Appl. Statist. (2004)
53, *Part* 1, *pp.* 83–93

Estimating percentiles of uncertain computer code outputs

Jeremy Oakley

University of Sheffield, UK

[Received June 2001. Final revision June 2003]

Summary. A deterministic computer model is to be used in a situation where there is uncertainty about the values of some or all of the input parameters. This uncertainty induces uncertainty in the output of the model. We consider the problem of estimating a specific percentile of the distribution of this uncertain output. We also suppose that the computer code is computationally expensive, so we can run the model only at a small number of distinct inputs. This means that we must consider our uncertainty about the computer code itself at all untested inputs. We model the output, as a function of its inputs, as a Gaussian process, and after a few initial runs of the code use a simulation approach to choose further suitable design points and to make inferences about the percentile of interest itself. An example is given involving a model that is used in sewer design.

Keywords: Deterministic computer code; Gaussian process; Uncertainty distribution

1. Introduction

This paper is motivated by a specific problem that was experienced by the Water Research Centre (WRC). A computer model, known as the SIMPOL model, is to be used to predict some quantity of interest y given a set of input parameters $\mathbf{x}$. We denote this model by the function $y=\eta(\mathbf{x})$, and we shall refer to any particular choice of values for the input parameters $\mathbf{x}$ as the inputs. The model is deterministic, so, if it is run repeatedly at the same inputs, it will always return the same output. However, the input parameters represent physical constants, the true values of which are unknown. Denoting these true values by $\mathbf{X}$, it then follows that the output at the true inputs, which we denote by $Y=\eta(\mathbf{X})$, is also unknown. The WRC are interested in extreme events that are related to the output quantity and wish to know the 95th percentile of the distribution of Y.

The interest is in whether it is possible to obtain an accurate estimate of the 95th percentile for a computationally expensive computer model, i.e. based on a small number of runs of the computer code. The full model has at least 10 uncertain input parameters. We develop methodology for estimating the 95th percentile and test it on a simplified example where only four input parameters are considered uncertain, and the rest are kept fixed. In this example we can obtain the true value of the 95th percentile, and so we can check the accuracy of the inferences that are obtained by our methods.

Making inference about the distribution of Y is known to users of computer codes as uncertainty analysis, and in this particular case we wish to know the 95th percentile of the uncertainty distribution. This is an example of a variety of statistical problems that are encountered by users

Address for correspondence: Jeremy Oakley, Department of Probability and Statistics, Hicks Building, University of Sheffield, Houndsfield Road, Sheffield, S3 7RH, UK.
E-mail: j.oakley@sheffield.ac.uk

 0035–9254/04/53083

of deterministic computer models. Other problems involve choosing design points to run computationally expensive codes at to minimize the uncertainty about the output at the remaining untested inputs, assessing the sensitivity of the output to specific inputs and calibrating models to observations of reality. Some examples of the methodology that has been developed to tackle these problems are Sacks *et al.* (1989), Craig *et al.* (1996), O'Hagan *et al.* (1999), Saltelli *et al.* (2000) and Kennedy and O'Hagan (2001).

For computationally cheap computer codes, a simple Monte Carlo approach can be used to determine the distribution of Y. Denoting the distribution of $\mathbf{X}$ by G, we draw a very large sample of inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from G and run the code at each input to obtain a sample from the distribution of Y. For computationally expensive models, this method is impractical; hence we are considering how to estimate the percentile on the basis of a small number of runs of the code.

There are three stages to the method that is presented here. The first problem is that we can evaluate the computer code only at a small number of distinct inputs. To obtain an accurate estimate of the percentile, we shall need to extract as much information as possible from each run of the computer code, i.e. we shall need a statistical model for the output of the code at untested inputs given these runs. Secondly, given this model and the distributions for the unknown inputs, we shall need a method for making inferences about the 95th percentile of the distribution of the output. Finally, given the method we shall need to consider how to choose the initial inputs to run the model at so that we can estimate the 95th percentile as accurately and as efficiently as possible. In Oakley and O'Hagan (2002) a technique was described that can be used to make inference about any summary of the distribution of Y based on a small number of runs of the code. Consequently, in this paper we shall be concentrating on the third stage to the method, which was not addressed in Oakley and O'Hagan (2002) for the case of estimating percentiles. Given the expense in obtaining runs of the computer code in these problems a careful consideration of the design can be advantageous. The design is particularly important for the 95th or more extreme percentiles because we need to balance obtaining information about $\eta(\cdot)$ everywhere with obtaining information about $\eta(\cdot)$ in a rather unrepresentative region, in some sense.

In the next section we describe the SIMPOL model and the uncertain inputs. In Section 3 we discuss inference about functions using a Gaussian process. Estimating percentiles by using the Gaussian process model is described in Section 4, and a method for choosing design points is developed in Section 5. The approach is applied to the SIMPOL model in Section 6.

2. The SIMPOL model

The SIMPOL model is used as an aid in the design of combined sewer overflows (CSOs) that are required to meet certain environmental standards. A combined sewer carries both sewage and surface-water from urban drainage. These sewers are common in the UK. In a storm, the excess of water may lead to an increase in pressure in the sewer pipes and a risk of flooding. To counter this, overflows (CSOs) are built into the sewer system at critical points. These overflows then discharge the excess water into rivers and streams. Clearly, there are environmental concerns when sewage is being discharged. Currently, CSOs are being upgraded to reduce both the number and the volume of spills. One method is to provide storage for the storm flow at the CSO. Once the storm has subsided, the stored flow can then continue through the sewer system. The output of the SIMPOL model is the volume of storage that is required to meet specific environmental standards. These standards usually involve the condition of the river following a spill from the CSO.

2.1. The model input parameters

Four input parameters in the model are treated as unknown and the remainder are kept fixed. These are as follows:

(a) the maximum pass forward rate at the CSO, the capacity of the sewer downstream of the CSO—this determines the flow rate at which the storage starts to fill;
(b) the average dry weather biochemical oxygen demand (BOD) of the sewer flow—the breakdown of organic matter uses oxygen, and the BOD is a measure of the potential potency of the oxygen demand, expressed as a concentration;
(c) the BOD sediment load, the maximum load that can be built up in a sewer, if sufficient deposition occurs—in dry weather, sedimentation can occur in sewers, and during a storm large sewer flows can erode the sediment, causing an increase in the BOD of the storm sewage;
(d) the BOD erosion concentration, the rate at which the sediment load is eroded.

The WRC have assumed independent log-normal distributions for the true values of all four inputs. We transform these inputs so that we can think of the output of the model as being a function of four inputs x_1, x_2, x_3 and x_4, where the true values all have standard normal distributions. Note that, for certain values of the inputs, the required environmental standards are met or even exceeded without the need for any extra storage volume. The SIMPOL model returns a negative output, which is then corrected to 0. For our purposes, we retain the original negative outputs, as these will still give us information about how the output varies with the inputs.

3. Inference about functions using Gaussian processes

We now describe the Gaussian process model for an unknown function $\eta(\cdot)$. Gaussian processes have been used before for modelling computer codes, and examples are Currin *et al.* (1991) and Haylock and O'Hagan (1996). The key requirement is that $\eta(\cdot)$ is a smooth function, so if we know the value of $\eta(\mathbf{x})$ we should have some idea about the value of $\eta(\mathbf{x}')$ for $\mathbf{x}$ close to $\mathbf{x}'$. It is this property of $\eta(\cdot)$ that will give us the opportunity to improve on Monte Carlo sampling, since the extra information that is available after each code run is ignored in the Monte Carlo approach.

For any set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we represent our uncertainty about $\{\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)\}$ through a multivariate normal distribution. The mean of $\eta(\mathbf{x})$ is given by

$$E\{\eta(\mathbf{x})|\boldsymbol{\beta}\} = \mathbf{h}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\beta}, \tag{1}$$

conditional on $\boldsymbol{\beta}$. The vector $\mathbf{h}(\cdot)$ consists of q known regression functions of $\mathbf{x}$, and $\boldsymbol{\beta}$ is a vector of coefficients. The choice of $\mathbf{h}(\cdot)$ is arbitrary, though it should be chosen to incorporate any beliefs that we might have about the form of $\eta(\cdot)$. The covariance between $\eta(\mathbf{x})$ and $\eta(\mathbf{x}')$ is given by

$$\operatorname{cov}\{\eta(\mathbf{x}), \eta(\mathbf{x}')|\sigma^2\} = \sigma^2\, c(\mathbf{x}, \mathbf{x}'), \tag{2}$$

conditional on σ^2, where $c(\mathbf{x}, \mathbf{x}')$ is a function which decreases as $|\mathbf{x} - \mathbf{x}'|$ increases and also satisfies $c(\mathbf{x}, \mathbf{x}) = 1\ \forall \mathbf{x}$. The function $c(\cdot, \cdot)$ must ensure that the covariance matrix of any set of outputs $\{y_1 = \eta(\mathbf{x}_1), \ldots, y_n = \eta(\mathbf{x}_n)\}$ is positive semidefinite. A typical choice is

$$c(\mathbf{x}, \mathbf{x}') = \exp\{-(\mathbf{x} - \mathbf{x}')^{\mathrm{T}} B(\mathbf{x} - \mathbf{x}')\}, \tag{3}$$

where B is a diagonal matrix of (positive) roughness parameters. Conventionally, a weak prior for $\boldsymbol{\beta}$ and σ^2 in the form $p(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{-2}$ is used. In Oakley (2002) a means of including proper prior information about the function $\eta(\cdot)$ is presented, through the use of the conjugate prior, the normal inverse gamma distribution. We have

$$p(\boldsymbol{\beta}, \sigma^2) \propto (\sigma^2)^{-(d+q+2)/2} \exp[-\{(\boldsymbol{\beta} - \mathbf{z})^{\mathrm{T}} V^{-1} (\boldsymbol{\beta} - \mathbf{z}) + a\}/2\sigma^2]. \tag{4}$$

(Recall that q is the number of regressors in the mean function.)

The output of $\eta(\cdot)$ is observed at n design points, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, to obtain data $\mathbf{y}$. Given the prior in expression (4) it can be shown that

$$\frac{\eta(\mathbf{x}) - m^*(\mathbf{x})}{\hat{\sigma}\sqrt{c^*(\mathbf{x}, \mathbf{x})}} | \mathbf{y}, B \sim t_{d+n}, \tag{5}$$

where

$$m^*(\mathbf{x}) = \mathbf{h}(\mathbf{x})^{\mathrm{T}} \hat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x})^{\mathrm{T}} A^{-1} (\mathbf{y} - H\hat{\boldsymbol{\beta}}), \tag{6}$$

$$c^*(\mathbf{x}, \mathbf{x}') = c(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})^{\mathrm{T}} A^{-1} \mathbf{t}(\mathbf{x}') + (\mathbf{h}(\mathbf{x})^{\mathrm{T}} - \mathbf{t}(\mathbf{x})^{\mathrm{T}} A^{-1} H)(H^{\mathrm{T}} A^{-1} H)^{-1} (\mathbf{h}(\mathbf{x}')^{\mathrm{T}} - \mathbf{t}(\mathbf{x}')^{\mathrm{T}} A^{-1} H)^{\mathrm{T}}. \tag{7}$$

$$\mathbf{t}(\mathbf{x})^{\mathrm{T}} = (c(\mathbf{x}, \mathbf{x}_1), \ldots, c(\mathbf{x}, \mathbf{x}_n)), \tag{8}$$

$$H^{\mathrm{T}} = (\mathbf{h}^{\mathrm{T}}(\mathbf{x}_1), \ldots, \mathbf{h}^{\mathrm{T}}(\mathbf{x}_n)), \tag{9}$$

$$A = \begin{pmatrix} 1 & c(\mathbf{x}_1, \mathbf{x}_2) & \ldots & c(\mathbf{x}_1, \mathbf{x}_n) \\ c(\mathbf{x}_2, \mathbf{x}_1) & 1 & & \\ \vdots & & \ddots & \vdots \\ c(\mathbf{x}_n, \mathbf{x}_1) & \cdots & & 1 \end{pmatrix}, \tag{10}$$

$$\hat{\boldsymbol{\beta}} = V^*(V^{-1}\mathbf{z} + H^{\mathrm{T}} A^{-1} \mathbf{y}), \tag{11}$$

$$\hat{\sigma}^2 = \{a + \mathbf{z}^{\mathrm{T}} V^{-1} \mathbf{z} + \mathbf{y}^{\mathrm{T}} A^{-1} \mathbf{y} - \hat{\boldsymbol{\beta}}^{\mathrm{T}} (V^*)^{-1} \hat{\boldsymbol{\beta}}\}/(n + d - 2), \tag{12}$$

$$V^* = (V^{-1} + H^{\mathrm{T}} A^{-1} H)^{-1}, \tag{13}$$

$$\mathbf{y}^{\mathrm{T}} = (\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)). \tag{14}$$

Full details of the prior-to-posterior analysis can be found in O'Hagan (1994). In this paper we shall simply condition on a posterior estimate of B, rather than taking into account the uncertainty that we may have. The estimate could be the posterior mode, though with a small sample of data the likelihood can be fairly flat. For the SIMPOL model example, we found a cross-validation procedure to be more effective; each observation was left out in turn and B was chosen to minimize the error between the posterior mean of the omitted output and the known true value. Ideally we should be allowing for the uncertainty in B, though Kennedy and O'Hagan (2001) have suggested that this uncertainty may not be important. Neal (1999) used Markov chain Monte Carlo sampling to sample from the posterior distribution of B.

4. Estimating percentiles by simulation

If $\eta(\cdot)$ was a computationally cheap function, then the required 95th percentile could be determined by using Monte Carlo sampling, in principle to an arbitrary degree of accuracy. Denote this value of the true 95th percentile by ν. It is important to appreciate that ν is the 95th percentile of the distribution of $Y|\eta(\cdot)$, and not the marginal distribution of Y. We could estimate the

95th percentile of the marginal distribution of Y by repeatedly sampling $\mathbf{x}$ from G, and then $\eta(\mathbf{x})$ from its posterior distribution given in equation (5), again in principle to an arbitrary degree of accuracy. But it would clearly be false to report to the WRC that we then knew ν with certainty; running the actual code more times might lead to a different estimate, as the distribution of $\eta(\mathbf{x})$ would change.

When we have uncertainty about $\eta(\cdot)$, the distribution of $Y|\eta(\cdot)$ is itself uncertain, and so ν is therefore also uncertain. We must both estimate ν and consider our uncertainty about ν. In achieving this, the key step involves generating a function $\eta(\cdot)$ from its posterior distribution, with the property that the function generated, denoted by $\eta_{(i)}(\cdot)$, is computationally cheap. We can then determine the 95th percentile of $Y|\eta_{(i)}$ by using Monte Carlo sampling. A sample from the distribution of ν can therefore be obtained as follows.

Step 1: generate a random function $\eta_{(i)}(\cdot)$ from the distribution of $\eta(\cdot)$, with the property that $\eta_{(i)}(\mathbf{x})$ can be evaluated quickly for any input $\mathbf{x}$.
Step 2: determine the 95th percentile of $\eta_{(i)}(\mathbf{X})$, denoted by $\nu_{(i)}$, by using Monte Carlo methods.
Step 3: repeat steps 1 and 2 to obtain a sample $\nu_{(1)}, \ldots, \nu_{(K)}$ from the distribution of ν.

The posterior distribution for the outputs at any set of inputs is multivariate t, with the mean for each output $m^*(\mathbf{x})$ and covariance between any two outputs $\hat{\sigma}^2\, c^*(\mathbf{x}, \mathbf{x}')$. Denote the sample space of $\mathbf{X}$ by $\mathcal{X}$. We cannot obtain an exact realization of $\eta(\cdot)$, since in practice the set $\mathcal{X}$ of possible values of $\mathbf{x}$ is infinite and to sample $\eta(\cdot)$ means to sample $\eta(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$. Given that we shall evaluate $\eta_{(i)}(\cdot)$ at only a finite sample of inputs to obtain $\nu_{(i)}$, we might think of just sampling $\eta(\cdot)$ jointly at all the inputs in the Monte Carlo sample. This can be unreliable because the variance–covariance matrix of all the corresponding outputs is ill conditioned. Instead we use the following procedure to obtain approximate draws from the distribution of $\eta(\cdot)$.

We choose N points $\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_N$, which we shall refer to as the simulation design points, distinct from the original n design points, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ that are used to obtain the data vector $\mathbf{y}$. For the ith realization from the distribution of $\eta(\cdot)$, denoted by $\eta_{(i)}(\cdot)$, we generate random data $\mathbf{y}^{(i)} = \{\eta_{(i)}(\mathbf{x}'_1), \ldots, \eta_{(i)}(\mathbf{x}'_N)\}$. For $\mathbf{x} \notin \{\mathbf{x}'_1, \ldots, \mathbf{x}'_N, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\eta_{(i)}(\mathbf{x})$ is still unknown. However, $\eta_{(i)}(\mathbf{x})|\mathbf{y}, \mathbf{y}^{(i)}$ also has a t-distribution as given in expression (5), with the variance of $\eta_{(i)}(\mathbf{x})$ very small for all $\mathbf{x}$ of interest, if $\mathbf{x}'_1, \ldots, \mathbf{x}_N$ are well chosen and N is sufficiently large. Consequently, we can now approximate $\eta_{(i)}(\cdot)$ by $m^*_{(i)}(\cdot)$, which is the posterior mean of $\eta_{(i)}(\cdot)$ given the original n observations and the N new sampled observations. For suitably chosen $\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_N$, the error in approximating $\eta_{(i)}(\cdot)$ by $m^*_{(i)}(\cdot)$ should be minimal. This process is then repeated to obtain a new realization, $\eta_{(j)}(\cdot)$.

5. Choice of design points

Various schemes have been suggested for choosing design points to run a computer code at, when the aim is to estimate the output over some range of inputs of interest. Examples are given in Sacks *et al.* (1989) and Haylock and O'Hagan (1996). In this case, however, the sole interest is in estimating the 95th percentile of the output. Consequently, we shall not necessarily need to know much information about the function $\eta(\cdot)$ over the entire input space. Suppose that we can identify a subset R of $\mathcal{X}$ such that we are almost certain that the 95th percentile of the output will occur at some input $\mathbf{x}$, with $\mathbf{x} \in R$, even though we still have uncertainty about $\eta(\cdot)$ over all of $\mathcal{X}$. In this case, we shall only require accurate information about $\eta(\cdot)$ over the region R. A novel approach to choosing the design points will be required here.

An outline of the design scheme is as follows. The design points will be chosen in two stages. The first set of design points is chosen with the purpose of learning about $\eta(\cdot)$ as a whole. Given the first set of runs, we then explore the posterior distribution of $\eta(\cdot)$ by using the simulation procedure that was described in the previous section, to identify regions of the input space that produce large output values. A second set of design points is then chosen to cover these regions, so that we can reduce our uncertainty about $\eta(\cdot)$ over the range of inputs that are relevant for estimating ν.

If we have proper prior information about $\eta(\cdot)$, then we can generate random functions from the prior distribution of $\eta(\cdot)$, and so the first set of design points may not be strictly necessary. However, depending on the strength of the prior knowledge, a few runs of the computer code to improve our overall understanding of $\eta(\cdot)$ may still be helpful before attempting to identify R. In the SIMPOL example, no proper prior knowledge was available to us, and so for the remainder of this section we shall assume that we have weak prior knowledge about $\eta(\cdot)$.

5.1. Choosing the first set of design points

As mentioned before, there are various schemes for choosing design points to learn about $\eta(\cdot)$ as a whole. Denote the data that are obtained after the code has been run at the first set of design points by $\mathbf{y}_1$. The suggestion in Haylock and O'Hagan (1996) was to choose design points to minimize the preposterior expected value

$$\int_{\mathcal{X}} \operatorname{var}\{\eta(\mathbf{x})|\mathbf{y}_1\}\,\mathrm{d}G(\mathbf{x}).$$

A drawback with this scheme (and certain others) is that it is a function of the unknown roughness parameters B, and so a prior estimate of B is needed. However, for common modelling choices regarding $G(\cdot)$ and $\eta(\cdot)$, and initial assumptions about the structure of the design, it can be computationally very convenient to use this criterion, given an estimate of B, and this was the scheme that was adopted in the SIMPOL example.

5.2. Choosing the second set of design points

We now wish to identify a region R in which we shall concentrate the second set of design points. This is done as follows. We generate a random function $\eta_{(i)}(\cdot)$, and we define the 95th percentile of $\eta_{(i)}(\mathbf{X})$ to be $\nu_{(i)}$. We then estimate $\nu_{(i)}$ by using the Monte Carlo approach; a set of inputs $\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_J^*\}$ is randomly drawn from $G(\mathbf{x})$, and $\nu_{(i)}$ is estimated by the 95th percentile of $\{m_{(i)}^*(\mathbf{x}_1^*), \ldots, m_{(i)}^*(\mathbf{x}_J^*)\}$. The estimate of $\nu_{(i)}$ must be the value of $m_{(i)}^*(\mathbf{x})$ for one of the inputs $\mathbf{x}$ in the set $\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_J^*\}$. Consequently, for the ith generated function $\eta_{(i)}(\cdot)$, we use the notation $\mathbf{x}_{(i)}^*$ to denote the input in the set $\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_J^*\}$ whose expected output $m_{(i)}^*(\mathbf{x}_{(i)}^*)$ is the estimate of $\nu_{(i)}$. Now suppose that we have performed the simulation procedure K times and have obtained $\{\nu_{(1)} = m_{(1)}^*(\mathbf{x}_{(1)}^*), \ldots, \nu_{(K)} = m_{(K)}^*(\mathbf{x}_{(K)}^*)\}$. We can now think of R as being the convex hull containing $\{\mathbf{x}_{(1)}^*, \mathbf{x}_{(2)}^*, \ldots, \mathbf{x}_{(K)}^*\}$. If we denote the extra data that are obtained after the code has been run at the second set of design points by $\mathbf{y}_2$, then we should choose the second set of design points to minimize the preposterior expected value of

$$\int_R \operatorname{var}\{\eta(\mathbf{x})|\mathbf{y}_1, \mathbf{y}_2\}\,\mathrm{d}\mathbf{x}. \tag{15}$$

Given the region R, the sample of inputs $\{\mathbf{x}_{(1)}^*, \mathbf{x}_{(2)}^*, \ldots, \mathbf{x}_{(K)}^*\}$ may not necessarily occur uniformly across R. Consequently, we introduce a weight function into expression (15), so that we instead choose design points to minimize

$$\int_R \mathrm{var}\{\eta(\mathbf{x})|\mathbf{y}_1,\mathbf{y}_2\}\, w(\mathbf{x})\,\mathrm{d}\mathbf{x}, \tag{16}$$

for some function $w(\mathbf{x})$. This function is chosen by fitting a density function to the sample $\{\mathbf{x}^*_{(1)},\mathbf{x}^*_{(2)},\ldots,\mathbf{x}^*_{(K)}\}$. We do not believe that it is necessary to truncate $w(\mathbf{x})$ to lie inside R, as we cannot claim that our method precisely determines the boundary of R to begin with.

Clearly, the success of this approach in identifying a region R and choosing an appropriate weight function $w(\mathbf{x})$ will depend on the nature of the function $\eta(\cdot)$. For monotonic functions (or non-monotone functions with an overall increasing or decreasing trend), we would certainly expect to be able to do better this way than by just choosing design points to cover $\mathcal{X}$ in general. For non-monotonic functions, rather than focusing on identifying a single region R, we may still be able to exclude some regions of $\mathcal{X}$ where we are sure that $\eta(\mathbf{x})$ is above or below ν. We could then choose the second set of design points to cover the remaining regions of the sample space uniformly. This should still lead to a reduction in posterior uncertainty about ν, compared with a more general design.

We have the criterion for choosing the second set of design points given in expression (16). However, actually finding the design that will minimize this integral may itself not be straightforward. Consequently, we use a suboptimal method that is selected on the grounds of convenience. Two methods are combined; maximin Latin hypercube sampling, given in Mitchell and Morris (1995), and a maximum entropy method that was suggested by Shewry and Wynn (1987). A large number of maximin Latin hypercube samples of inputs are generated from the density function $w(\mathbf{x})$. For each sample of inputs, the determinant of the variance–covariance matrix of the outputs is computed. The sample with the largest determinant is then chosen to be the second set of design points. Maximizing this determinant has the effect of minimizing the remaining entropy of $\eta(\cdot)$. This should still lead to a design that makes expression (16) small, even if it does not give the minimum.

5.3. Choosing the simulation design points

In general, when generating a random function the simulation design points are chosen to ensure that the variance of the function given both the original data and the new simulated data is small. Again, since the purpose of generating a function here will either be to estimate the 95th percentile or to determine at which inputs the 95th percentile occurs, the variance of the function generated will not need to be small over the entire input space. The following approach can be used to remove unnecessary inputs from the simulation design points.

After running the model at the first set of design points, we can find lower and upper bounds for the 95th percentile without having to simulate any functions. We draw a large sample of J random inputs $\{\mathbf{x}^*_1,\ldots,\mathbf{x}^*_J\}$ and consider pointwise 99% intervals for $\eta(\mathbf{x}^*_i)$, for $i=1,\ldots,J$. We then find the 95th percentiles of the sets $\{m^*(\mathbf{x}^*_1) - t_{n-q,0.005}\hat{\sigma}\, c^*(\mathbf{x}^*_1),\ldots,m^*(\mathbf{x}^*_J) - t_{n-q,0.005}\hat{\sigma}\, c^*(\mathbf{x}^*_J)\}$ and $\{m^*(\mathbf{x}^*_1) + t_{n-q,0.005}\hat{\sigma}\, c^*(\mathbf{x}^*_1),\ldots,m^*(\mathbf{x}^*_J) + t_{n-q,0.005}\hat{\sigma}\, c^*(\mathbf{x}^*_J)\}$. This gives us an (overly conservative) interval which we denote by $(\nu_{\mathrm{L}},\nu_{\mathrm{U}})$ for the 95th percentile. When considering the simulation design points $\{\mathbf{x}'_1,\ldots,\mathbf{x}'_N\}$, for $i=1,\ldots,N$, we exclude $\mathbf{x}'_i$ from the design if

$$m^*(\mathbf{x}'_i) - t_{n-q,0.005}\hat{\sigma}\, c^*(\mathbf{x}'_i) > \nu_{\mathrm{U}}, \tag{17}$$

or

$$m^*(\mathbf{x}'_i) + t_{n-q,0.005}\hat{\sigma}\, c^*(\mathbf{x}'_i) < \nu_{\mathrm{L}}. \tag{18}$$

For simplicity, we first obtain a maximin Latin hypercube sample and then remove unnecessary design points by using the above procedure, to obtain suitable simulation design points.

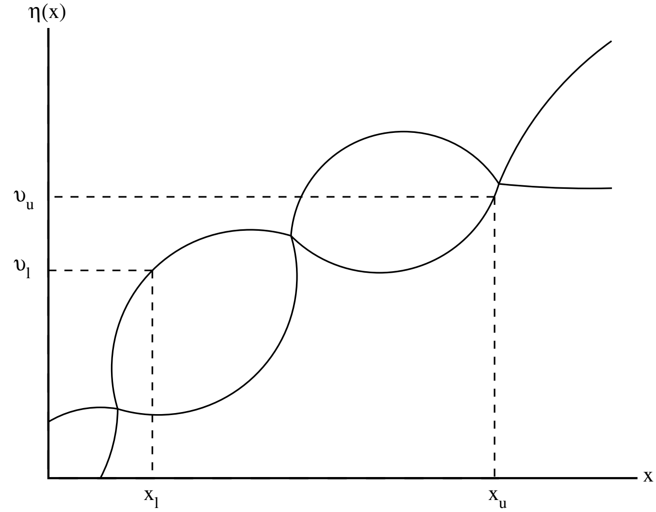

Fig. 1. Selecting design points for generating random functions: since we are confident that the 95th percentile will be between ν_{L} and ν_{U}, we should concentrate the simulation design points between x_l and x_u

To illustrate this, consider a one-dimensional function, in which we have run the code at three inputs. In Fig. 1 the full curves show pointwise 99% intervals for the function $\eta(x)$. After determining the interval $(\nu_{\mathrm{L}}, \nu_{\mathrm{U}})$, we can see that we only need to simulate the function between the inputs x_l and x_u. Once we have simulated $\eta(x)$ for $x \in (x_l, x_u)$, we can be confident that the estimate of the 95th percentile for the simulated function and the corresponding input at which the percentile occurs will not change irrespective of what values we simulate for $\eta(x)$ for $x \notin (x_l, x_u)$.

6. The method applied to the SIMPOL model

The true value of the 95th percentile is determined from a large sample of model runs and found to be 1142 units. We then consider using the Gaussian process model to obtain an efficient estimate. We set $\mathbf{h}(\mathbf{x}) = (1\ x_1\ x_2\ x_3\ x_4)^{\mathrm{T}}$ and

$$c(\mathbf{x}, \mathbf{x}') = \exp\{-(\mathbf{x} - \mathbf{x}')^{\mathrm{T}} B (\mathbf{x} - \mathbf{x}')\}, \tag{19}$$

for some diagonal matrix B. No proper prior information about the function $\eta(\cdot)$ is available to us, and so we set $p(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{-2}$. To obtain the first set of design points, we consider selecting two points in each dimension and then forming a 16-point product design to make

$$\int_{\mathcal{X}} \mathrm{var}\{\eta(\mathbf{x}) | \mathbf{y}_1\} \, \mathrm{d}G(\mathbf{x})$$

small. We denote the intital 16 observations by $\mathbf{y}_1$.

To obtain suitable simulation design points we begin with a Latin hypercube sample of size 200 and reduce this to 55 design points through rejecting unnecessary design points. We generate 1000 functions, and for each function we draw a random sample of 1000 inputs from $G(\mathbf{x})$. We obtain the sample of inputs $P = \{\mathbf{x}^*_{(1)}, \ldots, \mathbf{x}^*_{(1000)}\}$, where $m^*_{(i)}(\mathbf{x}^*_{(i)})$ is the estimate of $\nu_{(i)}$. On the basis of the initial 16 runs of the code, we estimate the 95th percentile of the output to be 1159.0, and a 95% interval for the 95th percentile is (1071,1255).

Table 1. Correlations between the inputs in R

	x_1	x_2	x_3	x_4
x_1	1	0.010	0.224	−0.079
x_2	0.010	1	−0.578	0.206
x_3	0.224	−0.578	1	0.537
x_4	−0.079	0.206	0.537	1

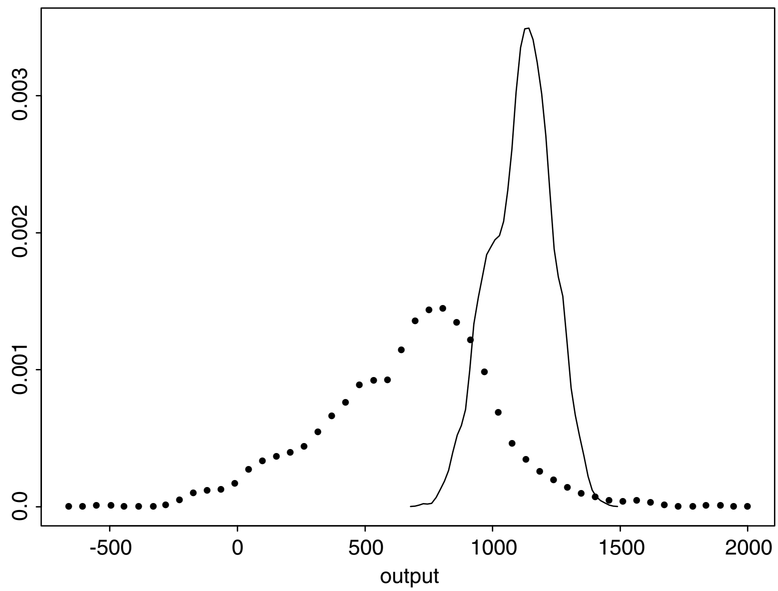

Fig. 2. Density function of $\eta(\mathbf{X})$, $(\cdots\cdots)$ and fitted density of $\eta(\mathbf{x})$ for $\mathbf{x} \in R$ (———)

In equation (16), we considered using a weight function $w(\mathbf{x})$ when minimizing the variance over the region R. To determine a suitable weight function, we fit a density function to $\mathbf{x}$ in the set P. Considering x_1, x_2, x_3 and x_4 separately, kernel density plots show that the distribution of each input is unimodal in the set R (the plots have been omitted here). Within R, there are also correlations between some of the input parameters, and these are given in Table 1. A multivariate normal distribution $N(\mathbf{a}, V)$ is fitted to $\mathbf{x}$ within the region R. This distribution is supposed to represent the region of the input space where outputs around the 95th percentile occur. Since we can obtain true outputs of the model easily, we check to see whether this is so. A large sample of inputs is drawn from $N(\mathbf{a}, V)$, and the output of the model at these inputs is evaluated. A kernel density estimate of these outputs is plotted in Fig. 2 as the full curve. Note that the modal output of this new density is very close to the true 95th percentile. The dotted curve shows the density estimate of Y, based on a large sample of outputs whose inputs are drawn from $G(\mathbf{x})$. This demonstrates that we have been successful in identifying an input region R which produces outputs that are concentrated around the 95th percentile of $Y|\eta(\cdot)$ (though this does not prove that R contains all $\mathbf{x}$-values with outputs around the 95th percentile).

After identifying R, eight new design points are chosen to cover R. These points are chosen by using the combination of maximin Latin hypercube sampling and maximum entropy, as described in Section 5.2. The eight new observations are denoted by $\mathbf{y}_2$.

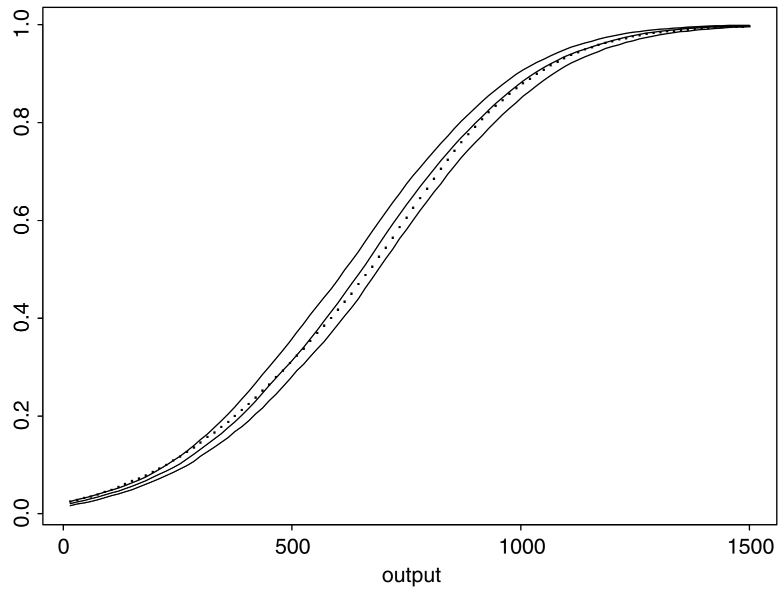

Fig. 3. Estimate of the distribution function given data $\mathbf{y}_1$ and $\mathbf{y}_2$: $\cdots\cdots$, true distribution function; ——, estimate and pointwise 95% bounds

We update the distribution of $\eta(\cdot)$ after learning the eight new outputs, and we use the simulation procedure again to obtain a final estimate of the 95th percentile. After simulating 1000 functions, we estimate the 95th percentile to be 1150.5, and a 95% interval for the 95th percentile is (1122.4,1172.1). In Fig. 3, we plot the median distribution function and a pointwise 95% interval.

We should also consider what our design selection strategy offers us over simply choosing all the design points to cover the input space, as described by G, in one go. 24 points are chosen by using the maximin Latin hypercube scheme. The estimate of the 95th percentile by using these observations is 1162. A 95% interval for the 95th percentile is (1125,1199).

Fitting the normal density to R to choose the second set of eight points after the initial 16 has given a smaller 95% interval for the 95th percentile than that obtained by a single 24-point design to learn about $\eta(\cdot)$ as a whole.

Finally, we should consider the uncertainty that we would have about ν if we were simply to use the Monte Carlo approach of generating a large sample of inputs from G and running the computer code at every input to estimate ν. Confidence intervals for Monte Carlo estimates are derived by using the procedure given in Campbell and Gardner (1988). We summarize these results in Table 2. The sample size is denoted by n. We can see that we have less uncertainty about ν from the Bayesian approach with 24 runs than we have from a Monte Carlo approach with 1000 runs.

7. Conclusions

In this paper the aim has been to obtain an accurate estimate of the 95th percentile based on a small number of runs of the code. This has been achieved, although the method that was used may not work as effectively when the input has a large number of dimensions, since the number of simulation design points that is needed can increase rapidly as the number of dimensions

Table 2. 95% intervals for the 95th percentile by using Bayesian and Monte Carlo methods

Method	*n*	*Interval*
Bayes	24	(1122.4, 1172.1)
Monte Carlo	100	(1017.0, 1259.4)
Monte Carlo	500	(1086.2, 1197.3)
Monte Carlo	1000	(1102.5, 1182.5)

increases. In the SIMPOL example, the output is monotonic with respect to each input, and this has resulted in the region R being noticeably smaller than $\mathcal{X}$. In non-monotone cases, it might not be possible to concentrate the design points in one particular region of the input space, and so we would not expect to improve as much on simply choosing design points to minimize the variance over the sample space of $\mathbf{X}$. It should still be possible to identify some regions of $\mathcal{X}$ where extra design points would not be beneficial. Finally, in various environmental applications we would expect there to be interest in extreme model outputs and we hope that this approach will be useful for other users of deterministic models.

Acknowledgements

I thank Edward Glennie and his colleagues at the WRC for the data and advice regarding the SIMPOL model. I also thank Tony O'Hagan for many valuable discussions, and two referees for their helpful comments and suggestions regarding the clarity of this paper.

References

Campbell, M. J. and Gardner, M. J. (1988) Calculating confidence intervals for some non-parametric analyses. *Br. Med. J.*, **296**, 1454–1456.
Craig, P., Goldstein, M., Seheult, A. H. and Smith, J. A. (1996) Bayes linear strategies for matching hydrocarbon reservoir history. In *Bayesian Statistics 5* (eds J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith), pp. 69–95. Oxford: Oxford University Press.
Currin, C., Mitchell, T. J., Morris, M. and Ylvisaker, D. (1991) Bayesian prediction of deterministic functions with applications to the design and analysis of computer experiments. *J. Am. Statist. Ass.*, **86**, 953–963.
Haylock, R. G. and O'Hagan, A. (1996) On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. In *Bayesian Statistics 5* (eds J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith), pp. 629–637. Oxford: Oxford University Press.
Kennedy, M. C. and O'Hagan, A. (2001) Bayesian calibration of computer models (with discussion). *J. R. Statist. Soc.* B, **63**, 425–464.
Mitchell, T. J. and Morris, M. D. (1995) Exploratory designs for computational experiments. *J. Statist. Planng Inf.*, **43**, 381–402.
Neal, R. (1999) Regression and classification using gaussian process priors. In *Bayesian Statistics 6* (eds J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith), pp. 69–95. Oxford: Oxford University Press.
Oakley, J. (2002) Eliciting Gaussian process priors for complex computer codes. *Statistician*, **51**, 81–97.
Oakley, J. E. and O'Hagan, A. (2002) Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, **89**, 769–784.
O'Hagan, A. (1994) *Kendall's Advanced Theory of Statistics*, vol. 2B, *Bayesian Inference*. London: Arnold.
O'Hagan, A., Kennedy, M. and Oakley, J. E. (1999) Uncertainty analysis and other inference tools for complex computer codes (with discussion). In *Bayesian Statistics 6* (eds J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith), pp. 503–524. Oxford: Oxford University Press.
Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989) Design and analysis of computer experiments. *Statist. Sci.*, **4**, 409–435.
Saltelli, A., Chan, K. and Scott, M. (eds) (2000) *Sensitivity Analysis*. New York: Wiley.
Shewry, M. C. and Wynn, H. P. (1987) Maximum entropy sampling. *J. Appl. Statist.*, **14**, 165–170.