &+ [\mathbf{h}(y_0) - \mathbf{H}'_D\mathbf{A}_D^{-1}\mathbf{s}_D(y_0)]'(\mathbf{H}'_D\mathbf{A}_D^{-1}\mathbf{H}_D)^{-1} \\ &\quad [\mathbf{h}(y_0) - \mathbf{H}'_D\mathbf{A}_D^{-1}\mathbf{s}_D(y_0)] && \text{(C.12)} \end{aligned}$$

In the above,

$$\hat{\boldsymbol{\beta}}_{GLS} = (\mathbf{H}_D'\mathbf{A}_D^{-1}\mathbf{H}_D)^{-1}(\mathbf{H}_D'\mathbf{A}_D^{-1}\mathbf{D})$$

and

$$(n-2)\hat{\sigma}^2_{GLS} = (\mathbf{D} - \mathbf{H}_D\hat{\boldsymbol{\beta}}_{GLS})'\mathbf{A}_D^{-1}(\mathbf{D} - \mathbf{H}_D\hat{\boldsymbol{\beta}}_{GLS}).$$

As estimate of b can be obtained by maximizing (C.9) with respect to b after substituting $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_{GLS}$ and $\sigma^2 = \hat{\sigma}^2_{GLS}$ in (C.9). As a result of the substitutions, the log-likelihood takes the simplified form

$$\ell(b) = -n\log(\hat{\sigma}^2_{GLS}) - \log(|\mathbf{R}|) \tag{C.13}$$

This form has been recommended by Santner et al. (2003). Note again, that both $\hat{\sigma}^2_{GLS}$ and $\mathbf{R}$ depend upon the smoothness parameter b.

Letting $\mathbf{D}^* = (\eta(g_1^*), \ldots, \eta(g_N^*))'$, we note that the conditional distribution of $[\mathbf{D}^* \mid b, \mathbf{D}, \eta(y_0)]$ is multivariate Student's t distribution. It is easy to describe this multivariate distribution using univariate conditional distributions, which are one-dimensional analogues of the multivariate version presented in Section C.4. In other words, we need the conditional distribution of $[\eta(g_j^*) \mid b, \mathbf{D}, \eta(y_0), \eta(g_1^*), \ldots, \eta(g_{j-1}^*)]$, for $j = 1, \ldots, N(= 10)$. Define

$$\mathbf{D}_j^{((n+j)\times 1)} = (\eta(g_1), \ldots, \eta(g_n), \eta(y_0), \eta(g_1^*), \ldots, \eta(g_{j-1}^*))' \tag{C.14}$$

$$\mathbf{H}_{j,D}' = [\mathbf{h}(g_1), \ldots, \mathbf{h}(g_n), \mathbf{h}(y_0), \mathbf{h}(g_1^*), \ldots, \mathbf{h}(g_{j-1}^*)] \tag{C.15}$$

$$\mathbf{s}_{(j,D)}(\cdot) = [c(\cdot, g_1), \ldots, c(\cdot, g_n), c(\cdot, y_0), c(\cdot, g_1^*), \ldots, c(\cdot, g_{j-1}^*)]' \tag{C.16}$$

$\mathbf{A}_{(j,D)} =$

$$\begin{pmatrix} c(g_1,g_1) & \ldots & c(g_1,g_n) & c(g_1,y_0) & c(g_1,g_1^*) & \ldots & c(g_1,g_{j-1}^*) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c(g_n,g_1) & \ldots & c(g_n,g_n) & c(g_n,y_0) & c(g_n,g_1^*) & \ldots & c(g_n,g_{j-1}^*) \\ c(y_0,g_1) & \ldots & c(y_0,g_n) & c(y_0,y_0) & c(y_0,g_1^*) & \ldots & c(y_0,g_{j-1}^*) \\ c(g_1^*,g_1) & \ldots & c(g_1^*,g_n) & c(g_1^*,y_0) & c(g_1^*,g_1^*) & \ldots & c(g_1^*,g_{j-1}^*) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c(g_{j-1}^*,g_1) & \ldots & c(g_{j-1}^*,g_n) & c(g_{j-1}^*,y_0) & c(g_{j-1}^*,g_1^*) & \ldots & c(g_{j-1}^*,g_{j-1}^*) \end{pmatrix} \tag{C.17}$$

Then it follows that

$$[\eta(g_j^*) \mid b, \mathbf{D}, \eta(y_0), \eta(g_1^*), \ldots, \eta(g_{j-1}^*)] \sim \mathcal{T}_1\left(\mu_{2,j}(g_j^*), c_{2,j}(g_j^*, g_j^*)\hat{\sigma}^2_{GLS,j}; n+j-2\right) \tag{C.18}$$

which is a univariate Student's t distribution with location parameter $\mu_{2,j}(g_j^*)$, scale parameter $c_{2,j}(g_j^*, g_j^*)\hat{\sigma}^2_{GLS,j}$, and $n+j-2$ degrees of freedom. The form of the parameters are given below.

$$\begin{aligned}\mu_{2,j}(\cdot) &= \mathbf{h}(\cdot)'\hat{\boldsymbol{\beta}}_{GLS,j} + (\mathbf{D}_j - \mathbf{H}_{j,D}\hat{\boldsymbol{\beta}}_{GLS,j})'\mathbf{A}_{j,D}^{-1}\mathbf{s}_{j,D}(\cdot) &\text{(C.19)}\\ c_{2,j}(g_1,g_2) &= c_{1,j}(g_1,g_2) + [\mathbf{h}(g_1) - \mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{s}_{j,D}(g_1)]'(\mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{H})^{-1}\\ &\quad [\mathbf{h}(g_2) - \mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{s}_{j,D}(g_2)] &\text{(C.20)}\end{aligned}$$

In the above, $c_{1,j}(g_1,g_2) = c(g_1,g_2) - \mathbf{s}_{(j,D)}(g_1)'\mathbf{A}_{(j,D)}^{-1}\mathbf{s}_{(j,D)}(g_2)$, and

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{GLS,j} &= (\mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{H}_{j,D})^{-1}(\mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{D}_j) &\text{(C.21)}\\ (n+j-2)\hat{\sigma}^2_{GLS,j} &= (\mathbf{D}_j - \mathbf{H}_{j,D}\hat{\boldsymbol{\beta}}_{GLS,j})'\mathbf{A}_{j,D}^{-1}(\mathbf{D}_j - \mathbf{H}_{j,D}\hat{\boldsymbol{\beta}}_{GLS,j}) &\text{(C.22)}\end{aligned}$$

In fact, the distribution of $[\mathbf{D}^* \mid b, \mathbf{D}, \eta(y_0)]$ is a multivariate Student's t distribution, and it is possible to simulate directly from any multivariate Student's t distribution, instead of simulating successively from the conditionals (C.18). However, our intention is to generalize, in a straightforward manner, this methodology of simulating dynamic sequences from one-dimensional functions to simulating dynamic sequences from multi-dimensional functions, which we provide in Section 6. In the latter case, the Student's t distribution will be matrix-variate, and there it is easy to simulate from the successive conditionals, which are multivariate Student's t distributions.

For $t = 2, \ldots, T$, one can show, by marginalizing the conditional distribution $[\eta(y_{t-1}) \mid \mathbf{D}^*, \mathbf{D}, \boldsymbol{\beta}, \sigma^2, b]$ with respect to the posterior $[\boldsymbol{\beta}, \sigma^2 \mid \mathbf{D}, \mathbf{D}^*, b]$ that,

$$[\eta(y_{t-1}) \mid b, \mathbf{D}, \mathbf{D}^*] \sim \mathcal{T}_1\left(\mu_{2,n+N}(\cdot), c_{2,n+N}(\cdot,\cdot)\hat{\sigma}^2_{GLS,n+N}; n+N-2\right) \tag{C.23}$$

In the above,

$$\begin{aligned}\mu_{2,n+N}(\cdot) &= \mathbf{h}(\cdot)'\hat{\boldsymbol{\beta}}_{GLS,n+N} + (\mathbf{D}_{n+N} - \mathbf{H}_{(D^*,D)}\hat{\boldsymbol{\beta}}_{GLS,n+N})'\mathbf{A}_{(D^*,D)}^{-1}\mathbf{s}_{(D^*,D)}(\cdot) &\text{(C.24)}\\ c_{2,n+N}(g_1,g_2) &= c_{1,n+N}(g_1,g_2) + [\mathbf{h}(g_1) - \mathbf{H}'_{(D^*,D)}\mathbf{A}_{(D^*,D)}^{-1}\mathbf{s}_{(D^*,D)}(g_1)]'\\ &\quad (\mathbf{H}'_{(D^*,D)}\mathbf{A}_{(D^*,D)}^{-1}\mathbf{H}_{(D^*,D)})^{-1}[\mathbf{h}(g_2) - \mathbf{H}'_{(D^*,D)}\mathbf{A}_{(D^*,D)}^{-1}\mathbf{s}_{(D^*,D)}(g_2)] &\text{(C.25)}\end{aligned}$$

In (C.25), $c_{1,n+N}(g_1,g_2) = c(g_1,g_2) - \mathbf{s}_{(D^*,D)}(g_1)'\mathbf{A}_{(D^*,D)}^{-1}\mathbf{s}_{(D^*,D)}(g_2)$.

$\hat{\boldsymbol{\beta}}_{GLS,n+N} =$

$$(\mathbf{H}'_{(D^*,D)}\mathbf{A}_{(D^*,D)}^{-1}\mathbf{H}_{(D^*,D)})^{-1}(\mathbf{H}'_{(D^*,D)}\mathbf{A}_{(D^*,D)}^{-1}\mathbf{D}_{n+N}) \tag{C.26}$$

$(n+N-2)\hat{\sigma}^2_{GLS,n+N} =$

$$(\mathbf{D}_{n+N} - \mathbf{H}_{(D^*,D)}\hat{\boldsymbol{\beta}}_{GLS,n+N})'\mathbf{A}_{(D^*,D)}^{-1}(\mathbf{D}_{n+N} - \mathbf{H}_{(D^*,D)}\hat{\boldsymbol{\beta}}_{GLS,n+N}) \tag{C.27}$$

C.3 Implementation and results

We return to the univariate example where the true function is $f(x) = \cos(x + \sin(x))$, which yielded the training data set $\mathbf{D}$. We now need to predict dynamic sequences corresponding to this function using the Gaussian process based statistical methodology developed above, without using the function $f(x) = \cos(x + \sin(x))$ any further. Note that, in order to implement our methodology, we need to estimate the smoothness parameter b. In fact, it is possible to compute a posterior distribution of b, but it is computationally convenient to estimate b using maximum likelihood methods, and then treat the estimated value, denoted by $\hat{b}$ as a fixed constant. Kennedy and O'Hagan (2001) found that the uncertainty in the smoothness parameter b is not very important.

The log-likelihood (C.13) is shown in Figure 1. We chose as the maximum likelihood estimate of b the value which approximately maximizes the log-likelihood, and also avoids numerical instabilities. The choice $\hat{b} = 50$ seems to meet these requirements. However, we have also experimented with many other plausible choices of b; the results were quite robust to these choices.

We describe implementation of our methodology with two dynamic sequences obtained from the same function $f(x) = \cos(x + \sin(x))$, started with two different initial values of y_0.

To obtain the grid $\mathbf{G}^*$, we fix the grid size $N = 20$, and again use Latin hypercube sampling to obtain $\mathbf{G}^* = \{-0.291, -0.179, -0.125, -0.066, -0.032, 0.080, 0.102, 0.186, 0.224, 0.340, 0.352, 0.468, 0.483, 0.549, 0.668, 0.722, 0.786, 0.813, 0.890, 0.999\}$.

We also experimented with grid size N varying between 20 and 30, with different sets $\mathbf{G}^*$ of increasing size, but the final prediction results remained almost exactly the same as those obtained with $N = 20$ and the particular choice of $\mathbf{G}^*$ as given above. We conclude that $N = 20$ is adequate for this experiment.

Starting with the initial value $y_0 = 0.55$ we simulated 500 dynamic sequences, following our proposal. The true sequence obtained by directly using the formula $y_t = \cos(y_{t-1} + \sin(y_{t-1}))$ is compared with the summaries of the dynamic posterior simulation. The results, for length of the dynamic sequence $T = 20$, are displayed in Figure 2. In the figure, the thin, solid line stands for the true dynamic sequence. The dotted line denotes the mean posterior dynamic sequence, and the thick, black lines are the approximate 95% credible intervals. Clearly, the entire true sequence falls within the approximate 95% credible intervals, with the mean posterior sequence estimating the true sequence quite accurately. In fact, although we have shown results for $T = 20$ time points only, this is due to clarity of visualization. We have obtained results for $T = 100$ (in fact, could have obtained predictions for any value of T, however large) and the results have been as ecouraging as the first 20 time points. We also attempted to predict the dynamic sequence after analytically integrating out $\mathbf{D}^*$, and using the simplified form of the marginal posterior corresponding to (C.67), as provided in the Appendix. However, just after $T = 6$, numerical problems appear and the methodology completely breaks down. Also, we could not produce more than 100 posterior dynamic sequences due to numerical problems. Figure 3 shows the predictions of the first 6 values of the dynamic sequence, based on 100 simulations. We note, after comparing Figures 2 and 3, that, as discussed in the Appendix, the posterior predictions obtained after integrating out $\mathbf{D}^*$, although limited, are very much similar to those obtained by retaining $\mathbf{D}^*$. However, since integrating out $\mathbf{D}^*$ seems to be useless for practical purposes, we shall not pursue the corresponding methodology any further.

We now consider a second experiment, with a dynamic sequence starting at $y_0 = 1.4$. Figure 4 shows the results of this experiment, with $T = 20$ (again, we had conducted the experiment for $T = 100$, but just for the sake of visualization, we chose to display results corresponding to 20 time points only). Somewhat surprisingly, the predictions this time are very inaccurate, with most part of the true dynamic sequence not falling within the approximate 95% credible intervals. An investigation of this phenomenon led to some important observations.

For both the experiments, we plotted the training data as well as the true dynamic sequence on the true function $f(x) = \cos(x+\sin(x))$. The function and the training data set are certainly the same in both cases, but the dynamic sequences in the two cases are different, due to the different initial values. The diagrams corresponding to initial values $y_0 = 0.55$ and $y_0 = 1.4$ are shown in Figures 5 and 6 respectively. The circles in the graphs indicate the plots of $y_t; t = 2, \ldots, T$ against the corresponding input values $y_t; t = 1, \ldots, T-1$. The plus signs denote the plots of the training data $f(g_1), \ldots, f(g_n)$ against the corresponding input values $g_1, \ldots, g_n$. From Figure 5, which corresponds to $y_0 = 0.55$, it is clear that the training data provides a lot of information about the dynamic sequence, in that both the training data and the true dynamic sequence come from the same part of the function, and they are evenly distributed on the function. In other words, the (spatial) interpolation is likely to be quite accurate. Hence, it is not surprising that the prediction of the dynamic sequence with initial value $y_0 = 0.55$ has been very encouraging.

On the other hand, Figure 6 tells a different story. The dynamic sequence started at $y_0 = 1.4$ mainly forms two widely separated clusters, disallowing any scope for interpolation. There is however, some additional subtlety inolved, which is quite important. It is to be noted that in the case of $y_0 = 0.55$, the range of the training data set includes y_0, while in the case of $y_0 = 1.4$, the training data set excludes y_0 from its range. Prediction of $\eta(y_0)$, given $\mathbf{D}$ is an interpolation problem in the former case, while it is a problem of extrapolation in the latter situation. The mean square error of prediction, when interpolating, must be no bigger than when extrapolating, so that interpolating is easier in this sense (Stein (1999), page 76). Hence, in the case of extrapolation, if $y_1 = \eta(y_0)$ is predicted inaccurately (which is likely, as discussed in details by Stein (1999)), then the predictions of $y_t = \eta(y_{t-1})$; $t = 2, \ldots, T$ are also likely to be inaccurate, due to their successive dependence on the previous predictions. It can be seen from Figure 4 that indeed $\eta(y_0)$ has been predicted inaccurately. This probably explains the poor performance of the dynamic simulation method when $y_0 = 1.4$ with $\mathbf{G}$ excluding it from its range.

It is to be noted that, due to sample path continuity of the smooth Gaussian process, the dynamic sequences are expected to be closely clustered together, occupying a very small part of the function. Hence, it is natural for the user to select a training data set from a small region where the dynamic sequence is expected to lie. However, if the training data is chosen such that it excludes a part of the function where the initial value lies, then the prediction is likely to be unreliable. We do not view this as a drawback of our proposed methodology; indeed, it is obviously a simple task to select a training data set that comes from, say, a neighborhood of the initial value. We demonstrate this next.

We select $\mathbf{G} = \{-0.651, -0.102, 0.334, 0.499, 1.074, 1.450\}$; hence $\mathbf{D} = \{0.309, 0.979, 0.789, 0.560, -0.374, -0.765\}$. These 6 training data points are plotted on the true function $\cos(x + \sin(x))$, along with the dynamic sequence; see Figure 7. The training data set does seem to be informative about the true dynamic sequence to be predicted. Hence, as expected, our proposed methodology predicts the true sequence quite accurately; see Figure 8 for details. We also experimented by taking a larger training data set consisting of 10 training data points, $\mathbf{G} = \{-0.686, -0.414, -0.110, 0.153, 0.302, 0.451, 0.670, 0.929, 1.26, 1.365\}$ and hence $\mathbf{D} = \{0.250, 0.685, 0.976, 0.954, 0.825, 0.631, 0.277, -0.158, -0.597, -0.698\}$. Figure 9, which shows plots of these points on the function $\cos(x + \sin(x))$ along with the true dynamic sequence, indicates that the training data set is expected to predict the true dynamic sequence very accurately. Indeed, Figure 10, which shows the prediction of the dynamic sequence with the above 10 training data points, suggest that the expectation is quite justified.

We now generalize the methodology provided in the case of one-dimensional functions with one-dimensional input space, to multidimensional functions with multidimensional input space, and also show how to take forcing inputs into account.

6 Complete methodological details for general dynamic computer code emulation

For $t = 1, \ldots, T$, let us denote by $\mathbf{y}_t$ a p-dimensional uncertain (random) output corresponding to a p-dimensional function treated as a black box, and let $\mathbf{z}_t$ be the q-dimensional non-random forcing input at time t. Recall that, the forcing inputs consist of constant inputs as well as time-varying inputs; but since both are assumed to be completely recorded over time, they will be treated as non-random. Then we have the relationship

$$\mathbf{y}_t = \boldsymbol{\eta}(\mathbf{z}_t, \mathbf{y}_{t-1}) = \boldsymbol{\eta}(\mathbf{v}_t) \tag{C.28}$$

where $\mathbf{v}_t{}' = (\mathbf{z}_t{}', \mathbf{y}_{t-1}{}')$ and $\boldsymbol{\eta}(\cdot) = (\eta_1(\cdot), \ldots, \eta_p(\cdot))'$ is the unknown p-dimensional random function, represented as a p-variate Gaussian process where the mean function is given by

$$\mathbf{E}\left[\boldsymbol{\eta}(\cdot)\right] = \mathbf{B}'\mathbf{h}(\cdot) \tag{C.29}$$

and, for any $(p+q)$ dimensional input vectors $\mathbf{g}_1, \mathbf{g}_2$, the covariance function is given by

$$\begin{aligned} \operatorname{cov}(\boldsymbol{\eta}(\mathbf{g}_1), \boldsymbol{\eta}(\mathbf{g}_2)) &= c(\mathbf{g}_1, \mathbf{g}_2)\boldsymbol{\Sigma} \\ &= \exp\{-(\mathbf{g}_1 - \mathbf{g}_2)'\mathbf{R}(\mathbf{g}_1 - \mathbf{g}_2)\}\boldsymbol{\Sigma} \end{aligned} \tag{C.30}$$

In (C.29) and (C.30), $\mathbf{B} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_p)$, where, for $j = 1, \ldots, p$, $\boldsymbol{\beta}_j$ is an m-dimensional column vector. Hence, $\mathbf{B}$ is a matrix with m rows and p columns; $\mathbf{h}^{(m\times 1)}(\cdot) = (h_1(\cdot), \ldots, h_m(\cdot))'$, where, for $i = 1, \ldots, m$, $h_i(\cdot)$ could be any function. For example, choosing $m = 2$, we may take, $h_1(\mathbf{g}) = 1$ and $h_2(\mathbf{g}) = \mathbf{g}$ for all $\mathbf{g}$. In (C.30) $\mathbf{R}$ is a diagonal matrix consisting of $(p+q)$ smoothness parameters.

C.1 The training data set

The computer code is run on a pre-selected design set $\mathbf{G} = (\mathbf{g}_1', \ldots, \mathbf{g}_n')'$ (here we write $\mathbf{g}_k^{\{(p+q)\times 1\}'} = (\mathbf{u}_{1k}^{(p\times 1)'}, \mathbf{u}_{2k}^{(q\times 1)'})$, which is of the same form and dimensionality as $\mathbf{v}_t'$), to obtain the training data matrix $\mathbf{D} = [\eta_j(\mathbf{g}_k)]$. In other words, $\mathbf{D}$ is given by

$$\mathbf{D}^{(n\times p)} = \begin{pmatrix} \eta_1(\mathbf{g}_1) & \eta_2(\mathbf{g}_1) & \cdots & \eta_p(\mathbf{g}_1) \\ \eta_1(\mathbf{g}_2) & \eta_2(\mathbf{g}_2) & \cdots & \eta_p(\mathbf{g}_2) \\ \cdots & \cdots & \cdots & \cdots \\ \eta_1(\mathbf{g}_n) & \eta_2(\mathbf{g}_n) & \cdots & \eta_p(\mathbf{g}_n) \end{pmatrix}$$

The objective is to obtain, for each $t = 1, \ldots, T$, the posterior distribution of $\mathbf{y}_t$, given $\mathbf{D}$.

Writing $\mathbf{H}_D^{(m\times n)'} = [\mathbf{h}^{(m\times 1)}(\mathbf{g}_1), \ldots, \mathbf{h}^{(m\times 1)}(\mathbf{g}_n)]$ and $\mathbf{A}_D^{(n\times n)} = [c(\mathbf{g}_r, \mathbf{g}_\ell)]$, it follows from the Gaussian process assumption of $\boldsymbol{\eta}(\cdot)$ that $\mathbf{D}$ is matrix normal, given by

$$[\mathbf{D} \mid \mathbf{B}, \Sigma, \mathbf{R}] \sim \mathcal{N}_{n,p}(\mathbf{H}_D\mathbf{B}, \mathbf{A}_D, \boldsymbol{\Sigma}) \tag{C.31}$$

By the above notation we mean that $\mathbf{H}_D\mathbf{B}$ is the mean matrix of $\mathbf{D}$, $\mathbf{A}_D$ is the left covariance matrix, and $\boldsymbol{\Sigma}$ is the right covariance matrix. The r-th row of $\mathbf{D}$ is multivariate normal with mean being the corresponding row of the mean matrix $\mathbf{H}_D\mathbf{B}$ and covariance matrix $\boldsymbol{\Sigma}$. Rows r and ℓ of $\mathbf{D}$ has covariance matrix $c(\mathbf{g}_r, \mathbf{g}_\ell)\boldsymbol{\Sigma}$. Similarly, the ℓ-th column of $\mathbf{D}$ is distributed as multivariate normal with mean being the ℓ-th column of $\mathbf{H}_D\mathbf{B}$ and with covariance matrix $\sigma_\ell^2\mathbf{A}_D$, where σ_ℓ^2 denotes the ℓ-th diagonal element of $\boldsymbol{\Sigma}$. More generally, we define the (r,ℓ)-th element by $\sigma_{r,\ell}$, with $\sigma_{\ell,\ell} = \sigma_\ell^2$. The covariance between columns r and ℓ is given by the matrix $\sigma_{r,\ell}\mathbf{A}_D$. For further details regarding matrix normal distributions, see Dawid (1981), Carvalho and West (2007), and the references therein.

C.2 Posterior predictive distribution of the output given the data

Letting now $\mathbf{s}_D(\cdot) = [c(\cdot, \mathbf{g}_1), \ldots, c(\cdot, \mathbf{g}_n)]'$ it follows that $[\boldsymbol{\eta}(\cdot) \mid \mathbf{B}, \boldsymbol{\Sigma}, \mathbf{R}, \mathbf{D}]$ is a p-variate normal distribution with mean function

$$\boldsymbol{\mu}_1(\cdot) = \mathbf{B}'\mathbf{h}(\cdot) + (\mathbf{D} - \mathbf{H}_D\mathbf{B})'\mathbf{A}_D^{-1}\mathbf{s}_D(\cdot) \tag{C.32}$$

and the covariance function, for any $(p+q)$-dimensional $\mathbf{g}_1, \mathbf{g}_2$, is given by $c_1(\mathbf{g}_1, \mathbf{g}_2)\boldsymbol{\Sigma}$, where

$$c_1(\mathbf{g}_1, \mathbf{g}_2) = c(\mathbf{g}_1, \mathbf{g}_2) - \mathbf{s}_D(\mathbf{g}_1)'\mathbf{A}_D^{-1}\mathbf{s}_D(\mathbf{g}_2) \tag{C.33}$$

C.3 Posterior predictive distribution of $\eta(\cdot)$ after marginalizing with respect to nuisance parameters B and Σ

Using the prior $\pi(\mathbf{B},\mathbf{\Sigma}) \propto \mid \mathbf{\Sigma} \mid^{-(p+1)/2}$, it follows that

$$[\mathbf{B} \mid \mathbf{\Sigma}, \mathbf{R}, \mathbf{D}] \sim \mathcal{N}_{m,p}(\hat{\mathbf{B}}_{GLS}, (\mathbf{H}'_D\mathbf{A}_D^{-1}\mathbf{H}_D)^{-1}, \mathbf{\Sigma}) \tag{C.34}$$

In the above, $\hat{\mathbf{B}}_{GLS} = (\mathbf{H}'_D\mathbf{A}_D^{-1}\mathbf{H}_D)^{-1}(\mathbf{H}'_D\mathbf{A}_D^{-1}\mathbf{D})$; note that this can be interpreted as the generalized least square (GLS) estimate. Marginalizing the conditional $[\boldsymbol{\eta}(\cdot) \mid \mathbf{B}, \mathbf{\Sigma}, \mathbf{R}, \mathbf{D}]$ with respect to (C.34) it can be shown that $[\boldsymbol{\eta}(\cdot) \mid \mathbf{\Sigma}, \mathbf{R}, \mathbf{D}]$ is a p-variate normal distribution given by

$$[\boldsymbol{\eta}(\cdot) \mid \mathbf{\Sigma}, \mathbf{R}, \mathbf{D}] \sim \mathcal{N}_p(\boldsymbol{\mu}_2(\cdot), c_2(\cdot,\cdot)\mathbf{\Sigma}) \tag{C.35}$$

where

$$\begin{aligned} \boldsymbol{\mu}_2(\cdot) &= \hat{\mathbf{B}}'_{GLS}\mathbf{h}(\cdot) + (\mathbf{D} - \mathbf{H}_D\hat{\mathbf{B}}_{GLS})'\mathbf{A}_D^{-1}\mathbf{s}_D(\cdot) & \text{(C.36)} \\ c_2(\mathbf{g}_1,\mathbf{g}_2) &= c_1(\mathbf{g}_1,\mathbf{g}_2) + [\mathbf{h}(\mathbf{g}_1) - \mathbf{H}'_D\mathbf{A}_D^{-1}\mathbf{s}_D(\mathbf{g}_1)]'(\mathbf{H}'_D\mathbf{A}_D^{-1}\mathbf{H}_D)^{-1} \\ &\quad [\mathbf{h}(\mathbf{g}_2) - \mathbf{H}'_D\mathbf{A}_D^{-1}\mathbf{s}_D(\mathbf{g}_2)] & \text{(C.37)} \end{aligned}$$

Writing $(n-m)\hat{\mathbf{\Sigma}}_{GLS} = (\mathbf{D} - \mathbf{H}_D\hat{\mathbf{B}}_{GLS})'\mathbf{A}_D^{-1}(\mathbf{D} - \mathbf{H}_D\hat{\mathbf{B}}_{GLS})$ (equivalently, $(n-m)\hat{\mathbf{\Sigma}}_{GLS} = \mathbf{D}'\mathbf{M}\mathbf{D}$, with $\mathbf{M} = \mathbf{A}_D^{-1} - \mathbf{A}_D^{-1}\mathbf{H}_D(\mathbf{H}'_D\mathbf{A}_D\mathbf{H}_D)^{-1}\mathbf{H}'_D\mathbf{A}_D^{-1}$), it can be shown that the posterior distribution of $\mathbf{\Sigma}$, given $\mathbf{R}$ and $\mathbf{D}$ is inverse Wishart with parameters $(n-m)\hat{\mathbf{\Sigma}}_{GLS}$ and $n-m$ (degrees of freedom). Integrating (C.35) with respect to the posterior of $\mathbf{\Sigma}$, we obtain the following p-variate Student's t distribution

$$[\boldsymbol{\eta}(\cdot) \mid \mathbf{R}, \mathbf{D}] \sim \mathcal{T}_p\left(\boldsymbol{\mu}_2(\cdot), c_2(\cdot,\cdot)\hat{\mathbf{\Sigma}}_{GLS}; n-m\right) \tag{C.38}$$

The smoothness parameters, the diagonal elements of $\mathbf{R}$, are estimated by maximum likelihood methods, and subsequent calculations are carried out conditional on this estimated value, as the form of (C.38) suggests.

C.4 Simulation of the set of latent variables $\mathbf{D}^*$ in the general set up

From (C.38) it is clear that for posterior simulation of the first value of the random sequence $\{\mathbf{y}_1 = \boldsymbol{\eta}(\mathbf{v}_1), \mathbf{y}_2 = \boldsymbol{\eta}(\mathbf{v}_2), \ldots, \mathbf{y}_T = \boldsymbol{\eta}(\mathbf{v}_T)\}$, we must simulate $\mathbf{y}_1 = \boldsymbol{\eta}(\mathbf{v}_1)$ from $\mathcal{T}_p\left(\boldsymbol{\mu}_2(\mathbf{v}_1), c_2(\mathbf{v}_1,\mathbf{v}_1)\hat{\mathbf{\Sigma}}_{GLS}; n-m\right)$. We next need to simulate $\mathbf{D}^*$ from $[\mathbf{D}^* \mid \mathbf{R}, \mathbf{D}, \boldsymbol{\eta}(\mathbf{v}_1)]$, given $\mathbf{G}^* = (\mathbf{g}_1^*, \ldots, \mathbf{g}_N^*)'$, where, for $j = 1, \ldots, N$, input $\mathbf{g}_j^*$ is of dimensionality $p+q$. Note that the above conditional is a matrix-variate Student's t distribution; it will be convenient to simulate $\mathbf{D}^*$ by successively simulating from the conditional distribution $[\boldsymbol{\eta}(\mathbf{g}_j^*) \mid \mathbf{R}, \mathbf{D}, \boldsymbol{\eta}(\mathbf{v}_1), \boldsymbol{\eta}(\mathbf{g}_1^*), \ldots, \boldsymbol{\eta}(\mathbf{g}_{j-1}^*)]$, for $j = 1, \ldots, N$. Before providing the explicit form of this p-variate Student's t distribution, let us first define

$$\mathbf{D}_j^{((n+j)\times p)} = \begin{pmatrix} \eta_1(\mathbf{g}_1) & \eta_2(\mathbf{g}_1) & \cdots & \eta_p(\mathbf{g}_1) \\ \vdots & \vdots & \vdots & \vdots \\ \eta_1(\mathbf{g}_n) & \eta_2(\mathbf{g}_n) & \cdots & \eta_p(\mathbf{g}_n) \\ \eta_1(\mathbf{v}_1) & \eta_2(\mathbf{v}_1) & \cdots & \eta_p(\mathbf{v}_1) \\ \eta_1(\mathbf{g}_1^*) & \eta_2(\mathbf{g}_1^*) & \cdots & \eta_p(\mathbf{g}_1^*) \\ \vdots & \vdots & \vdots & \vdots \\ \eta_1(\mathbf{g}_{j-1}^*) & \eta_2(\mathbf{g}_{j-1}^*) & \cdots & \eta_p(\mathbf{g}_{j-1}^*) \end{pmatrix} \tag{C.39}$$

$$\mathbf{H}'_{j,D} = [\mathbf{h}(\mathbf{g}_1),\ldots,\mathbf{h}(\mathbf{g}_n),\mathbf{h}(\mathbf{v}_1),\mathbf{h}(\mathbf{g}_1^*),\ldots,\mathbf{h}(\mathbf{g}_{j-1}^*)] \tag{C.40}$$

$$\mathbf{s}_{(j,D)}(\cdot) = [c(\cdot,\mathbf{g}_1),\ldots,c(\cdot,\mathbf{g}_n),c(\cdot,\mathbf{v}_1),c(\cdot,\mathbf{g}_1^*),\ldots,c(\cdot,\mathbf{g}_{j-1}^*)]' \tag{C.41}$$

$\mathbf{A}_{(j,D)} =$

$$\begin{pmatrix} c(\mathbf{g}_1,\mathbf{g}_1) & \ldots & c(\mathbf{g}_1,\mathbf{g}_n) & c(\mathbf{g}_1,\mathbf{v}_1) & c(\mathbf{g}_1,\mathbf{g}_1^*) & \ldots & c(\mathbf{g}_1,\mathbf{g}_{j-1}^*) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c(\mathbf{g}_n,\mathbf{g}_1) & \ldots & c(\mathbf{g}_n,\mathbf{g}_n) & c(\mathbf{g}_n,\mathbf{v}_1) & c(\mathbf{g}_n,\mathbf{g}_1^*) & \ldots & c(\mathbf{g}_n,\mathbf{g}_{j-1}^*) \\ c(\mathbf{v}_1,\mathbf{g}_1) & \ldots & c(\mathbf{v}_1,\mathbf{g}_n) & c(\mathbf{v}_1,\mathbf{v}_1) & c(\mathbf{v}_1,\mathbf{g}_1^*) & \ldots & c(\mathbf{v}_1,\mathbf{g}_{j-1}^*) \\ c(\mathbf{g}_1^*,\mathbf{g}_1) & \ldots & c(\mathbf{g}_1^*,\mathbf{g}_n) & c(\mathbf{g}_1^*,\mathbf{v}_1) & c(\mathbf{g}_1^*,\mathbf{g}_1^*) & \ldots & c(\mathbf{g}_1^*,\mathbf{g}_{j-1}^*) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c(\mathbf{g}_{j-1}^*,\mathbf{g}_1) & \ldots & c(\mathbf{g}_{j-1}^*,\mathbf{g}_n) & c(\mathbf{g}_{j-1}^*,\mathbf{v}_1) & c(\mathbf{g}_{j-1}^*,\mathbf{g}_1^*) & \ldots & c(\mathbf{g}_{j-1}^*,\mathbf{g}_{j-1}^*) \end{pmatrix} \tag{C.42}$$

Then it follows that

$$[\boldsymbol{\eta}(\mathbf{g}_j^*) \mid \mathbf{R},\mathbf{D},\boldsymbol{\eta}(\mathbf{v}_1),\boldsymbol{\eta}(\mathbf{g}_1^*),\ldots,\boldsymbol{\eta}(\mathbf{g}_{j-1}^*)] \sim \mathcal{T}_p\left(\boldsymbol{\mu}_{2,j}(\cdot), c_{2,j}(\cdot,\cdot)\hat{\boldsymbol{\Sigma}}_{GLS,j}; n+j-m\right) \tag{C.43}$$

where

$$\boldsymbol{\mu}_{2,j}(\cdot) = \hat{\mathbf{B}}'_{GLS,j}\mathbf{h}(\cdot) + (\mathbf{D}_j - \mathbf{H}_{j,D}\hat{\mathbf{B}}_{GLS,j})'\mathbf{A}_{j,D}^{-1}\mathbf{s}_{j,D}(\cdot) \tag{C.44}$$

$$\begin{aligned} c_{2,j}(\mathbf{g}_1,\mathbf{g}_2) &= c_{1,j}(\mathbf{g}_1,\mathbf{g}_2) + [\mathbf{h}(\mathbf{g}_1) - \mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{s}_{j,D}(\mathbf{g}_1)]'(\mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{H})^{-1} \\ &\quad [\mathbf{h}(\mathbf{g}_2) - \mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{s}_{j,D}(\mathbf{g}_2)] \end{aligned} \tag{C.45}$$

In the above, $c_{1,j}(\mathbf{g}_1,\mathbf{g}_2) = c(\mathbf{g}_1,\mathbf{g}_2) - \mathbf{s}_{(j,D)}(\mathbf{g}_1)'\mathbf{A}_{(j,D)}^{-1}\mathbf{s}_{(j,D)}(\mathbf{g}_2)$, and

$$\hat{\mathbf{B}}_{GLS,j} = (\mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{H}_{j,D})^{-1}(\mathbf{H}'_{j,D}\mathbf{A}_{j,D}^{-1}\mathbf{D}_j) \tag{C.46}$$

$$(n+j-m)\hat{\boldsymbol{\Sigma}}_{GLS,j} = (\mathbf{D}_j - \mathbf{H}_{j,D}\hat{\mathbf{B}}_{GLS,j})'\mathbf{A}_{j,D}^{-1}(\mathbf{D}_j - \mathbf{H}_{j,D}\hat{\mathbf{B}}_{GLS,j}) \tag{C.47}$$

It is clear from the above expressions that for simulation of each successive $\boldsymbol{\eta}(\mathbf{g}_j)$, it is only necessary to successively augment the quantities (C.39), (C.40), (C.41), (C.42), (C.44), (C.45), (C.46), (C.47) with the immediately preceding simulation $\boldsymbol{\eta}(\mathbf{g}_{j-1})$. Hence, simulation of $\mathbf{D}^*$ is computationally efficient.

C.5 Simulation of the posterior dynamic sequence given D and D*

Once we have obtained a realization of $\mathbf{D}^*$, we can then proceed to simulate $\mathbf{y}_t$ from $[\boldsymbol{\eta}(\mathbf{v}_t) \mid \mathbf{R}, \mathbf{D}, \mathbf{D}^*]$. We let $\mathbf{D}_{n+N} = \begin{pmatrix} \mathbf{D} \\ \mathbf{D}^* \end{pmatrix}$ be the augmented matrix,

$$\begin{aligned}
\mathbf{H}'_{(D^*,D)} &= [\mathbf{h}(\mathbf{g}_1), \ldots, \mathbf{h}(\mathbf{g}_n), \mathbf{h}(\mathbf{g}_1^*), \ldots, \mathbf{h}(\mathbf{g}_N^*)], \\
\mathbf{s}_{(D^*,D)}(\cdot) &= [c(\cdot, \mathbf{g}_1), \ldots, c(\cdot, \mathbf{g}_n), c(\cdot, \mathbf{g}_1^*), \ldots, c(\cdot, \mathbf{g}_N^*)]', \text{ and} \\
\mathbf{A}_{(D^*,D)} &= \begin{pmatrix} \mathbf{A}^{(11)}_{(D^*,D)} & \mathbf{A}^{(12)}_{(D^*,D)} \\ \mathbf{A}^{(21)}_{(D^*,D)} & \mathbf{A}^{(22)}_{(D^*,D)} \end{pmatrix}
\end{aligned}$$

In the above, $\mathbf{A}^{(11)}_{(D^*,D)} = \mathbf{A}_D$, $\mathbf{A}^{(22)}_{(D^*,D)} = \mathbf{A}_{D^*}$, and the (i,j)-th element of $\mathbf{A}^{(12)}_{(D^*,D)} = {\mathbf{A}^{(21)}_{(D^*,D)}}'$ is $c(\mathbf{g}_i, \mathbf{g}_j^*)$, for $i = 1, \ldots, n$ and $j = 1, \ldots, N$. It follows that, for $t = 2, \ldots, T$,

$$[\boldsymbol{\eta}(\mathbf{v}_t) \mid \mathbf{R}, \mathbf{D}, \mathbf{D}^*] \sim \mathcal{T}_p\left(\boldsymbol{\mu}_{2,n+N}(\cdot), c_{2,n+N}(\cdot,\cdot)\hat{\boldsymbol{\Sigma}}_{GLS,n+N}; n+N-m\right) \tag{C.48}$$

In the above,

$$\boldsymbol{\mu}_{2,n+N}(\cdot) = \hat{\mathbf{B}}'_{GLS,n+N}\mathbf{h}(\cdot) + (\mathbf{D}_{n+N} - \mathbf{H}_{(D^*,D)}\hat{\mathbf{B}}_{GLS,n+N})'\mathbf{A}^{-1}_{(D^*,D)}\mathbf{s}_{(D^*,D)}(\cdot) \tag{C.49}$$

$$\begin{aligned}
c_{2,n+N}(\mathbf{g}_1, \mathbf{g}_2) &= c_{1,n+N}(\mathbf{g}_1, \mathbf{g}_2) + [\mathbf{h}(\mathbf{g}_1) - \mathbf{H}'_{(D^*,D)}\mathbf{A}^{-1}_{(D^*,D)}\mathbf{s}_{(D^*,D)}(\mathbf{g}_1)]' \\
&\quad (\mathbf{H}'_{(D^*,D)}\mathbf{A}^{-1}_{(D^*,D)}\mathbf{H}_{(D^*,D)})^{-1}[\mathbf{h}(\mathbf{g}_2) - \mathbf{H}'_{(D^*,D)}\mathbf{A}^{-1}_{(D^*,D)}\mathbf{s}_{(D^*,D)}(\mathbf{g}_2)]
\end{aligned} \tag{C.50}$$

In (C.50), $c_{1,n+N}(\mathbf{g}_1, \mathbf{g}_2) = c(\mathbf{g}_1, \mathbf{g}_2) - \mathbf{s}_{(D^*,D)}(\mathbf{g}_1)'\mathbf{A}^{-1}_{(D^*,D)}\mathbf{s}_{(D^*,D)}(\mathbf{g}_2)$. In (C.49) and (C.48), $\hat{\mathbf{B}}_{GLS,n+N}$ and $\hat{\boldsymbol{\Sigma}}_{GLS,n+N}$ are given by the following:

$\hat{\mathbf{B}}_{GLS,n+N} =$

$$(\mathbf{H}'_{(D^*,D)}\mathbf{A}^{-1}_{(D^*,D)}\mathbf{H}_{(D^*,D)})^{-1}(\mathbf{H}'_{(D^*,D)}\mathbf{A}^{-1}_{(D^*,D)}\mathbf{D}_{n+N}) \tag{C.51}$$

$(n+N-m)\hat{\boldsymbol{\Sigma}}_{GLS,n+N} =$

$$(\mathbf{D}_{n+N} - \mathbf{H}_{(D^*,D)}\hat{\mathbf{B}}_{GLS,n+N})'\mathbf{A}^{-1}_{(D^*,D)}(\mathbf{D}_{n+N} - \mathbf{H}_{(D^*,D)}\hat{\mathbf{B}}_{GLS,n+N}) \tag{C.52}$$

Using the above expressions, one can easily simulate an approximate realization from the posterior distribution of the dynamic sequence. Obviously, the simulation method must be repeated, starting with a new realization of $[\boldsymbol{\eta}(\mathbf{v}_1) \mid \mathbf{R}, \mathbf{D}]$, to obtain the approximate joint posterior distribution of the dynamic sequence (and certainly the marginal posterior distributions as well).

At the first glance, the simulation method, particularly in the general set up, may seem computationally burdensome because of the complicated looks of the expressions. We assure, however, that this is not actually the case. Note that, since the smoothness parameters in $\mathbf{R}$ are held fixed, the matrix $\mathbf{A}_{(D^*,D)}$ remains fixed as well. This implies that simulation from $[\boldsymbol{\eta}(\mathbf{v}_t) \mid \mathbf{R}, \mathbf{D}, \mathbf{D}^*]$ requires inversion of the matrix $\mathbf{A}_{(D^*,D)}$ *only once*, before even beginning the implementation of the simulation algorithm. Thus there is no need to recompute the inverse of the matrix for each simulation, easing computational burden remarkably. Clearly, the computation remains efficient even if the matrix $\mathbf{A}_{(D^*,D)}$ is of large size (which will be the case if N is large). Moreover, since the grid $\mathbf{G}^* = (\mathbf{d}_1^*, \ldots, \mathbf{d}_N^*)'$ is discrete, and will be chosen by the implementor, numerical difficulties associated with the inversion of the matrix can be very easily avoided, unless the implementor chooses to select an arbitrarily fine grid. In this context, it is useful to remark that selecting an arbitrarily fine grid does not necessarily result in more accuracy as compared to a relatively less fine, but judiciously chosen grid; this has already been observed in the case of the one-dimensional example given in Section 5. The same phenomenon will also be observed in the case of the 4-dimensional example provided in Section 7. Indeed, there are formal approaches to selecting an optimal set of design (input) points and there is no reason to consider an arbitrarily fine grid; see Santner et al. (2003) and the references therein for details regarding choice of optimal design. Recently, Tokdar (2007) provides another criterion for selecting design points.

Thus, it seems possible to efficiently compute the distributions of dynamic sequences of long lengths with reasonable accuracy, without any numerical problem whatsoever! We believe that any other methodology (if proposed in the future) for simulating dynamic sequences will fail to have this very important property. Thus, our proposed methodology seems to be quite general and useful in practice.

We remark that in case the grid $\mathbf{G}$, which corresponds to the training data set $\mathbf{D}$, is sufficiently fine, then it implies that, given $\mathbf{D}$, the unknown function $\boldsymbol{\eta}(\cdot)$ has already been learned with sufficient accuracy. In such a case, we can obtain the dynamic sequence by simply simulating from $[\boldsymbol{\eta}(\mathbf{v}_t) \mid \mathbf{R}, \mathbf{D}]$; hence simulation of $\mathbf{D}^*$ is unnecessary in that situation. However, in practice we can not expect to learn the computer model so accurately; the code is always evaluated at a set of pre-selected design points only.

7 Application of the proposed methodology to a multi-dimensional computer model with forcing inputs

We now demonstrate the performance of our proposed methodology with a real four-variate dynamic computer code. However, this dynamic code is not a model for any real physical process, but may be looked upon as a dynamic emulator for a toy climate model. We present this example essentially as "proof of concept" in support of our methodology, albeit, as clarified above, it is unrelated to any genuine physical process. Given an initial input, this code can produce 99 dynamic outputs. Hence, including the initial input, a time series consisting of 100 time points is produced by the black box. It is useful to mention that this code is not expensive to run. This is important for evaluating our methodology, since the exact answer can be obtained easily to make comparison with the predictions obtained using our proposal. If the code were expensive to run, then the exact answers would be difficult to obtain, precluding evaluation of our methodology.

It is also useful to mention that, of four components of the code, two components are almost linear functions of the corresponding inputs, while the other two components substantially deviate from linearity. We demonstrate that, dynamic outputs of all four components are predicted satisfactorily by our methodology, irrespective of whether the original input-output relationship is linear or not.

Likening our toy example to a climate change model, we denote the inputs of the above-mentioned computer code at time t by the following

$Temp(t) =$ Average global temperature at time t.
$Melt(t) =$ Ice melting rate at time t.
$Barren(t) =$ Area of barren land at time t.
$Carb(t) =$ World's total carbon emission at time t.

The forcing variable for this computer code is

$Deforest(t) =$ Area of forest lost at time t.

For any t, the four dimensional input $\mathbf{y}_{t-1} = (Temp(t-1), Melt(t-1), Barren(t-1), Carb(t-1))'$ together with the forcing input $z_t = Deforest(t)$ produce the four-dimensional output $\mathbf{y}_t = \boldsymbol{\eta}(\mathbf{v}_t) = (Temp(t), Melt(t), Barren(t), Carb(t))'$. For evaluation of our proposed method we use the code in two ways. In one way we treat the code as it is, and use the above multivariate (4-variate) Student's t distribution for simulation from $[\boldsymbol{\eta}(\mathbf{v}_t) \mid \mathbf{R}, \mathbf{D}^*, \mathbf{D}]$. In another way, we treat this code as four different codes, and assume that in each code, except one variable, all others are forcing inputs. That is, for the first code we assume that only $Temp$ is the dynamic output, and $\{Melt, Barren, Carb, Deforest\}$ are all forcing inputs. In other words, we assume that $\mathbf{y}_{t-1} = Temp(t-1)$ and $\mathbf{z}_t$=$\{Melt(t),\ Barren(t),\ Carb(t),\ Deforest(t)\}$ produce $y_t = \eta(\mathbf{v}_t) = Temp(t)$. Thus, in this set-up, $[\eta(\mathbf{v}_t) \mid \mathbf{D}^*, \mathbf{D}]$ is one-dimensional. We do the same in turn for $Melt(t)$, $Barren(t)$ and $Carb(t)$.

The training data set $\mathbf{D}$ is obtained by first selecting a set of design points $\mathbf{G}$ from the input space; the design is generated by Latin hypercube sampling (for details, see Santner et al. (2003)). We chose 50 such points. Once the design points were chosen, we then evaluated the computer code on the set of design points for a single time step and stored the outputs along with the corresponding inputs (design points).

We also needed to choose the grid $\mathbf{G}^*$ for the random set $\mathbf{D}^*$. The grid points can be selected efficiently if one has *a priori* knowledge of the region where all possible simulations of dynamic sequences are expected to lie. From our experiences, and from insight gained from the one-dimensional example already studied in Section 5, we can expect the dynamic outputs to lie within a neighborhood of the specified initial input. To form an idea regarding the neighborhood, we used a pilot run of our simulation algorithm; this entails selecting a 'trial grid' $\mathbf{G}^*$, which is coarse, consisting of a few points, and then running our dynamic simulation method. The resulting 'trial' posterior distribution of the dynamic sequence provided a good guess of the neighborhood where the original dynamic sequence is expected to belong. We then selected a finer grid $\mathbf{G}^*$, consisting of 50 points from the neighborhood to improve simulation accuracy. The grid points in the neighborhood are again selected by Latin hypercube sampling. However, we also repeated the experiment with size of the grid $\mathbf{G}^*$ being 60, 70, 80 and 100; the results remained almost exactly the same as in the initial case where the size of $\mathbf{G}^*$ was 50. So, in this case, we conclude that 50 design points from an appropriate neighborhood of the initial input are adequate. This again demonstrates that making the grid arbitrarily fine does not necessarily result in significant improvement of accuracy of the posterior simulation, and is, in fact, wasteful.

As in the case of the univariate example, in this multivariate situation as well we obtain the posterior distributions of the dynamic sequence by simulating 500 random sequences using our proposed methodology.

C.1 Results

Figure 11 shows the predictions of the dynamic output sequence when the original code is treated as four individual codes. The solid line denotes the true output, which has been obtained by actually running the dynamic code for comparison purpose. The dashed line stands for the posterior mean, which we use as an estimate of the true dynamic sequence. The dot-dashed lines are the approximate 95% credible intervals of the posterior distributions. It is notable that all four outputs are predicted very satisfactorily. In the case of $Temp$ and $Melt$ the posterior variances are so close to zero that the posterior mean and the approximate 95% credible intervals seem to coincide. The reason for this is that the original individual codes for $Temp$ and $Melt$ are very close to linear. Since the simulated function must pass through all the points in the training data set $(\mathbf{G}, \mathbf{D})$, and since those points exhibit a linear relationship, in our methodology, the linear mean function of the Gaussian process dominates the calculations. Compared to $Temp$ and $Melt$, prediction of $Barren$ and $Carb$ are more interesting as the codes in these cases substantially deviate from linearity. Hence, as expected, the figure shows that the approximate 95% credible intervals are much wider than in the case of $Temp$ and $Melt$.

Figure 12 shows the predictions when the code is treated as it is, that is, when the output is considered a 4-dimensional vector, as it should be. Clearly, again the predictions of the four outputs turned out to be very satisfactory. The credible regions are larger than the corresponding univariate case, but this is to be expected, since in the univariate case except one variable, there is no uncertainty about the remaining three variables.

As mentioned before, this example is essentially "proof of concept". However, it does succeed in revealing the potential of the proposed methodology. We anticipate that our proposal will play a very vital role in the case of very highly complex dynamic codes with great scientific importance. Models associated with future climate change seem to be an important area of application of our dynamic emulation methodology.

8 Conclusion

In this paper we have proposed a novel and general methodology for emulation of complex dynamic computer models. To our knowledge the statistical literature does not contain any material on dynamic computer code emulation. We have provided the theory for both one-dimensional and multidimensional cases, explained that our methodology is computationally efficient, and demonstrated with two examples (one-dimensional and 4-dimensional computer models) that our methodology is expected to work in practice. An important computational aspect that deserves mention is the parallelisability of our methodology discussed in Section 4.

However, issues regarding choice of optimal design points is important, and although there is literature regarding this issue, there still does not seem to exist a generally accepted method of choosing an optimal design.

Appendix

Demonstration that marginalization of $\mathbf{D}^*$ increases computational burden and causes numerical instability

We have indicated that the posterior dynamic sequence $[y_1 = \eta(y_0), y_2 = \eta(y_1), \ldots, y_t = \eta(y_{T-1}) \mid \mathbf{D}]$ can be simulated easily by introducing the set of latent variables $\mathbf{D}^*$, and the underlying procedure is to first simulate y_1 from $[\eta(y_0) \mid \mathbf{D}]$, the distribution of which is known. Then $\mathbf{D}^*$ must be simulated from the known conditional distribution $[\mathbf{D}^* \mid \mathbf{D}, \eta(y_0)]$. Finally, the remaining part of the dynamic sequence can be simulated by simulating y_{t+1} from the known conditional distributions $[\eta(y_t) \mid \mathbf{D}, y_t]$, for $t = 2, \ldots, T-1$. At the first sight it seems that it is desirable to integrate out $\mathbf{D}^*$ analytically to reduce the number of variables to be simulated (indeed, apart from simulation of $y_1, \ldots, y_T$, we also need to simulate $\mathbf{D}^*$). However, we show that integrating out $\mathbf{D}^*$ will require inversion of matrices of increasing size, which is linear in time t, at each step of the resulting simulation procedure. Moreover, the elements of the matrices are random, and hence, numerical instabilities can not be prevented by existing approximation methods; in other words, one completely loses control over the dimensionality and elements of the matrix. Hence computational burden and numerical instability typically arises in this simulation method.

To avoid introducing more and messy notation and to avoid more complicated expressions, here we consider simulation from the prior dynamic sequence $[y_1 = \eta(y_0), y_2 = \eta(y_1), \ldots, y_t = \eta(y_{T-1})]$. Conditioning this on the training data set $\mathbf{D}$ presents no new issue, and can be safely avoided for the sake of clarity.

Note that,

$$[\eta(y_t) \mid y_t = \eta(y_{t-1}), y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0)]$$

$$\begin{aligned} &= \int [\eta(y_t) \mid \mathbf{D}^*, y_t = \eta(y_{t-1}), y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0)] \\ &\quad [\mathbf{D}^* \mid y_t = \eta(y_{t-1}), y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0)] d\mathbf{D}^* && \text{(C.53)} \\ &= \int [\eta(y_t) \mid \mathbf{D}^*, y_t][\mathbf{D}^* \mid y_t = \eta(y_{t-1}), y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0)] d\mathbf{D}^* && \text{(C.54)} \end{aligned}$$

Equation (C.54) follows from (C.53) by the assumed Markov property. To compute (C.54) we need the distribution of $[\mathbf{D}^* \mid y_t = \eta(y_{t-1}), y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0)]$. This is proportional to

$$\begin{aligned} & [\mathbf{D}^*][y_t = \eta(y_{t-1}), y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0) \mid \mathbf{D}^*] && \text{(C.55)} \\ = \; & [\mathbf{D}^*][y_1 = \eta(y_0) \mid \mathbf{D}^*][y_2 = \eta(y_1) \mid \mathbf{D}^*, y_1 = \eta(y_0)] \\ \ldots \; & [y_t = \eta(y_{t-1}) \mid \mathbf{D}^*, y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0)] && \text{(C.56)} \\ = \; & [\mathbf{D}^*][y_1 = \eta(y_0) \mid \mathbf{D}^*][y_2 = \eta(y_1) \mid \mathbf{D}^*, y_1] \ldots [y_t = \eta(y_{t-1}) \mid \mathbf{D}^*, y_{t-1}] && \text{(C.57)} \end{aligned}$$

Step (C.57) again follows from step (C.56) due to the Markov property already assumed. Now write

$$\mathbf{A}_t = \sum_{i=0}^{t-1} \left[\frac{\mathbf{A}_{D^*}^{-1}}{t} + \frac{\mathbf{A}_{D^*}^{-1} \mathbf{s}_{D^*}(y_i) \mathbf{s}_{D^*}(y_i)' \mathbf{A}_{D^*}^{-1}}{1 - \mathbf{s}_{D^*}(y_i)' \mathbf{A}^{-1} \mathbf{s}_{D^*}(y_i)} \right] \tag{C.58}$$

and

$$\mathbf{B}_t = \mathbf{A}_{D^*}^{-1} \sum_{i=0}^{t-1} \frac{\{\eta(y_i) - \mathbf{h}(y_i)'\boldsymbol{\beta}\} \mathbf{s}_{D^*}(y_i)}{1 - \mathbf{s}_{D^*}(y_i)' \mathbf{A}_{D^*}^{-1} \mathbf{s}_{D^*}(y_i)} \tag{C.59}$$

It follows from (C.57) that

$$[\mathbf{D}^* \mid y_t = \eta(y_{t-1}), y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0)] \sim \mathcal{N}_N(\mathbf{H}_{D^*}\boldsymbol{\beta} + \mathbf{A}_t^{-1}\mathbf{B}_t, \sigma^2 \mathbf{A}_t^{-1}) \tag{C.60}$$

which is an N-dimensional normal distribution with mean vector $\mathbf{H}_{D^*}\boldsymbol{\beta} + \mathbf{A}_t^{-1}\mathbf{B}_t$ and covariance matrix $\sigma^2 \mathbf{A}_t^{-1}$. Integrating (C.60) with respect to the distribution of $[\eta(y_t) \mid \mathbf{D}^*, y_t]$, which is a univariate normal distribution with mean

$$\mathbf{h}(y_t)'\boldsymbol{\beta} + \mathbf{s}_{D^*}(y_t)' \mathbf{A}_{D^*}^{-1} (\mathbf{D}^* - \mathbf{H}_{D^*}\boldsymbol{\beta})$$

and variance

$$\sigma^2 \{1 - \mathbf{s}_{D^*}(y_t)' \mathbf{A}_{D^*}^{-1} \mathbf{s}_{D^*}(y_t)\},$$

we obtain

$$[\eta(y_t) \mid y_t = \eta(y_{t-1}), y_{t-1} = \eta(y_{t-2}), \ldots, y_1 = \eta(y_0)] \sim \mathcal{N}(\mathbf{h}(y_t)'\boldsymbol{\beta} + \mathbf{B}_t^*/\mathbf{A}_t^*, \sigma^2/\mathbf{A}_t^*) \tag{C.61}$$

where

$$\begin{aligned}\mathbf{A}_t^* &= \frac{1}{1-\mathbf{s}_{D^*}(y_t)'\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)}\\ &- \left\{\frac{\mathbf{s}_{D^*}(y_t)'\mathbf{A}_{D^*}^{-1}}{1-\mathbf{s}_{D^*}(y_t)'\mathbf{A}^{-1}\mathbf{s}_{D^*}(y_t)}\right\}\left\{\mathbf{A}_t+\frac{\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)\mathbf{s}_{D^*}(y_t)'\mathbf{A}_{D^*}^{-1}}{1-\mathbf{s}_{D^*}(y_t)'\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)}\right\}^{-1}\\ &\quad\left\{\frac{\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)}{1-\mathbf{s}_{D^*}(y_t)'\mathbf{A}^{-1}\mathbf{s}_{D^*}(y_t)}\right\}\end{aligned} \tag{C.62}$$

and

$$\mathbf{B}_t^* = \mathbf{B}_t'\left\{\mathbf{A}_t+\frac{\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)\mathbf{s}_{D^*}(y_t)'\mathbf{A}_{D^*}^{-1}}{1-\mathbf{s}_{D^*}(y_t)'\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)}\right\}^{-1}\left\{\frac{\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)}{1-\mathbf{s}_{D^*}(y_t)\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)}\right\} \tag{C.63}$$

We illustrate the distribution of $[\eta(y_1) \mid y_1 = \eta(y_0)]$ using the above expressions.

Prior distribution of $\eta(y_1)$ given $y_1 = \eta(y_0)$, marginalised over $\mathbf{D}^*$

For simplicity, we consider $\mathbf{G}^* = \{g^*\}$ and $\mathbf{D}^* = \{\eta(g^*)\}$. Then $\mathbf{A}$ is reduced to the scalar 1, $\mathbf{s}_{D^*}(y_1) = c(y_1, g^*)$, $\mathbf{s}_{D^*}(y_0) = c(g^*, y_0)$, $\mathbf{A}_t = 1/\{1 - c^2(g^*, y_0)\}$, and $\mathbf{B}_t = c(g^*, y_0)\{\eta(y_0) - \mathbf{h}(y_0)'\boldsymbol{\beta}\}/\{1 - c^2(g^*, y_0)\}$. It follows that $\mathbf{A}_t^* = 1/\{1 - c^2(y_1, g^*)c^2(g^*, y_0)\}$ and $\mathbf{B}_t^* = c(y_1, g^*)c(g^*, y_0)\{\eta(y_0) - \mathbf{h}(y_0)'\boldsymbol{\beta}\}/\{1 - c^2(y_1, g^*)c^2(g^*, y_0)\}$. Hence, with $\mathbf{G}^* = \{g^*\}$ and $\mathbf{D}^* = \{\eta(g^*)\}$, we obtain

$$[\eta(y_1) \mid y_1 = \eta(y_0)] \sim$$

$$\mathcal{N}\left[\mathbf{h}(y_1)'\boldsymbol{\beta} + c(y_1, g^*)c(g^*, y_0)\{\eta(y_0) - \mathbf{h}(y_0)'\boldsymbol{\beta}\}, \sigma^2\{1 - c^2(y_1, g^*)c^2(g^*, y_0)\}\right] \tag{C.64}$$

If we plug in $g^* = y_0$ or $g^* = y_1$ in (C.64), then we obtain

$$[\eta(y_1) \mid y_1 = \eta(y_0)] \sim \mathcal{N}\left[\mathbf{h}(y_1)'\boldsymbol{\beta} + c(y_1, y_0)\{\eta(y_0) - \mathbf{h}(y_0)'\boldsymbol{\beta}\}, \sigma^2\{1 - c^2(y_1, y_0)\}\right] \tag{C.65}$$

However, there arises a subtle question as to whether it is legitimate to plug in $g^* = y_0$ or $g^* = y_1$. Note that the form of the conditional distribution (C.64) has been derived using the fact that g^* is a fixed constant, so that $(\eta(g^*), \eta(y_0))$ and $(\eta(y_1), \eta(g^*))$ given $y_1 = \eta(y_0)$ are distributed as bivariate normal. But the joint distribution of $(\eta(y_1), \eta(y_0))$, given $y_1 = \eta(y_0)$ is certainly not bivariate normal, since the second random variable reduces to a constant. A direct consequence of this degeneracy seems to be the fact that, if instead of substituting $g^* = y_0$ or $g^* = y_1$ in the final form (C.64), $g^* = y_0$ or $g^* = y_1$ is plugged directly into the original form of $\mathbf{A}_t^*$ or $\mathbf{B}_t^*$ given, as above, by $\mathbf{A}_t^* = 1/\{1 - c^2(y_1, g^*)c^2(g^*, y_0)\}$ and $\mathbf{B}_t^* = c(y_1, g^*)c(g^*, y_0)\{\eta(y_0) - \mathbf{h}(y_0)'\boldsymbol{\beta}\}/\{1 - c^2(y_1, g^*)c^2(g^*, y_0)\}$, then division by zero entails. Thus the theoretical validity of (C.65) may be questioned; it will be argued that the simulations from conditionals of the form (C.65) must encounter severe, and possibly insurmountable, instabilities. We conjecture, from our past experiences, that the latter might stem from the incorrect derivation of the conditionals, as demonstrated above. However, if we *choose to assume* legitimacy of the substitution in the final form (C.64), then we can proceed further, as follows.

In order to derive (C.64), we assumed that $\mathbf{G}^*$ and hence $\mathbf{D}^*$ are singletons. Actually, the same form (C.64) can be arrived at from a much more general point of view. Assume that $\mathbf{G}^* = \{g_1^*, g_2^*, \ldots, g_N^*\}$, and correspondingly $\mathbf{D}^* = \{\eta(g_1^*), \eta(g_2^*), \ldots, \eta(g_N^*)\}$. Then

$$[\eta(y_1) \mid y_1 = \eta(y_0)] = \int [\eta(y_1) \mid \mathbf{D}^*, y_1][\mathbf{D}^* \mid y_1 = \eta(y_0)] d\mathbf{D}^* \tag{C.66}$$

It is easy to see that, if, for some $k \in \{1, \ldots, N\}$, $g_k^* = y_1$ or $g_k^* = y_0$, then $\mathbf{D}^*$ contains $\eta(y_1)$ or $\eta(y_0)$. Hence, either the first factor of the integrand, $[\eta(y_1) \mid \mathbf{D}^*] = \delta_{\eta(y_1)}$ or the second factor of the integrand $[\mathbf{D}^* \mid y_1 = \eta(y_0)] = \delta_{\eta(y_0)}$. In either case, (C.66) yields the conditional distribution (C.65). The same result can also be obviously obtained by directly plugging in $g_k^* = y_1$ or $g_k^* = y_0$ in the form implied by (C.61) in this case, but derivation using (C.66) is probably more intuitively appealing and simpler, requiring no linear algebraic manoevre.

Note that, if $\mathbf{G}^*$ does not contain y_1 or y_0, but contains g^*'s which are close to y_1 or y_0, then the corresponding conditional distribution $[\eta(y_1) \mid y_1 = \eta(y_0)]$, will be close to (C.66). The examples illustrated in this paper also support this.

Assuming legitimacy of the form of the conditional given by (C.65), one can obtain more generally that

$$[\eta(y_t) \mid \eta(y_{t-1}) \ldots \eta(y_0)] \sim \mathcal{N}(\mathbf{h}(y_t)'\boldsymbol{\beta} + \mathbf{s}_t(y_t)'\mathbf{A}_t^{-1}(\mathbf{D}_t - \mathbf{H}_t\boldsymbol{\beta}), \sigma^2\{1 - \mathbf{s}_t(y_t)'\mathbf{A}_t^{-1}\mathbf{s}_t(y_t)\}) \tag{C.67}$$

In the above,

$\mathbf{D}_t = (\eta(y_{t-1}), \ldots, \eta(y_0))$, $\mathbf{s}_t(y_t) = (c(y_t, y_0), \ldots, c(y_t, y_{t-1}))'$, $\mathbf{H}_t' = [\mathbf{h}(y_0), \ldots, \mathbf{h}(y_{t-1})]$, and

$$\mathbf{A}_t = \begin{pmatrix} c(y_0, y_0) & \ldots & c(y_0, y_{t-1}) \\ \ldots & \ldots & \ldots \\ c(y_{t-1}, y_0) & \ldots & c(y_{t-1}, y_{t-1}) \end{pmatrix}$$

It is very important to note the difference between successive simulation of the distribution (C.61), marginalised over $\mathbf{D}^*$, for $t = 1, \ldots, T$, or the conditionals (again, marginalised) given by (C.67), and successive simulation from the distribution $[\eta(y_t) \mid \mathbf{D}^*, y_t]$ conditional on simulated $\mathbf{D}^*$. In the latter case, the matrix $\mathbf{A}_{D^*}$ consists of non-random, completely known elements, and needs to be inverted *only once*, even before beginning to implement the simulation algorithm. Thus the algorithm is computationally very fast and presents no numerical difficulties; see Section C.5 for details in the general situation. On the other hand, simulation from (C.61) involves computation of (C.62) and (C.63) for simulation of each point of the dynamic sequence, and at every time t, the computations require inversion of the matrix $\mathbf{A}_t + \frac{\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)\mathbf{s}_{D^*}(y_t)'\mathbf{A}_{D^*}^{-1}}{1 - \mathbf{s}_{D^*}(y_t)'\mathbf{A}_{D^*}^{-1}\mathbf{s}_{D^*}(y_t)}$. The situation is the same in the case of simulations from the conditionals of the form (C.67). This is certainly computationally very burdensome. Moreover, and more importantly, the elements of the matrix are random, where no control can be exercised, and if y_t's are close, which is expected due to sample path continuity of $\eta(\cdot)$, the matrix is very likely to become numerically singular, preventing inversion. In fact, we have performed a lot of experiments, particularly to simulate successively from the conditionals (C.67) (after conditioning on the training data set $\mathbf{D}$), and in all cases, the methodology broke down after just a few time points, due to numerical problems associated with matrix inversions in each step. Indeed, when applied to each of the examples presented in Sections 5 and 7, the simulation procedure using marginalized conditionals simply failed to progress further than just a few steps, for each of the examples. On the other hand, retaining the latent variables $\mathbf{D}^*$ proved to be an excellent idea in all examples (including those presented in this paper) we have come across. Hence we recommend using the latent variables $\mathbf{D}^*$ for efficient and reliable simulation.

References

Carvalho, C. M. and West, M. (2007). "Dynamic Matrix-Variate Graphical Models." *Bayesian Analysis*, 2(1): 69–98. 796

Conti, S., Anderson, C. W., Kennedy, M. C., and O'Hagan, A. (2004). "A Bayesian Analysis of Complex Dynamic Computer Models." Available online at http://library.lanl.gov/cgi-bin/getdoc?event=SAMO2004&document=samo04-39.pdf. 808

Conti, S., Gosling, J. P., Oakley, J. E., and O'Hagan, A. (2007). "Gaussian process emulation of dynamic computer codes." Technical Report 571/07, Department of Probability and Statistics, University of Sheffield. Available online at http://j-p-gosling.staff.shef.ac.uk/Pub/DynEm.pdf. 808

Conti, S. and O'Hagan, A. (2007). "Bayesian emulation of complex multi-output and dynamic computer models." Technical Report 569/07, Department of Probability and Statistics, University of Sheffield. Available online at http://tonyohagan.co.uk/academic/ps/multioutput.ps. 808

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. New York: Wiley. 789

Dawid, A. P. (1981). "Some matrix-variate distribution theory: Notational considerations and a Bayesian application." *Biometrika*, 68: 265–274. 796

Haylock, R. G. and O'Hagan, A. (1996). "On inference for outputs of computationally expensive algorithms with uncertainty on the inputs." In Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M. (eds.), *Bayesian Statistics 5*, 629–637. Oxford: New York. 784

Kennedy, M. C. and O'Hagan, A. (2001). "Bayesian calibration of computer models (with discussion)." *Journal of the Royal Statistical Society. Series B*, 63(3): 425–464. 784, 785, 793

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

— (2004). "Probabilistic sensitivity analysis of complex models: a Bayesian approach." *Journal of the Royal Statistical Society. Series B*, 66(3): 751–769. 784

Saltelli, A., Tarantola, S., and Campolongo, F. (2000). "Sensitivity analysis as an ingredient of modeling." *Statistical Science*, 15: 377–395. 784

Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The design and analysis of computer experiments*. Springer Series in Statistics. New York, Inc.: Springer-Verlag. 786, 791, 800, 801

SenGupta, A. and Ugwuowo, F. L. (2006). "Asymmetric circular-linear multivariate regression models with applications to environmental data." *Environmental and Ecological Statistics*, 13: 299–309. 789

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. New York, Inc: Springer-Verlag. 785, 789, 794

Tokdar, S. T. (2007). "Towards a Faster Implementation of Density Estimation with Logistic Gaussian Process Priors." *Journal of Computational and Graphical Statistics*, 16(2): 1–23. 800

Acknowledgments

This work has benefitted from discussions with Prof. Tony O'Hagan, Dr. Jeremy Oakley, Dr. Stefano Conti, Dr. John Paul Gosling and the illustration of some of their unpublished works (Conti et al. (2004, 2007); Conti and O'Hagan (2007)), although the approach proposed here differs from that of Conti et al. (2007) and the other aforementioned unpublished works.. The author thanks Dr. John Paul Gosling for writing and kindly providing the 4-dimensional dynamic computer code of Section 7 on which performance of the proposed methodology is demonstrated. Dr. Jeremy Oakley provided maximum likelihood estimates of the smoothness parameters in the case of individual codes of Section 7. The author also expresses his gratitude to Prof. Jayanta Ghosh for helpful discussions; to two anonymous referees and the Editor-in-Chief of Bayesian Analysis, Prof. Bradley Carlin, for providing comments and suggestions which helped improve the quality and presentation of this paper.

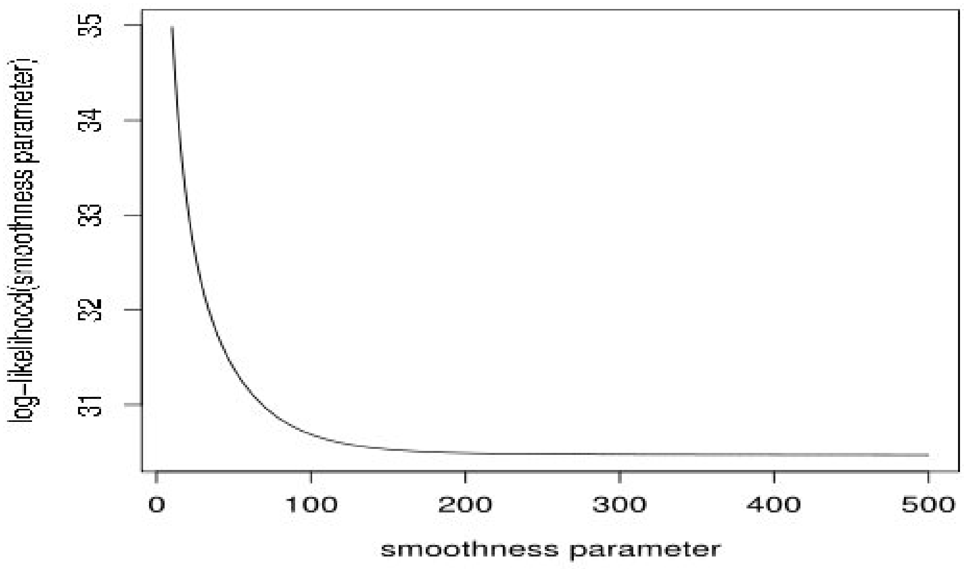

Figure 1: Plot of the log-likelihood of the smoothness parameter of the Gaussian process model corresponding to training data obtained from the true, deterministic function $f(x) = \cos(x + \sin(x))$.

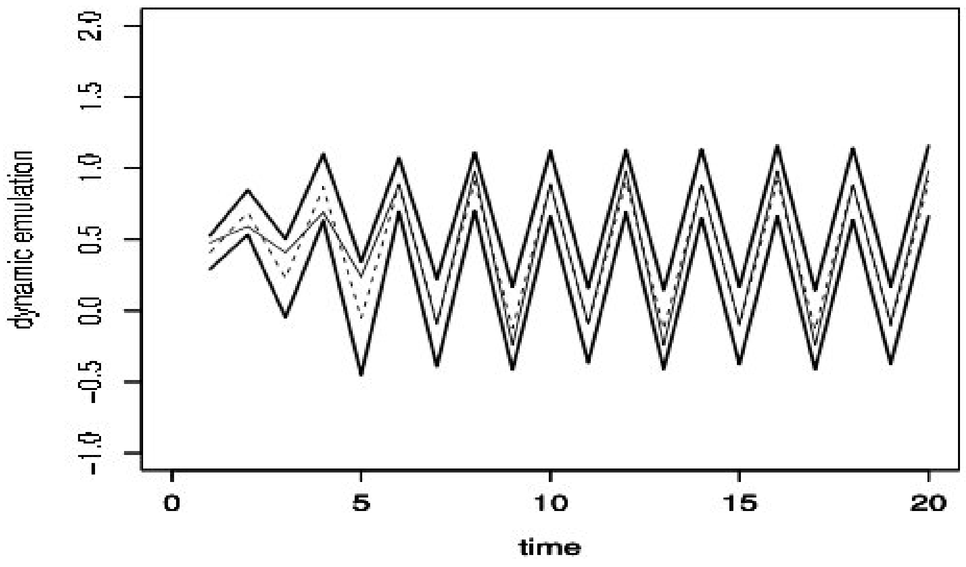

Figure 2: Prediction of dynamic sequence corresponding to the function $f(x) = \cos(x + \sin(x))$. The initial value is 0.55. The true, deterministic dynamic sequence is denoted by the thin, solid line. The mean posterior dynamic sequence obtained by our Gaussian process based methodology is denoted by the dotted line, with the corresponding 95% credible intervals denoted by the solid, thick lines. In this case, the true, deterministic dynamic sequence has been predicted accurately.

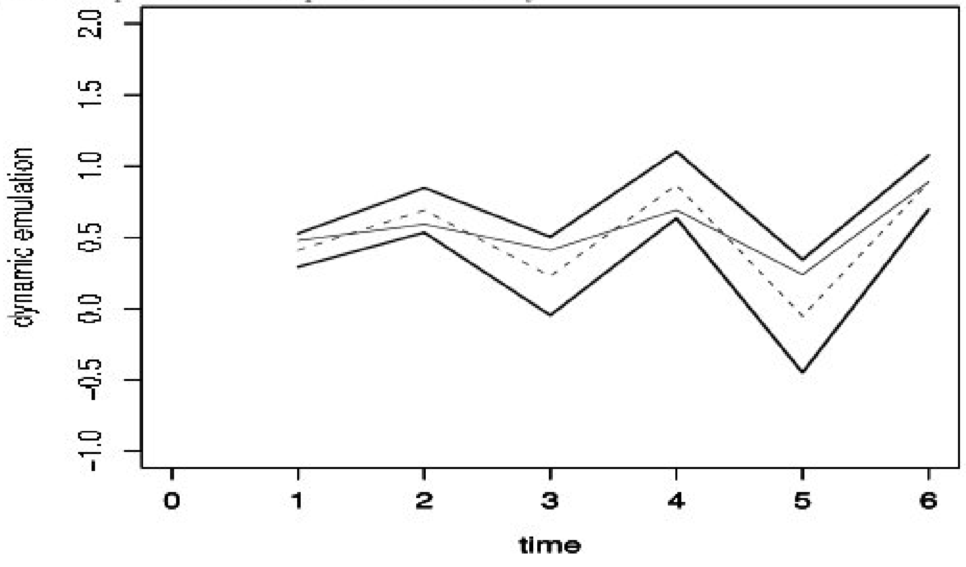

Figure 3: Prediction of the first 6 values of the dynamic sequence corresponding to the function $f(x) = \cos(x + \sin(x))$ after integrating out $\mathbf{D}^*$. The initial value is 0.55. 100 dynamic sequences are simulated. The true, deterministic dynamic sequence is denoted by the thin, solid line. The mean posterior dynamic sequence obtained by our Gaussian process based methodology is denoted by the dotted line, with the corresponding 95% credible intervals denoted by the solid, thick lines.

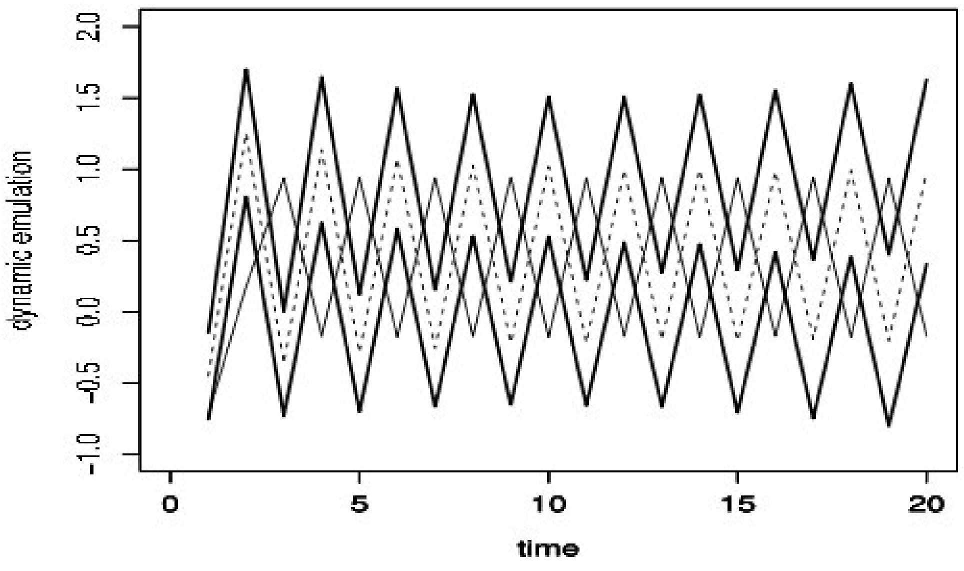

Figure 4: Prediction of dynamic sequence corresponding to the function $f(x) = \cos(x + \sin(x))$. The initial value is 1.4. The true, deterministic dynamic sequence is denoted by the thin, solid line. The mean posterior dynamic sequence obtained by our Gaussian process based methodology is denoted by the dotted line, with the corresponding 95% credible intervals denoted by the solid, thick lines. In this case, prediction of the true, deterministic dynamic sequence is inaccurate.

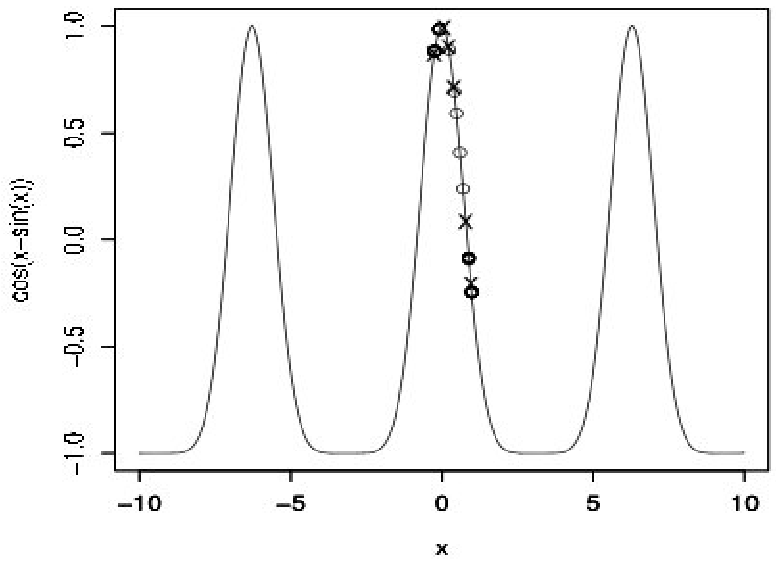

Figure 5: Plot of the function $f(x) = \cos(x + \sin(x))$.The circles on the graph are the plots of the true, deterministic dynamic sequence $y_2, \ldots, y_T$ against the corresponding inputs $y_1, \ldots, y_{T-1}$; the initial value for this sequence is $y_0 = 0.55$. The cross signs stand for the observed training data set $\mathbf{D}$. The training data seem to be quite informative about the true dynamic sequence, and y_0 is included within the range of $\mathbf{G}$.

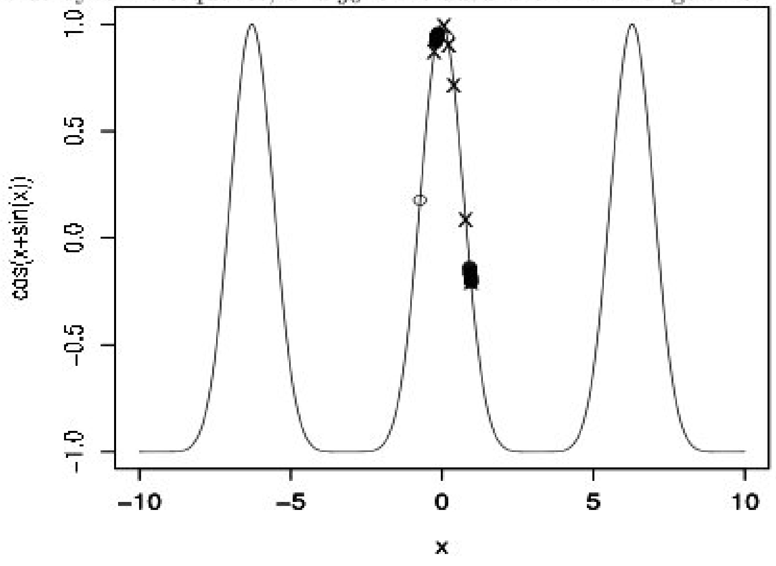

Figure 6: Plot of the function $f(x) = \cos(x + \sin(x))$.The circles on the graph are the plots of the true, deterministic dynamic sequence $y_2, \ldots, y_T$ against the corresponding inputs $y_1, \ldots, y_{T-1}$; the initial value for this sequence is $y_0 = 1.4$. The cross signs stand for the observed training data set $\mathbf{D}$. The training data do not seem to be informative about the true dynamic sequence and y_0 is not included within the range of $\mathbf{G}$.

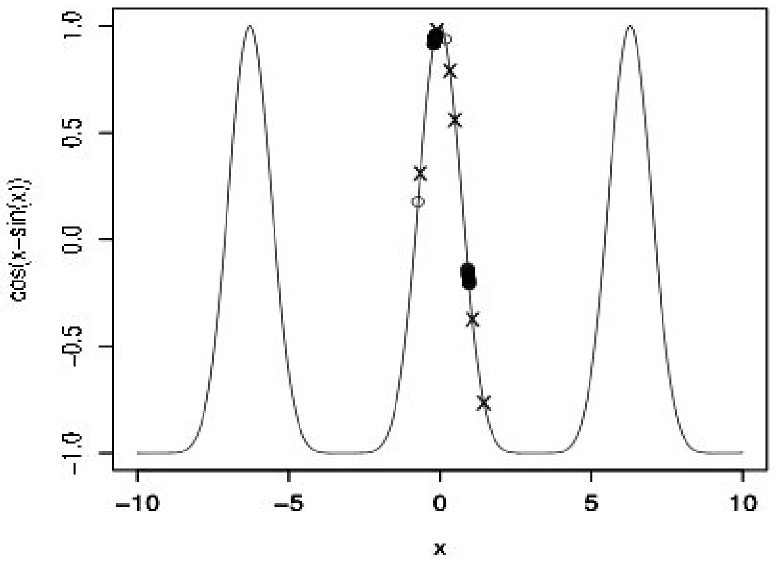

Figure 7: Plot of the function $f(x) = \cos(x + \cos(x))$.The circles on the graph are the plots of the true, deterministic dynamic sequence $y_2, \ldots, y_T$ against the corresponding inputs $y_1, \ldots, y_{T-1}$; the initial value for this sequence is $y_0 = 1.4$. The plus signs stand for the observed training data set $\mathbf{D}$ of size 6. Points of the grid $\mathbf{G}$ are selected from a neighborhood of $y_0 = 1.4$.

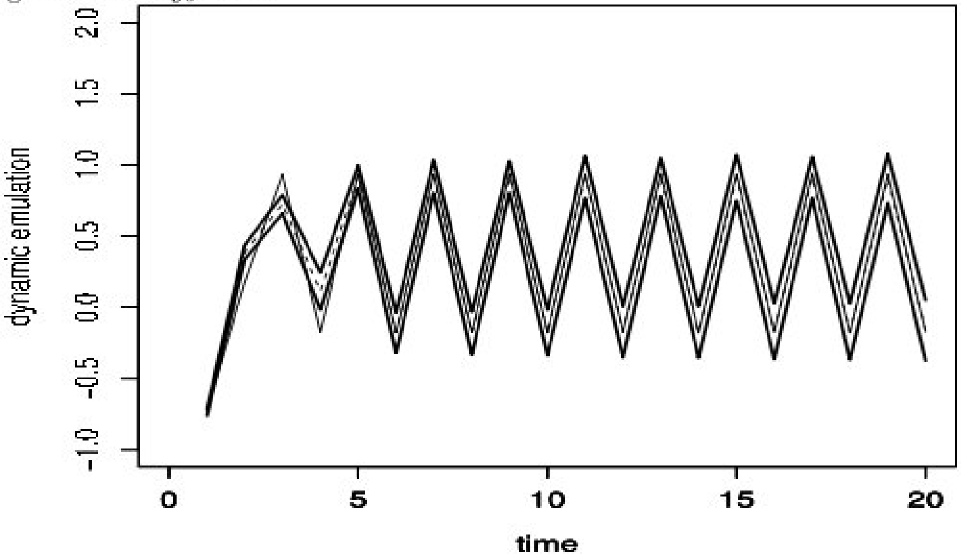

Figure 8: Prediction of dynamic sequence corresponding to the function $f(x) = \cos(x + \sin(x))$. The initial value is 1.4. 6 training data points used around 1.4. The true, deterministic dynamic sequence is denoted by the thin, solid line. The mean posterior dynamic sequence obtained by our Gaussian process based methodology is denoted by the dotted line, with the corresponding 95% credible intervals denoted by the solid, thick lines. In this case, prediction of the true, deterministic dynamic sequence is accurate.

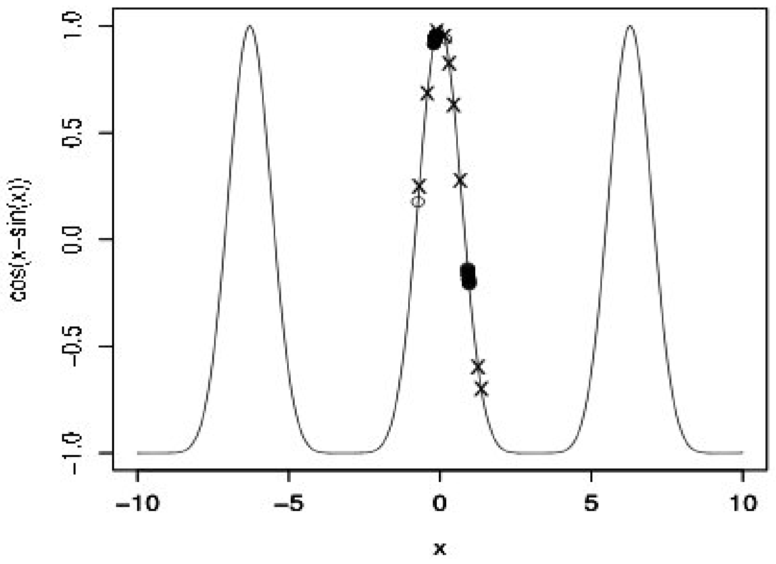

Figure 9: Plot of the function $f(x) = \cos(x + \cos(x))$.The circles on the graph are the plots of the true, deterministic dynamic sequence $y_2, \ldots, y_T$ against the corresponding inputs $y_1, \ldots, y_{T-1}$; the initial value for this sequence is $y_0 = 1.4$. The plus signs stand for the observed training data set $\mathbf{D}$ of size 10. Points of the grid $\mathbf{G}$ are selected from a neighborhood of $y_0 = 1.4$.

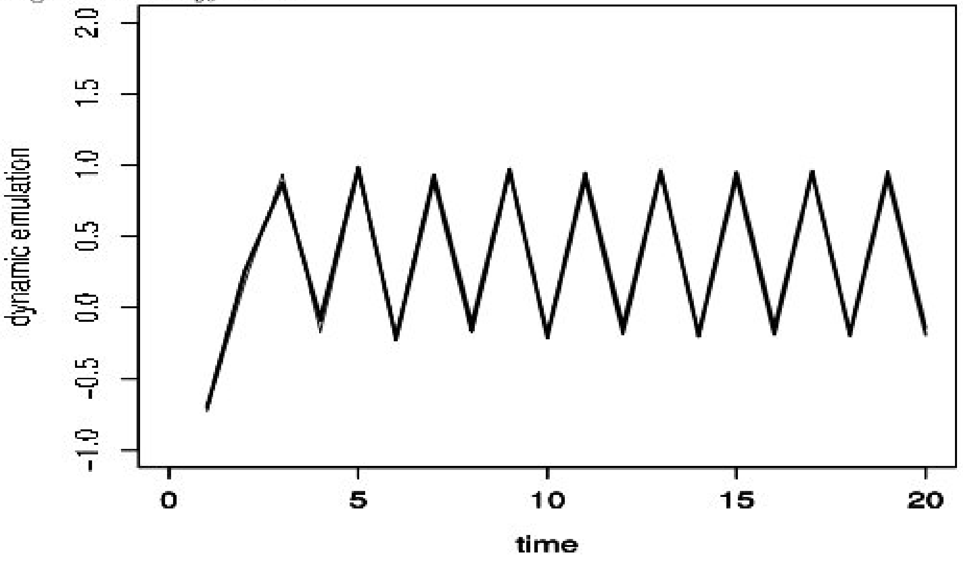

Figure 10: Prediction of dynamic sequence corresponding to the function $f(x) = \cos(x+\cos(x))$. The initial value is 1.4. 10 training data points used around 1.4. The true, deterministic dynamic sequence is denoted by the thin, solid line. The mean posterior dynamic sequence obtained by our Gaussian process based methodology is denoted by the dotted line, with the corresponding 95% credible intervals denoted by the solid, thick lines. In this case, prediction of the true, deterministic dynamic sequence is very accurate.

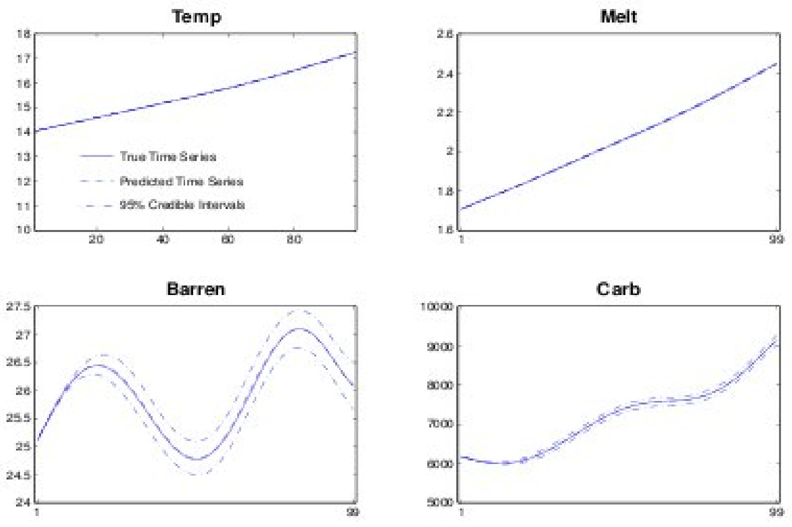

Figure 11: Prediction of dynamic outputs using proposed theory assuming four individual dynamic codes.

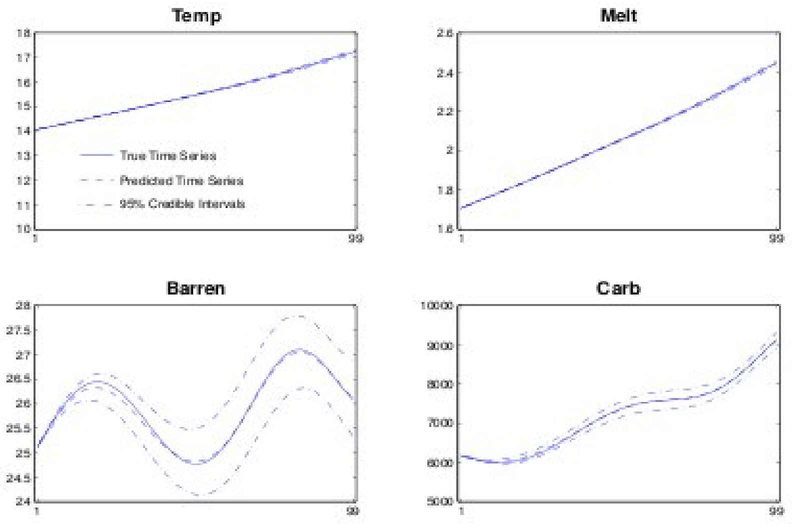

Figure 12: Prediction of dynamic outputs using proposed theory assuming that all four outputs are dynamic.