Efficient Emulators for Multivariate Deterministic Functions

Jonathan Rougier*
Department of Mathematics
University of Bristol, UK

August 29, 2007

Abstract

One of the challenges with emulating the response of a multivariate function to its inputs is the quantity of data that must be assimilated, which is the product of the number of model evaluations and the number of outputs. This paper shows how even large calculations can be made tractable. It is already appreciated that gains can be made when the emulator variance function is treated as separable in the model-inputs and model-outputs. Here an additional simplification on the structure of the regressors allows very substantial further gains. The result is that it is now possible to emulate rapidly—even on an entry-level desktop computer—models with hundreds of evaluations and hundreds of outputs. This is demonstrated through calculating costs in Kflops, and in an illustration. Even larger sets of outputs are possible if they have additional structure, e.g., spatial-temporal.

Keywords: Separable variance, Kronecker product, Outer-product emulator

1 Introduction

An emulator is a statistical framework for predicting the output from a complex deterministic function, such as a computer model. From the early papers of Sacks *et al.* (1989) and Currin *et al.* (1991), the technology of emulation has rapidly matured, to the point where there is a standard approach for scalar outputs using a Gaussian process (see, e.g., Santner *et al.*, 2003). This process is conditioned on evaluations of the model at different carefully-selected inputs, and the resulting updated process can be used to

*Department of Mathematics, University Walk, Bristol BS8 1TW, U.K. Email J.C.Rougier@bristol.ac.uk.

predict the model's response at any new set of inputs. In statistical inference, the emulator takes the place of the model, and account is taken of the uncertainty in the prediction. This uncertainty, which would go to zero if the number of evaluations were increased without limit, has been termed 'code uncertainty' by O'Hagan (2006).

Recently, attention has turned to the construction of multivariate emulators, i.e., emulators for functions with vector outputs. This is reviewed in section 2. Assimilating the data into an emulator—scalar or multivariate—is described in section 3, which uses a conjugate distribution to simplify the updating process. The main challenge with multivariate emulators is the quantity of data that must be assimilated, which is the product of the number of evaluations and the number of outputs. Section 4 describes a standard approach for simplifying the update in this case, which is to impose separability in the variance function. Such an emulator might be termed a 'separable emulator'.

This simplification only goes so far, though. Section 5 presents a further new simplification, which has a dramatic effect on the calculations. It restricts the choice of regressors in such a way that the resulting factorisation of the regression matrix is conformable with the factorisation of the residual variance matrix that follows from separability. This allows the updating equations to be simplified symbolically, to the point where the actual computations happen on much smaller objects. Paradoxically, this new restriction actually allows for a much more general treatment of the emulator, because without it the regression matrix typically has to be treated very crudely. Such an emulator is termed an 'outer product emulator'.

The power of the two simplifications is illustrated in section 6, where the time in seconds to construct an emulator is presented for various sizes for the model and choices for the emulator. The separable emulator is shown to be much more efficient than the basic emulator, but the outer-product emulator is an order of magnitude better still, both in the time for construction and the sizes of model that can be emulated. These practical results complement the theoretical analysis in terms of floating point operations in sections 4 and 5. Section 7 concludes, indicating a further extension suitable to models with large numbers of outputs, e.g., spatial-temporal.

This paper is a companion to Rougier *et al.* (2007), where an outer-product emulator is used to model an atmospheric model, incorporating a large amount of expert judgement about the model's behaviour.

2 Multivariate emulators

The concepts in this review, such as the distinction between model-inputs and the index of model-outputs, will be clarified further in section 3.

Broadly, there are five treatments for emulators of with multivariate out-

puts. The first four of these are probabilistic emulators based on Gaussian processes, typically subject to further constraints for tractability. The first (A) is simply to model each output component independently with a scalar emulator. This is unpopular, because it imposes the unattractive condition that the output components are conditionally independent given the model-inputs. Where the outputs are related, it is likely that deficiencies in the emulator will lead to systematic errors which ought to be represented as non-zero—typically positive—covariances.

The second treatment (B) is to model linear combinations of the output components with independent scalar emulators. The advantage of this approach is that linear transformations can make the independence of the emulators much more palatable. In Climate Science, where the number of output components can be huge, a method similar to Principal Components is used to select the linear combinations of the outputs. A better idea, suggested by Michael Goldstein, might be to use Canonical Correlation Analysis on both inputs and outputs, to select the linear combinations of the outputs. This has the advantage that the combinations are chosen in terms of their 'explainability' by the inputs, rather than simply their variation in the outputs. One disadvantage of this approach is that it double-counts the data (the evaluations of the model): once to select the linear combinations, and again to construct the emulator. Another is that it can be tricky to recover the underlying components, i.e., to undo the transformation. Sometimes this is not a problem, though. For example, if the emulator output is to be compared with observational data (e.g., for model calibration) then the observational data can also be linearly-transformed.

The third treatment (C) is to include the index of the model-outputs among the inputs to the emulator, but otherwise to operate within the scalar emulator framework used in treatments (A) and (B). This has been discussed in Kennedy and O'Hagan (2001), Bayarri *et al.* (2005), and Conti and O'Hagan (2007). It is focus of this paper, and has been implemented in Rougier *et al.* (2007). In a simple treatment, the model's response is represented as a linear combination of basis functions in the index of the model-outputs, where the coefficients are uncertain functions of the model-inputs. The crucial restriction is a common variance multiplier across the model-outputs. This makes this type of emulator suitable for model-outputs that are all of one type, for example a large collection of temperatures indexed by space and time. While it may also work over different output types, this would probably require some kind of pre-normalisation.

The crucial feature of the fourth treatment (D) is that it relaxes the common variance multiplier restriction, so that the emulator can operate over several different types of output without pre-normalisation. There are two different approaches here. The first approach (D1) is an extension of (C). Rather than being treated jointly and sharing a common variance, the coefficients on the basis functions are treated independently, and allowed to have

different variances, which is more flexible but less tractable. This approach has been implemented by Higdon *et al.* (2007) and Bayarri *et al.* (2007). It has the potential to handle different types, because a type-indicator in the index of model-outputs could selectively activate basis functions. However, both Higdon *et al.* and Bayarri *et al.* modelled a single-type output, so perhaps the simpler methods outlined in this paper would have been effective.

The second approach (D2) treats the model output as a general multivariate collection, subject to restrictions for tractability, suggested and implemented by both Conti and O'Hagan (2007) and Rougier (2007). The indexing of the model-outputs enters only indirectly, in the specification of the prior variance matrix of the outputs (so, that, for example, two outputs of the same type that were spatially close could have a positive prior covariance). Thus the disadvantage of this approach is that additional structure in the outputs, such as spatial proximity, has to be recovered empirically: it cannot be a pre-specified feature of the emulator, as it would be if the index of the model-outputs was used in basis functions. There are also some quite severe restrictions on the joint covariance structure that are imposed for tractability, as discussed in Rougier (2007). The advantage, though, is an efficient treatment of multiple output-types, without pre-normalisation.

Finally, the fifth approach (E) is a generalisation of (D2), and also chronologically the first treatment of multivariate outputs. This is the *Bayes linear* approach, exemplified by Craig *et al.* (1996, 1997), Craig *et al.* (2001),Goldstein and Rougier (2004, 2006, 2007). The collection of outputs is described by a mean function and variance function, both parameterised by the model-inputs. This is flexible, but demanding in computational time, as it has none of the simplifying structure that follows from probabilistic conjugacy, and also demanding in elicitation, as prior variances must be specified.

3 Conjugate emulator

3.1 Model structure

Consider a deterministic computer model with model-input $r \in \mathcal{R} \subset \mathbb{R}^p$, producing a set of outputs of the same type. This paper is concerned with models for which the set of outputs is finite and pre-specified, and does not depend on r. Typically this is either a feature of the model itself, or it can be superimposed on the model's raw outputs, e.g., by interpolating them onto a fixed set of knots in the domain of the model-outputs. In this case the model can be represented as

$$f(r) \triangleq \left\{ f_j(r) \right\}_{j=1}^{q} \tag{1}$$

where q is the number of outputs; braces with sub- and super-scripts are used throughout the paper to denote ordered tuples. Here '$\triangleq$' denotes 'defined as', and, below, '$\equiv$' denotes 'equivalent by definition'. Typically, the model-output index j maps to a tuple s_j, which denotes the coordinate of the jth value in the domain of the model-outputs; this domain is denoted $\mathcal{S}$.

Computer models with a fixed set of outputs will be referred to as having *regular* outputs. It is perfectly standard for computer models to have regular outputs, but the point is emphasised here because it is a cornerstone of the techniques in this paper.

For illustration, the computer model might be a climate model, evaluated to equilibrium for fixed forcing. The model-input r might represent the value of the model-parameters (e.g., those coefficients in the model that account, approximately, for the effect of sub-grid-scale processes), and j might index the ocean-cells in the finite difference scheme used to integrate the differential equations in the model. Then $f_j(r)$ might be a scalar quantity such as salinity. Each j would be associated with a triple of latitude, longitude, and depth, corresponding to the location of the mid-point of its cell, and this would be denoted s_j. In a more advanced solver, the raw output for input r might depend on r; this raw output must then be post-processed in order that the model can be treated as having regular outputs.

3.2 The emulator

Computer models can only be evaluated a finite number of times: often, for a large model like a climate model, only a small number of times. Consequently we are uncertain about the model output at an arbitrary choice of input variable $r \in \mathcal{R}$, what O'Hagan (2006) describes as 'code uncertainty'. An *emulator* predicts the model-output at any r, based on an ensemble of evaluations; in inference, the emulator then takes the place of the model. As the model has regular outputs the ensemble can be represented by the $n \times p$ design matrix R and $n \times q$ output matrix Y:

$$Y \triangleq \left\{ f_j(R_i) \right\}_{i=1,j=1}^{n,q}, \quad \text{or} \quad Y = f(R) \tag{2}$$

where R_i is the ith row of R, and contains the model-inputs for the ith evaluation of the model.

The distinctive feature of a statistical emulator is that its prediction is probabilistic. Thus the emulator is represented by the distribution $\pi\big(f(R') \mid Y\big)$, for any finite collection of model-inputs R', where the design matrix R is implicit in the joint distribution of $f(R')$ and Y. A probabilistic approach has the desirable feature that the updated emulator interpolates the ensemble, i.e. $\mathrm{E}\big(f(R_i) \mid Y\big) = Y_i$ and $\mathrm{Var}\big(f(R_i) \mid Y\big) = \mathbf{0}_{q\times q}$, for all $i = 1, \ldots, n$.

We write our emulator for the computer model in the general form

$$f_j(r) = \sum_{k=1}^{v} \beta_k \, g_k(r, s_j) + e(r, s_j) \quad j = 1, \ldots, q, \tag{3}$$

where $g \triangleq \{g_k\}_{k=1}^{v}$ are specified regressors defined on $\mathcal{R} \times \mathcal{S}$, $\beta \triangleq \{\beta_k\}_{k=1}^{v}$ are uncertain regression coefficients, and the stochastic process e is an uncertain residual defined on $\mathcal{R} \times \mathcal{S}$. For tractability it is important to have a common set of regressors for all outputs. This is another reason why the emulator in this paper is best-suited to a common output type (see section 2). If we specify a distribution for $[\beta, e(\cdot)]$, where $e(\cdot)$ denotes the residual from any finite collection of model-inputs, then (3) induces a distribution over any finite collection of model evaluations. This allows us to predict the model output at design matrix R', given the evaluations at design matrix R. For tractability, we choose a Normal Inverse Gamma distribution for $[\beta, e(\cdot)]$, introducing a common variance multiplier τ:

$$\beta \perp\!\!\!\perp e(\cdot) \mid \tau \tag{4a}$$

$$\beta \mid \tau \sim \mathrm{N}_v\big(m, \tau V\big) \tag{4b}$$

$$e(\cdot) \mid \tau \sim \mathrm{GP}\big(0, \tau\kappa(\cdot, \cdot)\big) \tag{4c}$$

$$\tau \sim \mathrm{IG}(a, d). \tag{4d}$$

Here N_v denotes a multivariate Gaussian distribution with specified mean and variance, GP denotes a Gaussian process with specified mean function and variance function, and IG denotes an Inverse Gamma distribution. The variance function $\kappa(\cdot, \cdot)$ in the Gaussian process for e is defined on $[\mathcal{R} \times \mathcal{S}] \times [\mathcal{R} \times \mathcal{S}]$.

Eq. (4) induces a Normal Inverse Gamma (NIG) distribution over any finite collection of model evaluations augmented by the multiplier τ. The convenient parameterisation of the NIG, and the predictive and updated distributions, are summarised in the Appendix. Hence our emulator for f is specified by a choice of regressors g and then, based on these, the Normal Inverse Gamma (NIG) hyperparameters $\big\{m, V, a, d\big\}$ and the residual variance function $\kappa(\cdot, \cdot)$.

3.3 Predicting and updating

The task is to predict $f(R')$ $(n' \times q)$ for any design matrix R' $(n' \times p)$, based on the evaluations Y $(n \times q)$. It will be convenient to represent both $f(R)$ and $f(R')$ as vectors. To this effect, consider an index $\ell = 1, \ldots, nq$. Define

$$i(\ell) \triangleq 1 + (\ell - 1) \bmod n \qquad j(\ell) \triangleq 1 + (\ell - 1) \operatorname{div} n, \tag{5}$$

from which

$$y \triangleq \operatorname{vec} Y \equiv \left\{Y_{i(\ell)j(\ell)}\right\}_{\ell=1}^{nq}, \tag{6}$$

where the operator 'vec' stacks the columns of a matrix into a vector, from left to right (i.e., Y is in column-major order for y). Define the $nq \times v$ regression matrix $G \triangleq G(R)$, where in general

$$G(R) \triangleq \left\{g_k(R_{i(\ell)}, s_{j(\ell)})\right\}_{\ell=1,k=1}^{nq,v} \tag{7}$$

and define the vector of residuals $e \triangleq e(R)$, where in general

$$e(R) \triangleq \left\{e(R_{i(\ell)}, s_{j(\ell)})\right\}_{\ell=1}^{nq}. \tag{8}$$

Now the ensemble can be written as a vector,

$$y = G\beta + e. \tag{9}$$

For predicting at design matrix R', we would write $y' = G'\beta + e'$.

According to our emulator, $f(R')$ is a known linear combination of β and e'. According to our joint distribution, $f(R') \mid \tau$ is Gaussian, and, integrating out τ, $f(R')$ is multivariate Student-t. The structure of this prediction is preserved after updating by Y, which simply updates the NIG hyperparameters (see the Appendix for the precise form).

The first part of the update is to compute the mean and variance of the Gaussian vector $\left\{\beta, e'\right\} \mid \left\{\tau, y\right\}$. The crucial object, in terms of computational expense, is the $(nq \times nq)$ variance matrix $\operatorname{Var}\left(y \mid \tau\right) \equiv \tau S$, where

$$S \triangleq GVG^T + W \tag{10a}$$

and $W \triangleq W(R, R)$, where in general

$$W(R, R') \triangleq \left\{\kappa\left([R_{i(\ell)}, s_{j(\ell)}], [R'_{i(\ell')}, s_{j(\ell')}]\right)\right\}_{\ell=1,\ell'=1}^{nq,n'q} \tag{10b}$$

for any two design matrices R and R'. The updating equations for $\left\{\beta, e'\right\}$ are then

$$\operatorname{E}\left(\beta \mid \tau, y\right) = m + VG^T S^{-1}(y - Gm) \tag{11a}$$

$$\operatorname{Var}\left(\beta \mid \tau, y\right) = \tau\left\{V - VG^T S^{-1}(VG^T)^T\right\} \tag{11b}$$

$$\operatorname{E}\left(e' \mid \tau, y\right) = (W')^T S^{-1}(y - Gm) \tag{11c}$$

$$\operatorname{Var}\left(e' \mid \tau, y\right) = \tau\left\{W'' - (W')^T S^{-1} W'\right\} \tag{11d}$$

$$\operatorname{Cov}\left(\beta, e' \mid \tau, y\right) = -\tau\, VG^T S^{-1} W', \tag{11e}$$

where

$$W' \triangleq W(R, R') \quad \text{and} \quad W'' \triangleq W(R', R'). \tag{11f}$$

For the variance multiplier,

$$\tau \mid y \sim \mathrm{IG}(a + nq, d + \mathrm{mhd}) \tag{12a}$$

where 'mhd' is the Mahalanobis distance

$$\mathrm{mhd} \triangleq (y - Gm)^T S^{-1} (y - Gm). \tag{12b}$$

4 Separable residual variance function

So far, our simplifications in the joint distribution have been restricted to the conditional independence of β and $e(\cdot)$, and the NIG distribution for $[\beta, e(\cdot), \tau]$. In this section and the next we make further simplifications to the joint distribution, concerning the form of the variance function $\kappa(\cdot, \cdot)$, in this section, and the regressors g, in section 5.

4.1 The Sherman-Morrison-Woodbury reformulation

The expensive part of the calculations in (11) is the inversion of S, which is best performed by finding the Cholesky decomposition of S, an operation requiring a single expense of $(nq)^3/3$ flops (Golub and Van Loan, 1996, p. 144).

The expression for S^{-1} can be represented differently, using the Sherman-Morrison-Woodbury (SMW) formula, which implies

$$(A^{-1} + BC^{-1}B^T)^{-1} = A - AB(C + B^T AB)^{-1}(AB)^T \tag{13}$$

for appropriately conformable and non-singular matrices; see, e.g., Golub and Van Loan (1996, p. 50), or Brookes (2005). Using the mapping $A \to W^{-1}$, $B \to G$, $C \to V^{-1}$,

$$S^{-1} = W^{-1} - (W^{-1}G)(V^{-1} + G^T W^{-1} G)^{-1}(W^{-1}G)^T \tag{14}$$

which has the advantage that the $nq \times nq$ matrix inverse is of W, not S. If the model has regular outputs we can arrange, if we so choose, for W to have special structure which makes it easier to invert, as discussed in section 4.2. Note that two further matrix inversions are required to compute S^{-1}, but both of these are of $v \times v$ matrices, where v is the number of regressors, and are therefore unlikely to be large enough to be troubling. Further, we might well specify V as diagonal, e.g., if the regressors are orthogonal, in which case its inversion is trivial.

As a second application of the SMW formula, the updated variance for β can be written

$$\mathrm{Var}\big(\beta \mid \tau, y\big) = \tau D^{-1} \quad \text{where} \quad D \triangleq V^{-1} + G^T W^{-1} G \tag{15}$$

using the mapping $A \to V$, $B \to G^T$, $C \to W$; the $(v \times v)$ variance matrix D can be used to simplify S^{-1}:

$$S^{-1} = W^{-1} - (W^{-1}G)D^{-1}(W^{-1}G)^T. \qquad (14')$$

4.2 Separability

Specifying a joint covariance structure over $\mathcal{R} \times \mathcal{S}$ can be demanding, and it is natural to simplify this task by treating κ as separable in r and s, so that

$$\kappa\big([r, s_j], [r', s_{j'}]\big) = \kappa^r(r, r')\,\kappa^s(s_j, s_{j'}). \qquad (16)$$

Note that separability in the residual does not imply separability in the emulator. This is somewhat reassuring, given that separability is a very strong constraint on our judgements about the model (see, e.g., the characterisation in O'Hagan, 1998). However, the type D2 emulators in section 2 *are* separable, as discussed in Rougier (2007).

In the case of a model with regular outputs, separability of κ gives rise to a Kronecker product factorisation of W:

$$W = W^s \otimes W^r \text{ where } \begin{cases} W^r \triangleq \big\{\kappa^r(R_i, R_{i'})\big\}_{i=1,i'=1}^{n,n} \\ W^s \triangleq \big\{\kappa^s(s_j, s_{j'})\big\}_{j=1,j'=1}^{q,q} \end{cases} \qquad (17)$$

and '$\otimes$' denotes the Kronecker product. Using the result that $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$, the dominant cost of the inversion of W in this case is not $(nq)^3/3$ flops but $n^3/3 + q^3/3$ flops. There are also Kronecker factorisations for W' and W'', defined in (11f), which will be introduced below. The properties of the Kronecker product can be found in, e.g., Mardia *et al.* (1979, Appendix A.2.5) or Brookes (2005).

A similar approach has been proposed when the outputs are not regular or κ is not separable, which is to replace the original W matrix with an approximate Kronecker product decomposition (Genton, 2007).

There is a second benefit to having a Kronecker structure for W, which is that sparsity in either W^r or W^s will be preserved in W. This sparsity can be exploited both to reduce the size of W (as it is represented in the computation) and, where there is banding, to speed up the crucial operations of finding the Cholesky decomposition and backsolving. Wendland (1995) describes a family of covariance functions which are sparse. Additional sparsity can be imposed, approximately, by tapering the variance function of one or both of κ^r and κ^s (Furrer *et al.*, 2006).

4.3 Computation

There are two ways the SMW formula might be used, but only one is recommended. The wrong way is to use the SMW formula to compute S^{-1},

and then plug the value for S^{-1} into (11), the original updating equations. One problem with this approach is that the results are numerically unstable: when evaluating an expression such as $A^T S^{-1} A$ it is better to decompose S as $Q^T Q$ and then evaluate the inner product of $Q^{-T} A$. On the other hand, if we take the *expression* for S^{-1} and plug that into (11), then the central variance matrix is D^{-1}, and this can be decomposed easily, because it is $v \times v$.

The second and more important reason is the simple plug-in approach ignores the additional benefits of separability in the residual variance. Starting from (11), (12b), (14′), and (15), the expressions that must be computed in order to predict $f(R')$ are:

$$c \triangleq y - Gm \tag{18a}$$

$$D \triangleq V^{-1} + (G^T W^{-1} G) \tag{18b}$$

$$\mathrm{E}\left(\beta \mid \tau, y\right) = m + V\{(G^T W^{-1} c) - (G^T W^{-1} G) D^{-1} (G^T W^{-1} c)\} \tag{18c}$$

$$\mathrm{Var}\left(\beta \mid \tau, y\right) = \tau D^{-1} \tag{18d}$$

$$\mathrm{E}\left(e' \mid \tau, y\right) = (W')^T W^{-1} c - (G^T W^{-1} W')^T D^{-1} (G^T W^{-1} c) \tag{18e}$$

$$\begin{aligned}\mathrm{Var}\left(e' \mid \tau, y\right) = \tau\Big\{ & W'' - (W')^T W^{-1} W' \\ & + (G^T W^{-1} W')^T D^{-1} (G^T W^{-1} W')\Big\}\end{aligned} \tag{18f}$$

$$\mathrm{Cov}\left(\beta, e' \mid \tau, y\right) = -\tau V\left\{G^T W^{-1} W' - (G^T W^{-1} G) D^{-1} (G^T W^{-1} W')\right\} \tag{18g}$$

$$\mathrm{mhd} = c^T W^{-1} c - (G^T W^{-1} c)^T D^{-1} (G^T W^{-1} c) \tag{18h}$$

(with some re-arrangement to highlight the compound terms). The terms just involving Ws can be simplified symbolically, leading to an easier evaluation. In particular, define

$$W^{r\prime} \triangleq \left\{\kappa^r(R_i, R'_{i'})\right\}_{i=1,i'=1}^{n,n'} \tag{19a}$$

then

$$W^{-1} = W^{-s} \otimes W^{-r} \tag{19b}$$

$$W' = W^s \otimes W^{r\prime} \tag{19c}$$

$$W^{-1} W' = (W^{-s} \otimes W^{-r})(W^s \otimes W^{r\prime}) = I_q \otimes W^{-r} W^{r\prime} \tag{19d}$$

$$(W')^T W^{-1} W' = W^s \otimes (W^{r\prime})^T W^{-r} W^{r\prime}, \tag{19e}$$

where $W^{-r} \triangleq (W^r)^{-1}$ and likewise for W^{-s}.

As an illustration of the benefit of the symbolic step, consider the cost, in floating point operations (flops), of computing $(W')^T W^{-1} W'$, assuming we already have all three terms in (19b). Computed directly, i.e. according

to the lefthand side of (19e), this would be $(qq)(nn')$ flops to construct W', plus $2(n'q)(nq)(nq)$ and $2(n'q)(nq)(n'q)$ flops for the two matrix products, evaluating from left to right. Computed after the symbolic step, i.e. according to the righthand side of (19e), this would be $2n'nn + 2n'nn'$ flops for the two matrix products, and $(qq)(n'n')$ for the Kronecker product. To illustrate these costs, consider the outer-product emulator built in Rougier *et al.* (2007). This is modest in size: $n = 30$ and $q = 10$. Suppose we are just interested in predicting a single evaluation, so $n' = 1$. In this case the lefthand side costs 1863 kilo-flops (Kflops), and the righthand side costs 2.0 Kflops. The basic guideline for efficient computing is to leave the Kronecker products as late as possible, or even avoid them completely.

Both of these calculations have the unfortunate side-effect that the result is not necessarily symmetric. This is avoidable. The Cholesky decomposition of W^r, Q^r say, will have been computed prior to finding W^{-r}. In this case

$$(W^{r\prime})^T W^{-r} W^{r\prime} = \{(Q^r)^{-T} W^{r\prime}\}^T \{(Q^r)^{-T} W^{r\prime}\} \tag{20}$$

This calculation costs $(n^2/2)n'$ flops for the back-substitution, and approximately $n'nn'$ for the inner-product (taking advantage of symmetry). In this case the total cost for the Rougier *et al.* emulator is 0.5 Kflops: this calculation is both numerically better and faster than the direct calculation.

In the expression $c^T W^{-1} c$, separability allows the Kronecker product to be avoided completely. Reshape C as a $n \times q$ matrix, so that $c \equiv \operatorname{vec} C$. Then

$$W^{-1} c = (W^{-s} \otimes W^{-r}) \operatorname{vec} C = \operatorname{vec}(W^{-r} C W^{-s}) \tag{21}$$

using the general relation $\operatorname{vec}(ABC) = (C^T \otimes A) \operatorname{vec} B$. Thus

$$c^T W^{-1} c = \operatorname{sum}\{C * (W^{-r} C W^{-s})\} \tag{22}$$

where '$*$' denotes the Hadamard (i.e., component-wise) product, and 'sum' sums the components of a matrix. The lefthand side of (22) costs $2(nq)^2 + 2(nq)$ flops, assuming W^{-1} has already been computed; the righthand side, $2nq + 2nnq + 2nqq$ flops, assuming W^{-r} and W^{-s} have already been computed. For the Rougier *et al.* emulator these two costs are 181 Kflops and 25 Kflops, respectively.

5 Outer-product emulators

In this section we restrict our joint distribution further, with a particular structure for the regressors.

5.1 Factorising the regression matrix

There is a natural approach to constructing a multivariate emulator. At any given r, we think of the model output as being approximately (ignoring the residual) an uncertain linear combination of specified basis functions in s:

$$f(r,s) \approx \sum_{k'=1}^{v_s} \alpha_{k'}(r)\, g^s_{k'}(s). \tag{23}$$

Then we think of the coefficients as being themselves uncertain linear combinations of specified basis functions in r:

$$\alpha_{k'}(r) \approx \sum_{k''=1}^{v_r} \beta_{k'k''}\, g^r_{k''}(r) \quad k' = 1, \ldots, v_s, \tag{24}$$

using the same regressors for each coefficient. Combining these gives

$$f(r,s) \approx \sum_{k'=1}^{v_s} \sum_{k''=1}^{v_r} \beta_{k'k''}\, g^r_{k''}(r)\, g^s_{k'}(s) \equiv \sum_{k=1}^{v} \beta_k\, g_k(r,s) \tag{25}$$

where $v = v_s v_r$ and the set of regressors $\left\{g_k\right\}_{k=1}^{v}$ is constructed by taking the outer-product of a set of regressors in r and a set of regressors in s.

Therefore, starting from the two sets of regressors

$$g^r = \left\{g^r_k\right\}_{k=1}^{v_r} \quad \text{and} \quad g^s = \left\{g^s_k\right\}_{k=1}^{v_s}, \tag{26}$$

construct two regression matrices

$$G^r \triangleq \left\{g^r_k(R_i)\right\}_{i=1,k=1}^{n,v_r} \quad \text{and} \quad G^s \triangleq \left\{g^s_k(s_j)\right\}_{j=1,k=1}^{q,v_s}. \tag{27}$$

Note that G^r is a function of the design matrix R (so that $G^{r\prime}$ is the corresponding matrix for R'), but G^s is invariant to R. In an emulator such as (25), the regression matrix G from (7) can be factorised as

$$G = G^s \otimes G^r. \tag{28}$$

As will be shown below, this factorisation is highly beneficial when combined with rectangular outputs and a separable variance function. For this reason I suggest that the term *outer-product emulator* be restricted to situations where all three of these features are present.

5.2 Computation

In an outer-product emulator, the Kronecker factorisations of G and W are conformable. This means there are symbolic simplifications for the terms combining G and W. Once again, this allows the evaluation of Kronecker

products to be delayed or avoided, making the calculations much more efficient.

Consider, to start with, the term

$$G^T W^{-1} = (G^s \otimes G^r)^T (W^{-s} \otimes W^{-r}) \equiv H^s \otimes H^r \tag{29}$$

where $H^s \triangleq (G^s)^T W^{-s}$, and similarly for H^r. Then the three terms involving G and W are:

$$G^T W^{-1} G = H^s G^s \otimes H^r G^r \tag{30a}$$

$$G^T W^{-1} W' = (G^s \otimes G^r)^T (I_q \otimes W^{-r} W^{r\prime}) = (G^s)^T \otimes H^r W^{r\prime} \tag{30b}$$

$$G^T W^{-1} c = \mathrm{vec}\{H^r C (H^s)^T\}. \tag{30c}$$

Note that in (30a), $H^s G^s \equiv (G^s)^T W^{-s} G^s$, and similarly for $H^r G^r$. Starting from the Cholesky decomposition of W^s, Q^s say, it is better to compute (30a) as

$$H^s G^s = \{(Q^s)^{-T} G^s\}^T \{(Q^s)^{-T} G^s\}\}, \tag{30a$'$}$$

and likewise for $H^r G^r$.

As an illustration of the further benefits the come from an outer-product emulator, consider the cost of computing $G^T W^{-1} G$ directly, with the cost after symbolic simplification, as given in (30a$'$). The direct calculation costs $2v(nq)(nq) + 2v(nq)v$ flops, and is not certain to be symmetric. After the symbolic step, the cost is $(q^2/2)v_s + v_s q v_s + (n^2/2)v_r + v_r n v_r + (v_s)^2 (v_r)^2$ flops, and this *is* certain to be symmetric. For the Rougier *et al.* (2007) emulator, $v_r = v_s = 6$, so $v = 36$. The costs are 7258 Kflops and 6 Kflops, respectively. In fact, the cost of the lefthand side is higher than this, because the matrix G needs to be computed explicitly from G^r and G^s in this case, adding $(qv_r)(nv_s)$ flops; 11 Kflops in the example.

6 Illustration

The purpose of this illustration is to demonstrate the efficiency of the outer-product emulator over a range of values for n, the number of evaluations in the ensemble, q, the number of model-outputs, and v_r and v_s, the number of input- and output-regressors. This efficiency takes two forms. First, the outer-product emulator should be much faster, as already established from the calculation of costs in flops. Second, the outer-product emulator constructs much smaller objects, and it avoids the calculation of G altogether. Thus it should continue to function where other emulators cannot allocate enough memory.

The object is to predict ten further evaluations (i.e., $n' = 10$). Three calculations are considered. First, the direct inversion of S, following the calculations in (11) in section 3.3 (method S). Second, the construction of

S^{-1} using the SMW formula and the additional simplifications that arise from a separable variance function, as described in section 4.3 (method W). Third, the additional simplifications that arise in an outer-product emulator, as described in section 5.2 (method G). In all cases variances are computed as inner-products, following a Cholesky decomposition and back-substitution, as described at the beginning of section 4.3.

The results are given in Table 1. The calculations are implemented in the Statistical Computing Environment R (R Development Core Team, 2004). The R-code for this illustration, from which a fully-functioning outer-product emulator can easily be extracted, is available from the author. In R, the function `kronecker` is implemented in high-level code rather than, say, C or Fortran. Therefore a further substantial increase in speed would be available if this function were rewritten. The timings are for a 2007 MacBook Pro laptop (2.33 GHz Intel Core 2 Duo with 2 GB memory).

The performance of the outer-product emulator is really astounding. It computes in under two seconds a prediction that neither of the other two approaches could compute at all. It computes in under a second a prediction that on the separable emulator (method W) took over two minutes. For almost all of the combinations of parameters in the Table, it is effectively instantaneous. A much larger prediction than any of the ones in the Table ($n = 1000$, $q = 100$, $v_r = v_s = 20$) took under a minute.

7 Discussion

An outer-product emulator has three features. First, the underlying model has regular outputs, e.g., outputs that can be represented as a matrix in which the rows are model evaluations and the columns are output-components. This is standard for many models, or can be imposed by post-processing the raw model-output. Second, the residual variance function in the emulator can be represented as the product of a function for the model-input values and a function for the model-output coordinates. This separability is a very natural choice for constructing the joint variance function, and is a weaker choice than is typically made in scalar emulator construction, where the variance function for the model-input values itself is taken to be the product of a function for each input-component. Third—and this is the new feature—the regressors are constructed as the outer-product of a set of regressors in the model-inputs and a set of regressors in the model-outputs.

The outer-product emulator offers the opportunity to go an order of magnitude larger in emulating multivariate functions, both in terms of the number of evaluations in the ensemble, and the number of outputs. Alternatively, it offers the opportunity to build many emulators where previously there was only time for a few. This makes it easier to generate predictive diagnostics for emulators, as in Rougier *et al.* (2007), or to embed an outer-

Table 1: Timings for three different methods of updating the emulator, for different values of n (number of evaluations), q (number of outputs), and v_r and v_s (number of regressors). Method S, direct inversion of S (section 3.3); Method W, SMW inversion of S and separable residual variance function (section 4.3); Method G, outer-product emulator (section 5.2). The times are total elapsed time in seconds. Where the time is ∞, there was not enough memory to complete the calculation. The percentages for the W and G methods are relative to S, for given n, q, and v_r and v_s.

	Method	$q = 10$		$q = 30$		$q = 100$	
				$\boldsymbol{v_r = 7, v_s = 4}$			
	S	0.040		0.290		4.170	
$n = 30$	W	0.021	*52%*	0.110	*38%*	1.093	*26%*
	G	0.013	*33%*	0.045	*16%*	0.420	*10%*
	S	0.230		1.140		19.012	
$n = 60$	W	0.063	*27%*	0.300	*26%*	3.062	*16%*
	G	0.016	*7%*	0.063	*6%*	0.588	*3%*
	S	0.475		3.774		∞	
$n = 110$	W	0.132	*28%*	0.834	*22%*	197.320	
	G	0.024	*5%*	0.094	*2%*	1.164	
	S	1.224		15.024		∞	
$n = 200$	W	0.387	*32%*	2.386	*16%*	∞	
	G	0.054	*4%*	0.159	*1%*	1.329	
				$\boldsymbol{v_r = 15, v_s = 8}$			
	S	0.056		0.335		4.447	
$n = 30$	W	0.046	*82%*	0.151	*45%*	1.368	*31%*
	G	0.032	*57%*	0.071	*21%*	0.488	*11%*
	S	0.146		1.105		19.944	
$n = 60$	W	0.072	*49%*	0.416	*38%*	4.808	*24%*
	G	0.423	*290%*	0.088	*8%*	0.658	*3%*
	S	0.385		3.929		∞	
$n = 110$	W	0.371	*96%*	1.061	*27%*	159.533	
	G	0.049	*13%*	0.117	*3%*	0.896	
	S	1.187		15.663		∞	
$n = 200$	W	0.398	*34%*	4.244	*27%*	∞	
	G	0.087	*7%*	0.187	*1%*	1.376	

product emulator within a hierarchical statistical model, e.g. to estimate or to mix over the emulator hyperparameters.

The outer-product emulator also points the way to the natural generalisation of current practice, in which regressors to play a much more active role in the emulator. This is valuable for the emulator's predictive performance when the model input space is largely an extrapolation from the convex hull of the ensemble of evaluations. Typically this would be when the model is very expensive to evaluate, or the input-space is high-dimensional: many environmental models, e.g., climate models, have exactly these characteristics. A large number of regressors are possible because the regression matrix never has to be explicitly evaluated: it is collapsed, symbolically, into smaller objects.

The approach used in the outer-product emulator can be taken further, for applications where there are a very high number of model-outputs for each evaluation. Typically this would arise where the domain of the model-outputs was spatial-temporal. In this case it would be natural to specify the variance function for the model-output coordinates to be separable in space and time. If the model-outputs were regular in space and time and the output-regressors were constructed from the outer-product of space regressors and time regressors, then exactly the same Kronecker product approach could be used to factor the output regression matrix G^s and the output variance matrix W^s. In this way many thousands of outputs per evaluation could be handled in a multivariate emulator, if, say, they comprised five hundred spatial locations at each of one hundred times. This is a typical configuration for modelling the impact of atmospheric greenhouse gas emissions on C21st climate. Higdon *et al.* (2007) provide another example: 36 evaluations where each evaluation produces a 20×26 matrix of radii indexed by time and angle. This could be computed as a $n = 36$ and $q = 520$ problem, but the output could also be treated as separable in time and angle, in which case the construction of an emulator is simple and cheap, permitting the inclusion of many more evaluations (or more experiments, in this case), or a higher resolution for the outputs.

Acknowledgements

This paper was conceived while I was a Duke University Fellow as part of the SAMSI programme 'Development, Assessment and Utilization of Complex Computer Models', and a visitor to the IMAGe group at NCAR. I would like to thank Stephan Sain at NCAR for the initial insight that the regression matrix could have a Kronecker product factorisation, and Patrick and Mark at Organica Café in Boulder CO, where the paper was largely written.

Appendix: The convenient parameterisation of the Normal Inverse Gamma distribution

The Normal Inverse Gamma distribution for $[x, \tau]$, where x is a k-vector and τ a positive scalar, has the general form

$$x \mid \tau \sim \mathrm{N}_k\big(m, \tau V\big) \tag{A1a}$$
$$\tau \sim \mathrm{IG}(a, d). \tag{A1b}$$

It can also be written $[x, \tau] \sim \mathrm{NIG}_k(m, V, a, d)$. The convenient parameterisation of the IG is

$$\mathrm{IG}(\tau; a, d) = \frac{(d/2)^{a/2}}{\Gamma(a/2)} \tau^{-(1+a/2)} \exp\{-(d/2)\tau^{-1}\} \tag{A2}$$

where a denotes the degrees of freedom (i.e., shape) and d the scale. The mean of this distribution is $d/(a-2)$. Integrating out τ,

$$\pi(x) \propto \big(1 + (x-m)^T (d\,V)^{-1} (x-m)\big)^{(k+a)/2} \tag{A3}$$

or, in the standard parameterisation of the Multivariate Student-t,

$$x \sim \mathrm{St}_k\big(m, (d/a)V, a\big) \tag{A4}$$

which implies $\mathrm{E}\big(x\big) = m$, $\mathrm{Var}\big(x\big) = \{d/(a-2)\}V$.

If $x = [x_1, x_2]$ with lengths k_1 and k_2, then

$$\pi(x_1, \tau \mid x_2) = \pi(x_1 \mid \tau, x_2)\ \pi(\tau \mid x_2)\,. \tag{A5}$$

The first distribution is Gaussian:

$$x_1 \mid \tau, x_2 \sim \mathrm{N}_{k_1}\big(m_{1\cdot 2}, \tau V_{1\cdot 2}\big) \tag{A6a}$$

where

$$m_{1\cdot 2} \triangleq m_1 + V_{12}(V_{22})^{-1}(x_2 - m_2) \tag{A6b}$$
$$V_{1\cdot 2} \triangleq V_{11} - V_{12}(V_{22})^{-1} V_{21}. \tag{A6c}$$

The second distribution is Inverse Gamma:

$$\tau \mid x_2 \sim \mathrm{IG}\big(a + k_2, d + \mathrm{mhd}\big) \tag{A7a}$$

where

$$\mathrm{mhd} \triangleq (x_2 - m_2)^T (V_{22})^{-1} (x_2 - m_2). \tag{A7b}$$

Thus the updated distribution $[x_1, \tau] \mid x_2$ remains NIG:

$$[x_1, \tau] \mid x_2 \sim \text{NIG}_{k_1}\big(m_{1\cdot 2}, V_{1\cdot 2}, a + k_2, d + \text{mhd}\big). \tag{A8}$$

References

M.J. Bayarri, J.O. Berger, J. Cafeo, G. Garci-Donato, F. Liu, Parthasarathy R.J. Palomo, R. Paulo, J. Sacks, and D. Walsh, 2007. Computer model validation with functional output. *Annals of Statistics*. To appear.

M.J. Bayarri, J.O. Berger, M. Kennedy, A. Kottas, R. Paulo, J. Sacks, J.A. Cafeo, C.H. Lin, and J. Tu. Validation of a computer model for vehicle collision. Technical Report 163, National Institute of Statistical Sciences, 2005.

M. Brookes, 2005. The Matrix Reference Manual. On-line, `http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html`.

S. Conti and A. O'Hagan, 2007. Bayesian emulation of complex multi-output and dynamic computer models. In submission, currently available at `http://www.tonyohagan.co.uk/academic/ps/multioutput.ps`.

P.S. Craig, M. Goldstein, J.C. Rougier, and A.H. Seheult, 2001. Bayesian forecasting for complex systems using computer simulators. *Journal of the American Statistical Association*, **96**, 717–729.

P.S. Craig, M. Goldstein, A.H. Seheult, and J.A. Smith, 1996. Bayes linear strategies for matching hydrocarbon reservoir history. In J.M. Bernado, J.O. Berger, A.P. Dawid, and A.F.M. Smith, editors, *Bayesian Statistics 5*, pages 69–95. Oxford: Clarendon Press.

P.S. Craig, M. Goldstein, A.H. Seheult, and J.A. Smith, 1997. Pressure matching for hydrocarbon reservoirs: A case study in the use of Bayes Linear strategies for large computer experiments. In C. Gatsonis, J.S. Hodges, R.E. Kass, R. McCulloch, P. Rossi, and N.D. Singpurwalla, editors, *Case Studies in Bayesian Statistics III*, pages 37–87. New York: Springer-Verlag. With discussion.

C. Currin, T.J. Mitchell, M. Morris, and D. Ylvisaker, 1991. Bayesian prediction of deterministic functions, with application to the design and analysis of computer experiments. *Journal of the American Statistical Association*, **86**, 953–963.

R. Furrer, M.G. Genton, and D. Nychka, 2006. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, **15**, 502–523.

M.G. Genton, 2007. Separable approxmiations of space-time covariances. *Environmetrics*. forthcoming.

M. Goldstein and J.C. Rougier, 2004. Probabilistic formulations for transferring inferences from mathematical models to physical systems. *SIAM Journal on Scientific Computing*, **26**(2), 467–487.

M. Goldstein and J.C. Rougier, 2006. Bayes linear calibrated prediction for complex systems. *Journal of the American Statistical Association*, **101**, 1132–1143.

M. Goldstein and J.C. Rougier, 2007. Reified Bayesian modelling and inference for physical systems. *Journal of Statistical Planning and Inference*. Forthcoming as a discussion paper, currently available at `http://www.maths.dur.ac.uk/stats/people/jcr/Reify.pdf`.

G.H. Golub and C.F. Van Loan, 1996. *Matrix Computations*. Baltimore: Johns Hopkins University Press, 3rd revised edition.

D. Higdon, J. Gattiker, B. Williams, and M. Rightley. Computer model calibration using high dimensional output. Technical Report LA-UR-07-1444, Los Alamos National Laboratory, 2007. To appear in the *Journal of the American Statistical Association*.

M.C. Kennedy and A. O'Hagan, 2001. Bayesian calibration of computer models. *Journal of the Royal Statistical Society, Series B*, **63**, 425–464. With discussion.

K.V. Mardia, J.T. Kent, and J.M. Bibby, 1979. *Multivariate Analysis*. London: Harcourt Brace & Co.

A. O'Hagan, 1998. A Markov property for covariance structures. Unpublished, available at `http://www.shef.ac.uk/~st1ao/ps/kron.ps`.

A. O'Hagan, 2006. Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering and System Safety*, **91**, 1290–1300.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004. ISBN 3-900051-00-3, `http://www.R-project.org`.

J.C. Rougier, 2007. Lightweight emulators for multivariate deterministic functions. Unpublished, available at `http://www.maths.bris.ac.uk/~mazjcr/lightweight1.pdf`.

J.C. Rougier, S. Guillas, A. Maute, and A. Richmond, 2007. Emulating the Thermosphere-Ionosphere Electrodynamics General Circulation Model (TIE-GCM). In submission, currently available at `http://www.maths.bris.ac.uk/~mazjcr/EmulateTIEGCM.pdf`.

J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn, 1989. Design and analysis of computer experiments. *Statistical Science*, **4**(4), 409–423. With discussion, pp. 423–435.

T.J. Santner, B.J. Williams, and W.I. Notz, 2003. *The Design and Analysis of Computer Experiments*. New York: Springer.

H. Wendland, 1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, **4**(1), 389–396.