Biometrika (2000), **87,** 1, *pp.* 1–13

Printed in Great Britain

Predicting the output from a complex computer code when fast approximations are available

BY M. C. KENNEDY AND A. O'HAGAN
Department of Probability and Statistics, University of Sheffield, Sheffield, S3 7RH, U.K.
m.c.kennedy@sheffield.ac.uk a.ohagan@sheffield.ac.uk

SUMMARY

We consider prediction and uncertainty analysis for complex computer codes which can be run at different levels of sophistication. In particular, we wish to improve efficiency by combining expensive runs of the most complex versions of the code with relatively cheap runs from one or more simpler approximations. A Bayesian approach is described in which prior beliefs about the codes are represented in terms of Gaussian processes. An example is presented using two versions of an oil reservoir simulator.

Some key words: Bayesian uncertainty analysis; Computer experiment; Gaussian process; Multi-level code.

1. COMPUTER SIMULATIONS

Complex mathematical models, implemented in large computer codes, have been used to study real systems in many areas of scientific research (Sacks et al., 1989), usually because physical experimentation is too costly and sometimes impossible, as in the case of large environmental systems. A 'computer experiment' involves running the code with various input values for the purpose of learning something about the real system.

Often a simulator can be run at different levels of complexity, with versions ranging from the most sophisticated high level code to the most basic. For example, in § 4 we consider two codes which simulate oil pressure at a well of a hydrocarbon reservoir. Both codes use finite element analysis, in which the rocks comprising the reservoir are represented by small interacting grid blocks. The flow of oil within the reservoir can be simulated by considering the interaction between the blocks. The two codes differ in the resolution of the grid, so that we have a very accurate, slow version using many small blocks and a crude approximation using large blocks which runs much faster.

Alternatively, a mathematical model could be expanded to include more of the scientific laws underlying the physical processes. Simple, fast versions of the code may well include the most important features, and are useful for preliminary investigations. In real-time applications the number of runs from a high level simulator may be limited by expense. Then there is a need to trade-off the complexity of the expensive code with the availability of the simpler approximations.

The purpose of the current paper is to explore ways in which runs from several levels of a code can be used to make inference about the output from the most complex code. We may also have uncertainty about values for the input parameters which apply in any given application. Uncertainty analysis of computer codes describes how this uncertainty on the inputs affects our uncertainty about the output.

In principle a complex code will approximate reality better than a simple code, but in extreme cases a single run of a complex code may take a number of days, even on a powerful computer. For example, the oil reservoir simulator used by Craig et al. (1996) using a large finite element grid can take between one and three days to produce a single output. Another potential problem with complex codes is the need to specify large numbers of parameters, which can be difficult to identify from physical data and often impossible to measure directly.

Young, Parkinson & Lees (1996) give more detailed criticisms of the way in which complex simulators are used, and suggest simpler models that are derived using physical data. The codes we consider are deterministic; that is, running the code with the same inputs always produces identical outputs, and we have no observation error.

The structure of the paper is as follows. In § 2 we describe a Bayesian analysis of multi-level codes using an autoregressive model. Bayesian uncertainty analysis is introduced in § 3, and we illustrate the use of the autoregressive model and the associated uncertainty analysis in § 4, using data from an oil reservoir simulator. In § 5 we consider an alternative model for a series of computer codes of increasing complexity, and we conclude with some discussion in § 6.

2. BAYESIAN ANALYSIS OF COMPUTER CODES

2·1. *General assumptions*

For our analysis we make the following assumptions.

(i) Different levels of the same code are correlated in some way. Extra complexity is usually achieved by expanding simple models, so that each level of code should share some basic features.

(ii) The codes have a degree of smoothness, in the sense that the output values for similar inputs are reasonably close. If the codes are extremely rough, then individual runs can only provide information about the output in a small surrounding neighbourhood, and the advantage of the Bayesian model is minimal.

(iii) Prior beliefs about each level of the code can be modelled using a Gaussian process.

(iv) Each code output is scalar. Computer codes often produce multivariate time series outputs, and in principle the methods we present would generalise quite easily if multivariate normality could be assumed for these outputs. We will consider only the univariate case in the current paper.

Suppose we have s levels of code $z_1(.), \ldots, z_s(.)$. We model the output y from the tth level code as $y = z_t(x)$, where $z_t(.)$ is a random function indexed by a p-dimensional input vector x. For any given input x_i we can run the code to observe data $y_i = z_t(x_i)$. The number of available runs will be limited by time and computer resources, so for large regions of the input space $z_t(x)$ will be unknown. The degree to which $z_t(x)$ is unknown depends on the position of x in relation to the tried input points, and also on the smoothness properties of $z_t(x)$.

Prediction of $z_t(.)$, or of functionals of $z_t(.)$, is therefore a problem of statistical inference. Statistical analysis of functions has been studied before from both non-Bayesian and Bayesian perspectives; Diaconis (1988) and O'Hagan (1992) provide useful reviews. In common with many of these techniques, we use a Gaussian process to model the code output. The use of such models for the analysis of computer code outputs has a substantial literature. An important review of early work in the field is given by Sacks et al. (1989).

Some more recent references from the same research group are Morris, Mitchell & Ylvisaker (1993), Welch et al. (1992) and Aslett et al. (1998).

In Bayesian statistics, the use of Gaussian processes to model unknown functions dates back to Kimeldorf & Wahba (1970), Blight & Ott (1975) and O'Hagan (1978). Neal (1999) is a recent general review, and O'Hagan, Kennedy & Oakley (1999) reviews a body of work related to the present paper.

2·2. *An autoregressive model*

For each $t = 1, \ldots, s$, we let D_t be the design set consisting of the n_t points $x_1^{(t)}, \ldots, x_{n_t}^{(t)}$. The output data are written $z^{\mathrm{T}} = (z_1^{\mathrm{T}}, \ldots, z_s^{\mathrm{T}})$, where $z_t^{\mathrm{T}} = (z_t(x_1^{(t)}), \ldots, z_t(x_{n_t}^{(t)}))$ is the vector of outputs for the level t code. These data are observed without error, since the codes are deterministic. The object of inference is $[z_s(.)\,|\,z]$, the top level code conditional on all the code data.

Consider the following assumption about two levels of code $z_t(.)$ and $z_{t-1}(.)$, where $z_t(.)$ is the higher level code:

$$\mathrm{cov}\{z_t(x), z_{t-1}(x')\,|\,z_{t-1}(x)\} = 0 \tag{1}$$

for all $x' \neq x$. This is a kind of Markov property: given the nearest point $z_{t-1}(x)$, we can learn no more about $z_t(x)$ from any other run $z_{t-1}(x')$ for $x' \neq x$.

We now introduce an autoregressive model which has the properties described above. Indeed, it is shown in the University of Nottingham technical report, 'A Markov property for covariance structures' by A. O'Hagan, that the assumption (1), together with stationarity of $z_t(x)$ over the x space for each t, implies precisely this model. We assume that

$$z_t(x) = \rho_{t-1} z_{t-1}(x) + \delta_t(x) \quad (t = 2, \ldots, s), \tag{2}$$

where ρ_{t-1} is a kind of regression parameter and $\delta_t(.)$ is independent of $z_{t-1}(.), \ldots, z_1(.)$. A similar Markov property is proposed in Currin et al. (1991), but they assume stationarity both in x and in t. Their assumption is stronger and leads to a Kronecker product form for the joint covariance structure, known in the geostatistics literature as separability. The assumption is too strong for multi-level codes, where we wish to allow different levels to have different correlation structures. Further details are given in the technical report of O'Hagan.

Conditional on hyperparameters β_t and σ_t^2, we model $\delta_t(.)$ as a stationary Gaussian process with mean $h(.)^{\mathrm{T}}\beta_t$, where $h(.)$ is a vector of q regression functions, and covariance function $c_t(x, x') = \mathrm{cov}\{\delta_t(x), \delta_t(x')\}$. Conditional on β_1 and σ_1^2, the simplest code $z_1(.)$ is also assumed to have a stationary Gaussian process independent of the $\delta_t(.)$. For each of the covariance functions we assume the exponential form

$$c_t(x, x') = \sigma_t^2 \exp\{-b_t(x - x')^{\mathrm{T}}(x - x')\}, \tag{3}$$

where b_t is a roughness parameter. The use of (3) represents a belief that the code has a high degree of smoothness, in particular that it is infinitely differentiable, and is isotropic. Various other forms have been suggested in the computer experiments literature; see for example Sacks et al. (1989) or Currin et al. (1991). Isotropy could be relaxed by using instead of (3) a product of correlations, in which each term in the product has a different roughness b_{ti}, and our results would generalise easily in this case. However, we have generally found the form (3) to provide good results, and the additional b_{ti} parameters are difficult to estimate using the available data. Finally we assume independent non-informative priors $p(\beta_t, \sigma_t^2, b_t) \propto b_t^{-1}\sigma_t^{-2}$ for each β_t, σ_t^2 and b_t. It would be relatively

straightforward to incorporate proper prior distributions for these hyperparameters, but in practice we do not expect useful prior information about them to be available.

For each level t, we select design points D_t such that $D_t \subseteq D_{t-1}$. The conditional independence of each $\delta_t(.)$ then implies that level t data depend on $z_1, \ldots, z_{t-1}$ through z_{t-1} alone. This property is useful for estimation of the model parameters for each code level. More details are given below. The notation $A_t(D_k, D_l)$ is used for the matrix of correlations between points in D_k and D_l, with i, j element

$$A_t(x_i^{(k)}, x_j^{(l)}) = \exp\{-b_t(x_i^{(k)} - x_j^{(l)})^{\mathrm{T}}(x_i^{(k)} - x_j^{(l)})\}$$

for all $x_i^{(k)} \in D_k$ and $x_j^{(l)} \in D_l$. We will use $A_t(D_k)$ as a shorthand for $A_t(D_k, D_k)$.

2·3. *Posterior distribution for a code with two levels*

To simplify the exposition we first describe the analysis of a code which has only two levels, $s = 2$, corresponding to a fast simulator $z_1(.)$ and a slow simulator $z_2(.)$. We let $\beta = (\beta_1, \beta_2)$ and $\phi = (\sigma_1^2, \sigma_2^2, b_1, b_2, \rho_1)$. The Gaussian process models together with (2) imply that $[z_2(x), z_2(x'), z \mid \beta, \phi]$ is multivariate normal. Standard results for normal distributions can then be used to show that $[z_2(.) \mid z, \phi]$, after integrating out β analytically, is a Gaussian process with mean function

$$m'(x) = h'(x)^{\mathrm{T}}\hat{\beta} + t(x)^{\mathrm{T}}V^{-1}(z - H\hat{\beta}), \tag{4}$$

where

$$h'(x)^{\mathrm{T}} = (\rho_1 h(x)^{\mathrm{T}}, h(x)^{\mathrm{T}}), \quad H = \begin{pmatrix} h(x_1^{(1)})^{\mathrm{T}} & 0 \\ \vdots & \vdots \\ h(x_{n_1}^{(1)})^{\mathrm{T}} & 0 \\ \rho_1 h(x_1^{(2)})^{\mathrm{T}} & h(x_1^{(2)})^{\mathrm{T}} \\ \vdots & \vdots \\ \rho_1 h(x_{n_2}^{(2)})^{\mathrm{T}} & h(x_{n_2}^{(2)})^{\mathrm{T}} \end{pmatrix},$$

$$\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2)^{\mathrm{T}} = (H^{\mathrm{T}}V^{-1}H)^{-1}H^{\mathrm{T}}V^{-1}z, \tag{5}$$

$$t(x)^{\mathrm{T}} = \mathrm{cov}\{z_2(x), z^{\mathrm{T}}\} = (\rho_1\sigma_1^2 A_1(\{x\}, D_1), \rho_1^2\sigma_1^2 A_1(\{x\}, D_2) + \sigma_2^2 A_2(\{x\}, D_2)), \tag{6}$$

and the data covariance matrix V may be expressed in block form as

$$V = \begin{pmatrix} \sigma_1^2 A_1(D_1) & \rho_1\sigma_1^2 A_1(D_1, D_2) \\ \rho_1\sigma_1^2 A_1(D_2, D_1) & \rho_1^2\sigma_1^2 A_1(D_2) + \sigma_2^2 A_2(D_2) \end{pmatrix}.$$

The posterior mean of β is $\hat{\beta}$. The covariance function for $[z_2(x) \mid z, \phi]$ can be written

$$\begin{aligned} c'(x, x') = {} & c(x, x') - t(x)^{\mathrm{T}}V^{-1}t(x) \\ & + (h'(x) - t(x)^{\mathrm{T}}V^{-1}H)^{\mathrm{T}}(H^{\mathrm{T}}V^{-1}H)^{-1}(h'(x) - t(x)^{\mathrm{T}}V^{-1}H), \end{aligned} \tag{7}$$

where $c(x, x') = c_2(x, x') + \rho_1^2 c_1(x, x')$. The posterior mean function (4) is a cheap approximation for the expensive top level code, and can be used to predict the code output at untried inputs. Provided we make enough runs of the slow code, (4) should be more accurate than runs of the fast code. The posterior covariance function (7) with $x = x'$ can be used to measure the uncertainty on the prediction of $z(x)$.

2·4. *Estimating the model hyperparameters*

The covariance structure of the data is simplified as a result of our choice of design points and the Markov property. The parameters β are estimated by their posterior mean (5). The distribution of the data conditional on ϕ can be written as the product

$$p(z\,|\,\phi) = p(z_2\,|\,z_1, \rho_1, b_2, \sigma_2^2)p(z_1\,|\,b_1, \sigma_1^2). \tag{8}$$

From the independence condition we can estimate the parameters $(\rho_1, b_2, \sigma_2^2)$ independently of (b_1, σ_1^2), by maximising each term in the product (8). Each of these has a normal form, and it can easily be shown that we should minimise

$$\log|A_1(D_1)| + n_1 \log \sigma_1^2 + (z_1 - \hat{\beta}_1 1_{n_1})^{\mathrm{T}}\{\sigma_1^2 A_1(D_1)\}^{-1}(z_1 - \hat{\beta}_1 1_{n_1})$$

to choose b_1, σ_1^2. For the second level of code, β_2, σ_2^2, b_2 and ρ_1 are estimated using z_2 and z_1, since the Markov property implies that the parameters depend only on these data. We define $d_2 = z_2 - \rho_1 z_1(D_2)$, where $z_1(D_2)$ denotes the vector of outputs from $z_1(.)$ at points in D_2. Then b_2, σ_2^2 and ρ_1 are chosen to minimise

$$\log|A_2(D_2)| + n_2 \log \sigma_2^2 + (d_2 - \hat{\beta}_2 1_{n_2})^{\mathrm{T}}\{\sigma_2^2 A_2(D_2)\}^{-1}(d_2 - \hat{\beta}_2 1_{n_2}). \tag{9}$$

Once these parameter values have been found we assume they are fixed.

2·5. *Extending the model for s code levels*

When there are more than two levels of code data, the normality of $z_s(.)$, conditional on all the hyperparameters and the observed runs, still holds. The mean and variance can easily be calculated from (2). The expressions in § 2·3 apply, but with different forms for V, $t(x)$, $h'(x)$, H and $c(x, x')$. To simplify the notation, we define

$$\Pi_i^j = \prod_{n=i}^{j} \rho_n.$$

The V matrix will have s blocks. The $(1, 1)$ block is simply $V^{(1,1)} = \sigma_1^2 A_1(D_1)$. For $k > 1$ the (k, k) block is

$$\begin{aligned} V^{(k,k)} = {} & \sigma_k^2 A_k(D_k) + \rho_{k-1}^2 \sigma_{k-1}^2 A_{k-1}(D_k) + \rho_{k-1}^2 \rho_{k-2}^2 \sigma_{k-2}^2 A_{k-2}(D_k) \\ & + \ldots + (\Pi_1^{k-1})^2 \sigma_1^2 A_1(D_k), \end{aligned}$$

and for $k < l$ the off-diagonal (k, l) block is given by

$$\begin{aligned} V^{(k,l)} = {} & \Pi_k^{l-1} \sigma_k^2 A_k(D_k, D_l) + \rho_{k-1} \Pi_{k-1}^{l-1} \sigma_{k-1}^2 A_{k-1}(D_k, D_l) + \Pi_{k-2}^{k-1} \Pi_{k-2}^{l-1} \sigma_{k-2}^2 A_{k-2}(D_k, D_l) \\ & + \ldots + \Pi_2^{k-1} \Pi_2^{l-1} \sigma_2^2 A_2(D_k, D_l) + \Pi_1^{k-1} \Pi_1^{l-1} \sigma_1^2 A_1(D_k, D_l). \end{aligned}$$

The $t(x)$ vector can be written as $t(x)^{\mathrm{T}} = (t_1(x)^{\mathrm{T}}, \ldots, t_s(x)^{\mathrm{T}})$, where, for $i = 2, \ldots, s$, $t_i(x)$ is constructed using the relation

$$t_i(x) = \rho_{i-1} t'_{i-1}(x) + \Pi_i^s A_i(x, D_i) \tag{10}$$

and $t_1(x) = \Pi_1^s \sigma_1^2 A_1(x, D_1)$. Here $t'_{i-1}(x)$ is used to denote the subset of elements from $t_{i-1}(x)$ corresponding to the elements of D_{i-1} that are also in D_i. For $h(x) = 1$, we have

$h'(x)^{\mathrm{T}} = (\Pi_1^{s-1}, \Pi_2^{s-1}, \ldots, \rho_{s-1}, 1)$, and the H matrix is lower diagonal, given by

$$H = \begin{pmatrix} 1_{n_1} & & & & & & \\ \rho_1 1_{n_2} & 1_{n_2} & & & & 0 & \\ \rho_1\rho_2 1_{n_3} & \rho_2 1_{n_3} & 1_{n_3} & & & & \\ \Pi_1^3 1_{n_4} & \rho_2\rho_3 1_{n_4} & \rho_3 1_{n_4} & 1_{n_4} & & & \\ & \vdots & & & & & \\ \Pi_1^{s-1} 1_{n_s} & & \ldots & & \rho_{s-2}\rho_{s-1} 1_{n_s} & \rho_{s-1} 1_{n_s} & 1_{n_s} \end{pmatrix}.$$

We also have

$$c(x, x') = c_s(x, x') + \rho_{s-1}^2 c_{s-1}(x, x') + \rho_{s-1}^2\rho_{s-2}^2 c_{s-2}(x, x') + \ldots + (\Pi_1^{s-1})^2 c_1(x, x').$$

For estimating the hyperparameters of the Bayesian model, the generalisation from (9) is straightforward.

3. Uncertainty analysis

Typically, the code will be used to predict the real phenomenon in a situation where some or all of the inputs are unknown. In general, suppose that the model input is a random vector X, with probability distribution G. The input may involve unknown physical constants or the result of another process, for example. Uncertainty analysis is a way of measuring how uncertainty in the outputs is induced by this uncertainty in the inputs.

The distribution of $z_2(X)$, resulting from the uncertainty about X, is known as the uncertainty distribution. The conventional approach to uncertainty analysis uses a Monte Carlo technique: first a random sample of inputs is generated from G, and then the top level code $z_2(.)$ is evaluated at each of these inputs, yielding a sample from the uncertainty distribution. Various summaries of the uncertainty distribution can then be estimated from this sample. Accurate results using this method may require a very large number of runs, since Monte Carlo makes inefficient use of the data. When $z_2(.)$ is very expensive to run, the number of runs will be limited, resulting in poor Monte Carlo estimates.

Using the Bayesian approach, we hope to achieve accurate results from a relatively small number of code runs. For a single level code, this objective was achieved by Haylock & O'Hagan (1996). We now generalise these results, to take into account data from several levels of code.

It is important to note that we assume the Gaussian process model for $z_2(.)$ after the parameters have been transformed, such that $G(x) \sim N(0, I)$.

Let K and L denote the mean and variance of $z_2(X)$, respectively, so that

$$K = \int_{\mathscr{X}} z_2(x)\, dG(x)$$

and $L = K_2 - K^2$, where

$$K_2 = E_X\{z_2(x)^2\} = \int_{\mathscr{X}} z_2(x)^2\, dG(x).$$

After observing the code values, inferences about K and L are derived from the posterior distribution we have for $z_2(x)$. This technique is similar to performing Bayesian quadrature (O'Hagan, 1991). Details are given in the Appendix.

4. Hydrocarbon reservoir example

We consider part of the dataset used by Craig et al. (1996), which consists of outputs from a code used to simulate the oil production and pressure at three wells in a hydrocarbon reservoir, based on a finite element analysis. The reservoir is split into five regions, each with different characteristics. The code takes as inputs the porosity and permeability of the rock in each of the five regions, and produces as output a number of time series giving various measures of production for each of the three wells. The simulator can be run at different levels of complexity by altering the resolution of the finite element grid. The particular outputs we consider are well pressure readings from a particular well at a single time-point, using two codes $z_1(.)$ and $z_2(.)$. The relatively simple code $z_1(.)$ has a coarse finite element grid for the simulation process. This is a fast approximation to the more complex $z_2(.)$, which uses a fine grid. Craig et al. (1996) use the fast simulator to formulate prior beliefs about the reservoir, as part of a more general procedure that also incorporates prior beliefs elicited from experts. A Latin hypercube design of 180 points was generated in the 10-dimensional input space, and both simulators were run for each of these input configurations.

In the following experiments, a subset of the output values were used as data, and the remainder were used to assess the accuracy of the interpolator by comparing it with the known true output values from the slow simulator at these points. From the 180 design points we select a subset of 45 by removing 135 points one at a time by repeating the following simple algorithm.

Algorithm

Step 1. *Calculate the distance between each possible pair of points left in the design.*

Step 2. *Select the pair for which the distance is smallest, and remove the point that is further, of these two, from the centre of the design region.*

The resulting 45-point design is well spaced in the input space, and we used this as our fast-code design. Starting with this design, we repeated the above procedure to select a subset of 7 points at which to observe the slow code. The 135 points that were not used to estimate $z_1(.)$ were used to measure the prediction accuracy by calculating the root mean squared error, RMSE. We first compared $\hat{\rho}_1 \hat{z}_1(.) + \hat{\delta}_2(.)$ with the actual computer experiment $z_1(.)$ to see if the estimated inadequacy process $\hat{\delta}_2(.)$ gives any improvement, and obtained RMSE $= 32{\cdot}3$ and RMSE $= 266{\cdot}5$. In this case we see that estimating model inadequacy results in more accurate prediction if higher level code data are available. The predicted values are plotted against the corresponding actual values in Fig. 1, for both prediction strategies. Also note that calculation of $\hat{\rho}_1 \hat{z}_1(x) + \hat{\delta}_2(x)$ will typically be much cheaper than for $z_1(x)$. Even better accuracy can be achieved if we use $\hat{\rho}_1 z_1(.) + \hat{\delta}_2(.)$, which gives RMSE $= 29{\cdot}9$.

It is clear from Fig. 1 that there is a negative correlation between the fast code and the model inadequacy correction. Each slow-code value is overpredicted by the fast code, with the largest overpredictions occurring for large slow-code values. The inclusion of ρ_1, estimated as $0{\cdot}71$, is therefore important for this particular dataset. If $\rho_1 = 1$ is fixed, for example, then we obtain RMSE $= 52{\cdot}4$ and RMSE $= 50{\cdot}8$ using $\hat{z}_1(.) + \hat{\delta}_2(.)$ and $z_1(.) + \hat{\delta}_2(.)$, respectively.

One could argue that we should simply interpolate the runs of the more complex code $z_2(.)$ and ignore $z_1(.)$. If many runs of $z_2(.)$ are available, then this approach will give accurate predictions without any runs from faster versions of the code. In the example presented above, Bayesian interpolation of the seven slow-code runs gives RMSE $= 51{\cdot}3$,

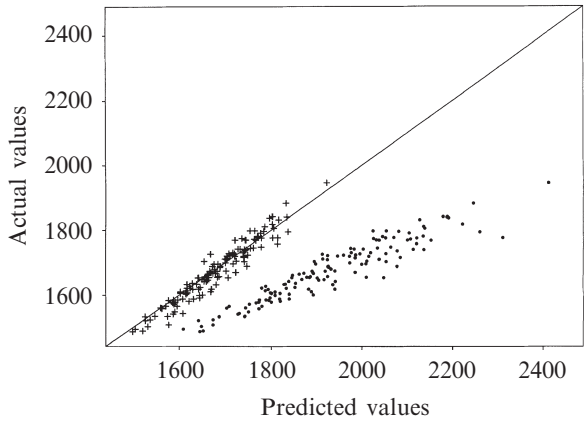

Fig. 1: Hydrocarbon reservoir example. Predicted and actual values of $z_2(.)$ using $\rho_1\hat{z}_1(.)+\hat{\delta}_2(.)$, shown by crosses, and the fast code $z_1(.)$, circular points.

which is better than the fast-code prediction alone, but not as good as the corrected predictors. It is interesting to consider how many more runs of the slow-code predictor would be necessary in order for this interpolator to compete with the adjusted predictor $\hat{\rho}_1 z_1(.)+\hat{\delta}_2(.)$. We constructed a series of Bayesian interpolators, using slow-code designs with 8, . . . , 15 points, and calculated RMSE for each design set. With 15 points we achieved RMSE $= 29{\cdot}2$, which is slightly smaller than the RMSE $= 29{\cdot}9$ achieved using $\hat{\rho}_1 z_1(.)+\hat{\delta}_2(.)$, but the cost of an additional 8 runs of the expensive slow code would be too high in this particular application.

As part of the investigation of Craig et al. (1996), geology experts were consulted and values were elicited for the mean and 95% probability bounds on each of the 10 input parameters. A multivariate normal distribution was fitted to the elicited prior information, to give $X \sim N(0, 0{\cdot}0651 I_{10})$ for the input parameters. The uncertainty analysis results are summarised by the values $E(K\,|\,z) = 1777$, $\mathrm{var}(K\,|\,z) = 1315$, $E(L\,|\,z) = 6770$ and $\mathrm{var}(L\,|\,z) = 2294008$. For comparison, the most accurate estimates of K and L we can obtain, together with the corresponding variances, are those found using a standard Bayesian uncertainty analysis on $z_2(.)$ with all 180 runs of the slow code. This resulted in $E(K\,|\,z) = 1747{\cdot}96$, $\mathrm{var}(K\,|\,z) = 118{\cdot}12$, $E(L\,|\,z) = 5427$ and $\mathrm{var}(L\,|\,z) = 305739$. The variances are smaller in this case, as expected, but the results from the autoregressive model are reasonably good considering they use only 7 runs of the slow code and 45 runs of the fast code.

5. A CUMULATIVE ROUGHNESS MODEL FOR MULTI-LEVEL CODES

The autoregressive model of § 2 assumes that $z_t(.)$ and $z_{t-1}(.)$ are related through the regression parameter ρ_{t-1}. A more structured way of dealing with dependence between multi-level codes is to imagine that each code level is a particular case of a larger single code, which includes a complexity parameter t. This might arise if the physical process in question is simulated using a finite element technique, where the complexity and speed vary depending on the resolution of the element grid.

We suppose that $z(x, t)$ is a collection of computer codes, ranging from the simplest $z(x, 0) = z_0$, to the most accurate simulator of reality $z(x, 1) = z(x)$. Somewhere between these extremes we may be able to run various levels of the simulator, corresponding to

$t = t_1, \ldots, t_s$, where $0 < t_1 < \ldots < t_s \leqslant 1$. We write $t = (t_1, \ldots, t_s)$. We wish to make inference about $z(., t_s)$, based on data from runs of the code at lower levels. In any given example, we will assume without loss of generality that $t_s = 1$, and we use the shorthand $\delta = (x - x')^{\mathrm{T}}(x - x')$.

Our new model assumes that

$$z(x, t) = z_0 + \int_0^t \zeta_d(x, \tau)\, d\tau$$

for $0 < t < 1$, where the function $\zeta_d(x, \tau)$ is a Gaussian process with mean β_d and covariance function $\sigma_d^2 \exp(-k\tau\delta)$, and k is a positive constant. Finally we assume that $\zeta_d(x, \tau)$ and $\zeta_d(x, \tau')$ are independent for $\tau \neq \tau'$. Our covariance function corresponds to a prior belief that the code becomes rougher as the complexity increases.

The prior mean of $z(x, t)$ is $z_0 + \beta_d t$ and the prior covariance is

$$\operatorname{cov}\{z(x, t), z(x', t)\} = \sigma_d^2 \int_0^t \exp(-k\tau\delta)\, d\tau = \frac{\sigma_d^2}{k\delta}\{1 - \exp(-k\delta t)\}. \tag{11}$$

The more general covariance function, for comparing codes of different levels, is given by

$$\operatorname{cov}\{z(x, t), z(x', t')\} = \frac{\sigma_d^2}{k\delta}\{1 - \exp(-k\delta t^*)\},$$

where $t^* = \min(t, t')$.

If we make the change of variable $u^2 = k\tau$, we can write the correlation function for $z(., t)$ as

$$c(\delta) = \int_{-\infty}^{\infty} \exp(-u^2\delta)\, dF(u), \tag{12}$$

where dF is the probability measure

$$dF(u) = \begin{cases} 2u/(kt) & \text{for } 0 \leqslant u \leqslant \sqrt{(kt)}, \\ 0 & \text{otherwise.} \end{cases}$$

Writing $c(\delta)$ in this form shows that it is a valid correlation function, using a result of Schoenberg (1938), see Matérn (1986, p. 17), which states that the class of continuous isotropic correlation functions valid in any dimension is the class of probability mixtures of Gaussian-type correlations of the form (12). This $c(\delta)$ function tends to 0 much more slowly than the exponential form (3), implying that distant points are more correlated under this model than they would be under the autoregressive model.

The model described above has $4 + s$ parameters z_0, β_d, $t_1, \ldots, t_s$, σ_d^2 and k, each of which needs to be estimated in some way. We adopt a simple strategy based on the variogram, as used in the classical kriging theory, popular in geostatistics. Specifically, we use the robust variogram estimator of Cressie & Hawkins (1980) to fit the parameters of the covariance function (11).

The prior distribution of $z(x, t)$ can be expressed as

$$z(x,t), z(x',t') | z_0, \beta_d, \sigma_d^2, k$$

$$\sim N\left(z_0 1_2 + \beta_d \begin{pmatrix} t \\ t' \end{pmatrix}, \sigma_d^2 t^* \begin{pmatrix} 1 & \dfrac{1-\exp(-k\delta t^*)}{k\delta t^*} \\ \dfrac{1-\exp(-k\delta t^*)}{k\delta t^*} & 1 \end{pmatrix} \right),$$

where $t^* = \min(t, t')$. The posterior distribution of $z(., t_s)$ is

$$z(., t_s) | z_0, \beta_d, \sigma_d^2, k, t, d \sim N(m_s'(.), v_s'(., .)),$$

where

$$m_s'(x) = z_0 + \beta_d t_s + t_s(x)^{\mathrm{T}} V^{-1}(d - m_d),$$

$$t_s(x)^{\mathrm{T}} = \mathrm{cov}\{z(x, t_s), d\} = \sigma_d^2(t_1 A_1(x, D_1), \ldots, t_s A_s(x, D_s)),$$

$$v_s'(x, x') = \frac{\sigma_d^2}{k\delta}\{1 - \exp(-k t_s \delta)\} - t_s(x)^{\mathrm{T}} V^{-1} t_s(x'),$$

$$m_d^{\mathrm{T}} = z_0 1_{n_1 + \ldots + n_s}^{\mathrm{T}} + \beta_d (t_1 1_{n_1}^{\mathrm{T}} \quad \ldots \quad t_s 1_{n_s}^{\mathrm{T}}),$$

$$V = \sigma_d^2 \begin{pmatrix} t_1 A_1(D_1, D_1) & \ldots & t_1 A_1(D_1, D_s) \\ \vdots & & \vdots \\ t_1 A_1(D_s, D_1) & \ldots & t_s A_s(D_s, D_s) \end{pmatrix}.$$

The (i, j) element of $A_h(D_l, D_m)$ is

$$(k t_h \| x_i^{(l)} - x_j^{(m)} \|)^{-1} \{1 - \exp(-k t_h \| x_i^{(l)} - x_j^{(m)} \|)\}, \tag{13}$$

where $x_i^{(l)}$ is the ith element of D_l and $\| x_i^{(l)} - x_j^{(m)} \|$ is $(x_i^{(l)} - x_j^{(m)})^{\mathrm{T}}(x_i^{(l)} - x_j^{(m)})$. When $x_i^{(l)} = x_j^{(m)}$, (13) reduces to 1. Note that random observation error on $z(x', t_s)$ can be included by adding λI to the (s, s) block of V, where λ is the Gaussian error variance. This would be appropriate if we were observing data from the real physical system and modelling it as the highest level code $z(x', t_s)$.

Uncertainty analysis for $z(X, t_s)$ is not as tractable as the corresponding analysis using the autoregressive model, because of the less tractable form for the correlation function $c(.)$. However, it is possible by using various approximations. The details are beyond the scope of this paper.

Example. For a univariate input x, we now present the analysis of a simulated three-level code which has the kind of behaviour described above, in which the roughness increases as more complexity is added. It should be noted that we do not use the model to generate the code data. First, we generate a 'true' function $y(x)$ as a realisation of a rough Gaussian process with mean $m(x) = 0{\cdot}2x - 0{\cdot}07x^2$ and covariance function $c(x, x') = 4\exp\{-200(x - x')^2\}$. From this, we simulate a vector of 101 points $y^{\mathrm{T}} = (y(0), y(0{\cdot}01), \ldots, y(1))$. We then define $y^*(x) = E\{y(x) | y\}$ and we let $z(x, t_3) = y^*(x)$.

We now create the lower-level codes $z(x, t_s)$ for $s = 1, 2$. For given s, we evaluate

$$y_s^{\mathrm{T}} = (y^*\{1/(3 \times 2^s)\}, y^*\{2/(3 \times 2^s)\}, \ldots, y^*\{1 - 1/(3 \times 2^s)\}),$$

and define $z(x, t_s)$ to be the posterior mean if we model these data as a Gaussian process with mean β and covariance $\sigma^2 \exp\{-2(x - x')^2\}$ and assume a small observation error. The resulting simulated three-level code is plotted in Fig. 2.

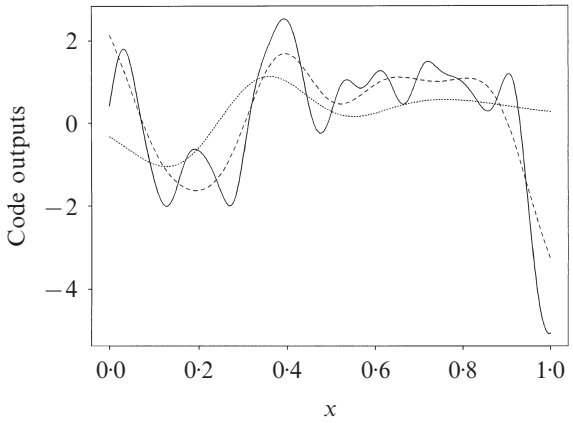

Fig. 2: Example. Simulated code with three levels: $z(x, t_1)$, dotted line, $z(x, t_2)$, dashed line, and $z(x, t_3)$, solid line.

The data consist of runs from each of the 3 levels at 15 design points 0, $\frac{1}{14}$, $\frac{2}{14}$, ..., 1. Analysis of the three level data using the cumulative roughness model produces parameter estimates $k = 3{\cdot}41$, $t = (0{\cdot}25, 0{\cdot}63, 1)$ and $\sigma_d^2 = 38{\cdot}1$. We obtain RMSE $= 1{\cdot}19$. For comparison, using the autoregressive model of § 2·1 with the same data gives RMSE $= 1{\cdot}44$, and, if we do not include any runs of the fast codes but simply interpolate the top level data with a Bayesian smoother, we obtain RMSE $= 1{\cdot}54$. For this example the use of information from the fast codes results in improved prediction of the top level code, particularly if we use the cumulative roughness model.

6. Discussion

The Bayesian methods of prediction and uncertainty analysis presented here can be adapted and applied to a wide range of multi-level computer codes. The ability to combine information from runs of the code at different levels is particularly useful when the slowest code is very expensive to run, as in the case of the oil reservoir example. Each run of the slow code in this case can take a number of days. Craig et al. (1996) estimate that about 36 runs of the fast code can be made in the time taken for a single run of the slow code, so that our 45 fast-code runs are worth 1·25 slow runs. It was shown in § 4 that using the Bayesian autoregression model and this extra '1·25 runs' gives a value for RMSE that is equivalent to using an additional 8 slow runs and the usual Bayesian model.

We have also shown how the uncertainty analysis techniques of Haylock & O'Hagan (1996) can be combined with the multi-level code model. Unfortunately the code is too expensive to obtain any true values with which to compare our uncertainty analysis estimates in § 4, although the uncertainty analysis results using the method of Haylock & O'Hagan (1996), based on 180 runs of the slow code, are consistent with our method.

We have demonstrated the use of two quite different models to deal with multi-level codes, corresponding to different prior beliefs about the way in which the levels are correlated. The choice of model will be specific to the application, and the cumulative roughness model might be modified to deal with a range of beliefs about the nature of $z(., t)$ as more complexity is added.

There are a number of ways in which we might extend the methods presented here. We

have assumed that the autoregression parameters ρ_i and the covariance parameters are fixed. This is common when analysing computer experiments. However, in a context where each run of the code is very expensive it is acceptable to use computationally intensive methods, such as Markov chain Monte Carlo, to analyse the outputs using a full Bayesian model. This would allow expert prior knowledge to be included in the analysis. In the case where many levels are included, it may be difficult to obtain prior information. More work is needed to find ways of estimating the autoregression parameters ρ_i and the parameters of the $\delta_t(.)$ functions, particularly for larger values of t, where data is relatively sparse.

Much of the computer experiments literature addresses the problem of choosing good design points. We have used design sets for the different levels such that $D_t \subseteq D_{t-1}$. In some ways this seems sensible. We effectively observe the difference between $z_t(.)$ and $z_{t-1}(.)$ at each point in D_t, and this strategy should be more robust to misspecification of the covariance parameters of $\delta_t(.)$. If this condition were relaxed the analysis would still be tractable, and we might want to investigate alternative strategies.

ACKNOWLEDGEMENT

This research has been supported by the Engineering and Physical Sciences Research Council, with an additional financial contribution from the National Radiological Protection Board. We would like to thank Peter Craig, Michael Goldstein and their colleagues at the University of Durham and Scientific Software Intercomp for providing us with the data for the oil reservoir example of § 4, and also to thank the editor and referees who suggested improvements to earlier versions of the paper.

APPENDIX

Uncertainty analysis

Let K denote the mean of $z_2(X)$, given by

$$K = \int_{\mathscr{X}} z_2(x)\, dG(x).$$

Conditioning on the hyperparameters ϕ, we have $K|\phi, z \sim N(\hat{K}, W)$, where

$$\hat{K} = \int_{\mathscr{X}} m'(x)\, dG(x) = h\hat{\beta} + TV^{-1}(z - H\tilde{\beta}),$$

$$\begin{aligned} W &= \int_{\mathscr{X}} \int_{\mathscr{X}} c'(x, x')\, dG(x)\, dG(x') \\ &= U - TV^{-1}T^{\mathrm{T}} + (h - TV^{-1}H)(H^{\mathrm{T}}V^{-1}H)^{-1}(h - TV^{-1}H)^{\mathrm{T}}, \end{aligned}$$

$$h = \int_{\mathscr{X}} h'(x)^{\mathrm{T}}\, dG(x), \quad T = \int_{\mathscr{X}} t(x)^{\mathrm{T}}\, dG(x), \quad U = \int_{\mathscr{X}} \int_{\mathscr{X}} c(x, x')\, dG(x)\, dG(x').$$

In our examples we will assume that $h(x) = (1)$, in which case the h vector is simply $(\rho_1, 1)$. Closed forms for T and U are obtained by completing the square in the integrands to leave standard forms. Using (6) we can write $T = (T^{(1)}, T^{(2)})$, where the ith element of $T^{(1)}$ is

$$T_i^{(1)} = \rho_1 \sigma_1^2 (1 + 2b_1)^{-p/2} \exp\left(- \frac{b_1 x_i^{(1)\mathrm{T}} x_i^{(1)}}{1 + 2b_1} \right) \quad (i = 1, \ldots, n_1),$$

and the ith element of $T^{(2)}$ is

$$T_i^{(2)} = \rho_1^2\sigma_1^2(1+2b_1)^{-p/2}\exp\left(-\frac{b_1 x_i^{(1)\mathrm{T}} x_i^{(1)}}{1+2b_1}\right)$$
$$+\sigma_2^2(1+2b_2)^{-p/2}\exp\left(-\frac{b_2 x_i^{(2)\mathrm{T}} x_i^{(2)}}{1+2b_2}\right) \quad (i=1,\ldots,n_2).$$

The U integral reduces to $\rho_1^2\sigma_1^2(1+4b_1)^{-p/2}+\sigma_2^2(1+4b_2)^{-p/2}$. The distribution of the variance L given ϕ, z cannot be found in closed form. It is nevertheless possible to derive expressions for $E(L|\phi,z)$ and $\mathrm{var}(L|\phi,z)$, following the approach of Haylock & O'Hagan (1996). Details are available from the authors.

References

Aslett, R., Buck, R. J., Duvall, S. G., Sacks, J. & Welch, W. J. (1998). Circuit optimization via sequential computer experiments. Design of an output buffer. *Appl. Statist.* **47**, 31–48.

Blight, B. J. N. & Ott, L. (1975). A Bayesian approach to model inadequacy for polynomial regression. *Biometrika* **62**, 79–88.

Craig, P. S., Goldstein, M., Seheult, A. H. & Smith, J. A. (1996). Bayes linear strategies for matching hydrocarbon reservoir history. In *Bayesian Statistics 5*, Ed. J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, pp. 69–95. Oxford University Press.

Cressie, N. & Hawkins, D. M. (1980). Robust estimation of the variogram, I. *J. Int. Assoc. Math. Geol.* **12**, 115–25.

Currin, C., Mitchell, T., Morris, M. & Ylvisaker, D. (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *J. Am. Statist. Assoc.* **86**, 953–63.

Diaconis, P. (1988). Bayesian numerical analysis. In *Statistical Decision Theory and Related Topics IV*, Ed. S. S. Gupta and J. Berger, pp. 163–75. New York: Springer.

Haylock, R. & O'Hagan, A. (1996). On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. In *Bayesian Statistics 5*, Ed. J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, pp. 629–37. Oxford University Press.

Kimeldorf, G. S. & Wahba, G. (1970). A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Ann. Math. Statist.* **41**, 495–502.

Matérn, B. (1986). *Spatial Variation*, 2nd ed., Lecture Notes in Statistics Vol. 36. New York: Springer-Verlag.

Morris, M. D., Mitchell, T. J. & Ylvisaker, D. (1993). Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics* **35**, 243–55.

Neal, R. M. (1999). Regression and classification using Gaussian process priors (with Discussion). In *Bayesian Statistics 6*, Ed. J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, pp. 475–501. Oxford University Press.

O'Hagan, A. (1978). Curve fitting and optimal design for prediction (with Discussion). *J. R. Statist. Soc.* B **40**, 1–42.

O'Hagan, A. (1991). Bayes-Hermite quadrature. *J. Statist. Plan. Infer.* **29**, 245–60.

O'Hagan, A. (1992). Some Bayesian numerical analysis (with Discussion). In *Bayesian Statistics 4*, Ed. J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, pp. 345–63. Oxford University Press.

O'Hagan, A., Kennedy, M. C. & Oakley, J. E. (1999). Uncertainty analysis and other inference tools for complex computer codes (with Discussion). In *Bayesian Statistics 6*, Ed. J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, pp. 503–24. Oxford University Press.

Sacks, J., Welch, W. J., Mitchell, T. J. & Wynn, H. P. (1989). Design and analysis of computer experiments (with Discussion). *Statist. Sci.* **4**, 409–35.

Schoenberg, I. J. (1938). Metric spaces and completely monotone functions. *Ann. Math.* **39**, 811–41.

Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J. & Morris, M. D. (1992). Screening, predicting, and computer experiments. *Technometrics* **34**, 15–25.

Young, P., Parkinson, S. & Lees, M. (1996). Simplicity out of complexity in environmental modelling: Occam's razor revisited. *J. Appl. Statist.* **23**, 165–210.

[*Received September* 1998. *Revised June* 1999]