Efficient Emulators for Multivariate Deterministic Functions

Jonathan Rougier[*]
Department of Mathematics
University of Bristol, UK

June 13, 2008

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 residual covariance function is treated as separable in the model-inputs and model-outputs. Here an additional simplification on the structure of the regressors in the emulator mean function allows very substantial further gains. The result is that it is now possible to emulate rapidly—on a desktop computer—models with hundreds of evaluations and hundreds of outputs. This is demonstrated through calculating costs in floating-point operations, and in an illustration. Even larger sets of outputs are possible if they have additional structure, e.g. spatial-temporal.

KEYWORDS: GAUSSIAN PROCESS, 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 (Craig *et al.*, 1997, 2001; Kennedy and O'Hagan, 2001; Bayarri *et al.*, 2005; Conti and O'Hagan, 2007; Higdon *et al.*, 2007; Bayarri *et al.*, 2007). A standard framework for assimilating the data into an emulator—scalar or multivariate—is described in section 2, 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 3 describes a standard approach for simplifying the update in this case, which is to impose separability in the residual covariance function. Such an emulator might be termed a 'separable emulator'.

This simplification only goes so far, though, and for large problems it has been necessary, typically, to dimensionally-reduce the number of output components, e.g. by modelling the first few principal components. Section 4 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 algebraically, 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'.

Both sections 3 and 4 have separate subsections on computation. Accuracy and efficiency are absolutely crucial in large applications, where the maximum possible benefit needs to be extracted in linear algebra operations using matrices with special structure, such positive definiteness and Kronecker product form. The costs of different approaches are contrasted in floating point operations (flops), depending on the size of the emulator, in particular the number of evaluations and the number of output-components. Section 4 also contains a subsection contrasting the outer-product emulator with the multivariate emulator proposed by Conti and O'Hagan (2007).

The power of the two simplifications is illustrated in section 5, 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 sections 3 and 4. Section 6 concludes, indicating a further extension suitable for 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 Conjugate emulator

2.1 Model structure

Consider a deterministic computer model with model-input $r \in \mathcal{R} \subset \mathbb{R}^p$, producing a set of outputs. 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 output can be represented as

$$f(r) \triangleq \big(f_1(r), \dots, f_q(r)\big)^T \tag{1}$$

where q is the number of outputs. 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. diffusivity), 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 the output index would be $s_j = (x_j, y_j, z_j)$. In a more advanced scheme with an adaptive spatial grid, the raw output for input r would depend on r; this raw output must then be post-processed in order that the model can be treated as having regular outputs. If the model-output comprised more than one type, such as salinity and pressure for each cell, then s_j would be extended to include a factor for the type of output, such as $s_j = (x_j, y_j, z_j, \mathsf{T}_j)$, where $\mathsf{T}_j \in \{\mathsf{S}, \mathsf{P}\}$.

2.2 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.

In this paper, the emulator is represented in the general form

$$f_j(r) = \sum_{k=1}^{v} \beta_k \, g_k(r, s_j) + e(r, s_j) \quad j = 1, \dots, q, \tag{2}$$

where $\mathcal{G} \triangleq \{g_1(\cdot), \dots, g_v(\cdot)\}$ are specified regressors defined on $\mathcal{R} \times \mathcal{S}$, $\beta \triangleq \big(\beta_1, \dots, \beta_v\big)^T$ is a vector of uncertain regression coefficients, and $e(\cdot)$ is an uncertain residual, represented as a stochastic process defined on $\mathcal{R} \times \mathcal{S}$.

If we specify a joint distribution for β and $e(\cdot)$, then (2) induces a distribution over any finite collection of model evaluations. For tractability, we choose a Normal Inverse Gamma distribution, introducing a common uncertain variance multiplier τ:

$$\beta \perp\!\!\!\perp e(\cdot) \mid \tau \tag{3a}$$

$$\beta \mid \tau \sim \mathrm{N}_v\big(m, \tau V\big) \tag{3b}$$

$$e(\cdot) \mid \tau \sim \mathrm{GP}\big(0, \tau\kappa(\cdot)\big) \tag{3c}$$

$$\tau \sim \mathrm{IG}(a, d). \tag{3d}$$

Here $\mathrm{N}_v(\cdot)$ denotes a multivariate Gaussian distribution with specified mean vector and variance matrix, $\mathrm{GP}(\cdot)$ denotes a Gaussian process with specified mean function and covariance function, and $\mathrm{IG}(\cdot)$ denotes an Inverse Gamma distribution. The residual mean function is simply $E\big(e(r, s)\big) = 0$ for all r and s, and the residual covariance function $\kappa(\cdot)$ is defined on $(\mathcal{R} \times \mathcal{S})^2$.

Eq. (3) 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 conditional distributions, are summarised in the Appendix. Hence our emulator for $f(\cdot)$ is specified by a choice of regressors $\mathcal{G}$ and then, based on these, the Normal Inverse Gamma (NIG) hyperparameters $\{m, V, a, d\}$ and the residual covariance function $\kappa(\cdot)$.

2.3 Updating and predicting

Denote any finite collection of m model-inputs as

$$\boldsymbol{R} \triangleq \begin{pmatrix} R_1^T \\ \vdots \\ R_m^T \end{pmatrix} = \begin{pmatrix} R_{11} & \dots & R_{1p} \\ \vdots & \ddots & \vdots \\ R_{m1} & \dots & R_{mp} \end{pmatrix} \tag{4}$$

where $\boldsymbol{R}$ is used to denote an arbitrary finite collection of model-inputs, as opposed to R and R' below, which denote the inputs where we have evaluated the model, and the inputs where we wish to predict the model-output.

It will be convenient to represent any finite collection of model-outputs as a vector, which also encompasses the special case where the model-output is a scalar. Define the mq-vector of model-outputs as

$$y(\boldsymbol{R}) \triangleq \operatorname{vec} \begin{pmatrix} f(R_1)^T \\ \vdots \\ f(R_m)^T \end{pmatrix} \tag{5}$$

where the operator 'vec' stacks the columns of a matrix into a vector, from left to right so that in running along $y(\boldsymbol{R})$, the model-evaluation index changes faster than the model-output index. The regression matrix $G(\boldsymbol{R})$ is defined in two stages. First, for any regressor $g_k(\cdot)$, define

$$G_{(k)}(\boldsymbol{R}) \triangleq \operatorname{vec} \begin{pmatrix} g_k(R_1, s_1) & \dots & g_k(R_1, s_q) \\ \vdots & \ddots & \vdots \\ g_k(R_m, s_1) & \dots & g_k(R_m, s_q) \end{pmatrix}. \tag{6a}$$

Then define the matrix $G(\boldsymbol{R})$ as

$$G(\boldsymbol{R}) \triangleq \left[G_{(1)}(\boldsymbol{R}), \dots, G_{(v)}(\boldsymbol{R})\right] \tag{6b}$$

i.e. the columns $G_{(k)}(\boldsymbol{R})$ stacked together, into an $mq \times v$ matrix. For the residual, define

$$e(\boldsymbol{R}) \triangleq \operatorname{vec} \begin{pmatrix} e(R_1, s_1) & \dots & e(R_1, s_q) \\ \vdots & \ddots & \vdots \\ e(R_m, s_1) & \dots & e(R_m, s_q) \end{pmatrix}, \tag{7}$$

another mq-vector. The emulator for the full set of evaluations can be written in vector form as

$$y(\boldsymbol{R}) = G(\boldsymbol{R})\beta + e(\boldsymbol{R}). \tag{8}$$

In any particular application we will have performed n evaluations for the design matrix R: n may be zero, in which case all predictions are trivial (provided that the NIG distribution is proper). Suppose therefore that $n > 0$ and we would like to predict the model-output at n' new inputs, in the design matrix R'. For simplicity, write $y(R) = y$ and $y(R') = y'$, similarly for G and G', and e and e', suppressing the R and R' arguments. The update of y' by y proceeds by conditioning. The observations and predictands are both known linear combinations of the vector (β, e, e'),

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

Conditional on τ, (β, e, e') and hence (β, e', y) is multivariate Normal, and so $(\beta, e') \,|\, y$ is multivariate Normal, and $y' \,|\, y$ is multivariate Normal. Integrating out τ, the predictand y' is multivariate t. The central part of the update is computing the mean and variance of $(\beta, e') \mid (y, \tau)$. The reason for focusing on (β, e') rather than y' directly is that the update of β only has to happen once: it does not depend on R'.

The multivariate Normal update of (β, e') by y, taking τ as given, is centred around the $nq \times nq$ variance matrix

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

where $W \triangleq \mathrm{Var}(e)$, and it follows that $\mathrm{Var}(y \mid \tau) \equiv \tau S$. The updating equations for (β, e') are then

$$\begin{aligned}
E(\beta \mid y, \tau) &= m + VG^T S^{-1}(y - Gm) && \text{(11a)}\\
\mathrm{Var}(\beta \mid y, \tau) &= \tau\{V - VG^T S^{-1}(VG^T)^T\} && \text{(11b)}\\
E(e' \mid y, \tau) &= (W')^T S^{-1}(y - Gm) && \text{(11c)}\\
\mathrm{Var}(e' \mid y, \tau) &= \tau\{W'' - (W')^T S^{-1} W'\} && \text{(11d)}\\
\mathrm{Cov}(\beta, e' \mid y, \tau) &= -\tau\, VG^T S^{-1} W', && \text{(11e)}
\end{aligned}$$

where $W' \triangleq \mathrm{Cov}(e, e')$ and $W'' \triangleq \mathrm{Var}(e')$.

For the final stage, recollect that y' is a known linear combination of (β, e'), and denote the inferred mean and variance of $y' \mid (y, \tau)$ as μ' and $\tau T'$, where T' does not depend on τ. Then the joint distribution of $(y', \tau) \,|\, y$ factorises as

$$\begin{aligned}
y' \mid (y, \tau) &\sim \mathrm{N}_{n'q}(\mu', \tau T') && \text{(12a)}\\
\tau \mid y &\sim \mathrm{IG}(a + nq, d + \xi) && \text{(12b)}
\end{aligned}$$

where ξ is the Mahalanobis distance

$$\xi \triangleq (y - Gm)^T S^{-1}(y - Gm). \tag{12c}$$

The emulator of y' is then

$$y' \sim \mathrm{St}_{n'q}\left(\mu', \frac{d+\xi}{a+nq}\, T', a+nq\right) \tag{13}$$

where $\mathrm{St}(\cdot)$ is a multivariate t distribution. Unless $a + nq$ is small, y' is effectively multivariate Normal with mean μ' and variance $\{(d+\xi)/(a+nq)\}T'$.

A useful test of the implementation of an emulator is to let R' be a subset of the rows of R. Because the emulator is updated by conditioning, it must interpolate R, in the sense that $E\big(f(R_i) \mid y\big) = Y_i$ and $\mathrm{Var}\big(f(R_i) \mid y\big) = \mathbf{0}$, and this is easy to check.

3 Separable residual covariance function

So far, our simplifications in the joint distribution have been restricted to the conditional independence of β and (e, e'), and the NIG distribution for (β, e, e', τ). In this section and the next we make further simplifications to our distribution for $y(\boldsymbol{R})$, concerning the form of the covariance function $\kappa(\cdot)$, in this section, and the regressors $\mathcal{G}$, in section 4.

3.1 The Sherman-Morrison-Woodbury reformulation

The expensive calculation in (11) is the inversion of S. Since S is a variance matrix, the first step is to find the Cholesky decomposition of S, an operation requiring $(nq)^3/3$ flops (Golub and Van Loan, 1996, p. 144). The total cost of inverting S, which also requires a double back-substitution, is $(nq)^3/3 + 2(nq)^3 = 7(nq)^3/3$ flops.

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

$$(A^{-1} + BC^{-1}B^{T})^{-1} = A - AB(C + B^{T}AB)^{-1}(AB)^{T} \tag{14}$$

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

$$S^{-1} = W^{-1} - (W^{-1}G)(V^{-1} + G^{T}W^{-1}G)^{-1}(W^{-1}G)^{T} \tag{15}$$

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 3.2. 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

$$\mathrm{Var}\big(\beta \mid y, \tau\big) \equiv \tau D^{-1} \quad \text{where} \quad D \triangleq V^{-1} + G^{T} W^{-1} G \tag{16}$$

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}:

$$S^{-1} = W^{-1} - (W^{-1}G) D^{-1} (W^{-1}G)^{T}. \tag{15$'$}$$

3.2 Separability in the residual variance

Specifying the residual covariance function over $(\mathcal{R} \times \mathcal{S})^2$ can be demanding, and it is natural to simplify this task by treating $\kappa(\cdot)$ as separable in r and s, so that

$$\kappa(r, s_j, r', s_{j'}) = \kappa^{r}(r, r') \times \kappa^{s}_{jj'}. \tag{17}$$

Note that separability in the covariance function of the residual does not imply separability in the covariance function of 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). Note also that the term in the model-inputs is represented by a covariance function, since the purpose of the emulator is to predict the model at any point in $\mathcal{R}$. The term in the output index, however, can be represented as a variance matrix, because the outputs are the same for every evaluation that we have or want to predict.

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

$$W = W^{s} \otimes W^{r} \tag{18a}$$

where '$\otimes$' denotes the Kronecker product, W^{s} is the $q \times q$ variance matrix

$$W^{s} \triangleq \begin{pmatrix} \kappa^{s}_{11} & \cdots & \kappa^{s}_{1q} \\ \vdots & \ddots & \vdots \\ \kappa^{s}_{q1} & \cdots & \kappa^{s}_{qq} \end{pmatrix} \tag{18b}$$

and W^{r} is the $n \times n$ Gram matrix of $k^{r}(\cdot)$ for R. Using the general result that $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$, the cost of the inverting W in this case is not $7(nq)^3/3$ flops but $7(n^3 + q^3)/3$ flops. There are also Kronecker factorisations for W' and W'', defined after (11). The properties of the Kronecker product can be found in, e.g., Golub and Van Loan (1996, sec. 4.5.5) or Brookes (2005, Matrix Relations).

A similar approach has been proposed when the outputs are not regular or $\kappa(\cdot)$ 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 by tapering W^r or W^s (Furrer *et al.*, 2006).

3.3 Computation

One approach to computation is to use the SMW formula is to compute S^{-1}, and then plug the value for S^{-1} into (11), the original updating equations. However, this simple plug-in approach ignores the additional benefits of separability in the residual variance. These benefits accrue from working as much as possible with the original objects, which permits algebraic simplifications in the updating equations. These algebraic simplifications are exact, while numerical calculations that do not take advantage of them are only approximate. Therefore this subsection is about enhancing accuracy and efficiency.

What can go wrong if S^{-1} is plugged in? There are two sources of numerical error. First, plugging-in S^{-1} breaks the standard rule of not inverting matrices in place, but rather solving them as systems of linear equations (the "pitfall of explicit inverse computation", Golub and Van Loan, 1996, p. 121). In other words, when evaluating $X = S^{-1}B$ it is better to solve $SX = B$ for X than to compute S^{-1} first and then perform the matrix multiplication. Second, plugging-in S^{-1} fails to take advantage of the fact that S is positive definite symmetric, with Cholesky decomposition $S = Q^T Q$. Suppose we had Q. In this case we could evaluate expressions such as $AS^{-1}B$ in two back-substitutions and a matrix product: $(Q^{-T}A^T)^T Q^{-T}B$. An obvious benefit of this approach is that symmetric expressions such as $AS^{-1}A^T$ can be computed efficiently, and are sure to be symmetric. But proceed in this way would incur a cost of $(nq)^3/3$ to find Q. Thus plugging-in S^{-1} is incompatible with a careful treatment of matrix expressions involving S^{-1}.

By proceeding algebraically, however, we can achieve an efficient calculation *and* a careful treatment of S^{-1}. Starting from (11), (12c), (15′), and

(16), the expressions that must be computed in order to update (β, e') are:

$$c \triangleq y - Gm \tag{19a}$$

$$D \triangleq V^{-1} + (G^T W^{-1} G) \tag{19b}$$

$$E(\beta \mid \tau, y) = m + V\{(G^T W^{-1} c) - (G^T W^{-1} G) D^{-1} (G^T W^{-1} c)\} \tag{19c}$$

$$\operatorname{Var}(\beta \mid \tau, y) = \tau D^{-1} \tag{19d}$$

$$E(e' \mid \tau, y) = (W')^T W^{-1} c - (G^T W^{-1} W')^T D^{-1} (G^T W^{-1} c) \tag{19e}$$

$$\begin{aligned}\operatorname{Var}(e' \mid \tau, y) = \tau \Big\{ & W'' - (W')^T W^{-1} W' \\ & + (G^T W^{-1} W')^T D^{-1} (G^T W^{-1} W') \Big\}\end{aligned} \tag{19f}$$

$$\operatorname{Cov}(\beta, e' \mid \tau, y) = -\tau V \big\{ G^T W^{-1} W' - (G^T W^{-1} G) D^{-1} (G^T W^{-1} W') \big\} \tag{19g}$$

$$\xi = c^T W^{-1} c - (G^T W^{-1} c)^T D^{-1} (G^T W^{-1} c) \tag{19h}$$

(with some re-arrangement to highlight the compound terms). The terms involving combinations of W's can be simplified algebraically. In particular, define

$$W^{r\prime} \triangleq \begin{pmatrix} \kappa^r(R_1, R'_1) & \dots & \kappa^r(R_1, R'_{n'}) \\ \vdots & \ddots & \vdots \\ \kappa^r(R_n, R'_1) & \dots & \kappa^r(R_n, R'_{n'}) \end{pmatrix} \tag{20a}$$

then

$$W^{-1} = W^{-s} \otimes W^{-r} \tag{20b}$$

$$W' = W^s \otimes W^{r\prime} \tag{20c}$$

$$W^{-1} W' = (W^{-s} \otimes W^{-r})(W^s \otimes W^{r\prime}) = I_q \otimes W^{-r} W^{r\prime} \tag{20d}$$

$$(W')^T W^{-1} W' = W^s \otimes (W^{r\prime})^T W^{-r} W^{r\prime}, \tag{20e}$$

where $W^{-r} \triangleq (W^r)^{-1}$ and similarly for W^{-s}. Note, for example, that (20d) is block-diagonal, and that all the blocks are the same. This structure would be buried within the plug-in approach, and would be compromised by the finite precision of floating point operations.

Terms involving G and the W's will be much further simplified following the additional structure proposed in the next section. To illustrate the practical benefits of working in terms of the W's, though, consider the expression $W^{-1}c$. Assume that the Cholesky decompositions of W^r and W^s have been computed; denote these as Q^r and Q^s, respectively. On the one hand we have the plug-in-style calculation

$$(W^s \otimes W^r)^{-1} c = \big(Q^{-s}(Q^{-s})^T \otimes Q^{-r}(Q^{-r})^T\big) c \tag{21}$$

where $Q^{-r} \triangleq (Q^r)^{-1}$, and similarly for Q^{-s}. The cost in flops is

$$(q^3 + q^3) + (n^3 + n^3) + n^2q^2 + 2(nq)^2.$$

The first parenthetical term is for W^{-s}: a back-substitution to find Q^{-s} and then a symmetric cross-product to find $Q^{-s}(Q^{-s})^T$; the second is the same for W^{-r}; the next term is the Kronecker product; the final term is the product with c. A better way to proceed is to appreciate that $x = W^{-1}c$ is the solution of the linear equation $W^r X W^s = C$, where $x \equiv \operatorname{vec} X$ and $c \equiv \operatorname{vec} C$ (Golub and Van Loan, 1996, p. 181). First solve $W^r Z = C$ for Z, then solve $W^s X^T = Z^T$ for X. Starting with the Cholesky decompositions, each set of linear equations requires a double back-substitution, so the cost in flops is

$$2n^2q + 2q^2n.$$

To illustrate, consider a modest emulator with $n = 30$ and $q = 10$. In this case the cost of the first calculation is 326 Kflops, and that of the second is 24 Kflops, i.e. an order of magnitude smaller. This is only a guide to the accuracy and efficiency advantages of exploiting the algebraic structure of S^{-1} rather than just plugging-in, since $W^{-1}c$ is computed implicitly in the plug-in. But it is highly suggestive, and borne out in the illustration in section 5.

4 Outer-product emulators

In this section we restrict our joint distribution for $y(\boldsymbol{R})$ further, with a particular structure for the regressors.

4.1 Factorising the regression matrix

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

$$f(s) \approx \sum_{k'=1}^{v_s} \alpha_{k'}\, g^s_{k'}(s) \qquad \text{for fixed } r \tag{22}$$

where $\mathcal{G}^s \triangleq \{g^s_1(\cdot), \dots, g^s_{v_s}(\cdot)\}$. Then we think of the coefficients as being themselves uncertain linear combinations of specified basis functions in r:

$$\alpha_{k'} = \alpha_{k'}(r) \approx \sum_{k''=1}^{v_r} \beta_{k'k''}\, g^r_{k''}(r) \quad k' = 1, \dots, v_s, \tag{23}$$

where $\mathcal{G}^r \triangleq \{g_1^r(\cdot), \dots, g_{v_r}^r(\cdot)\}$. Note that we are using the same regressors for each coefficient. Combining these gives

$$f(r,s) \approx \sum_{k'=1}^{v_s} \sum_{k''=1}^{v_r} \beta_{k'k''}\, g_{k''}^r(r)\, g_{k'}^s(s) \equiv \sum_{k=1}^{v} \beta_k\, g_k(r,s) \tag{24}$$

where $v = v_s v_r$ and $\mathcal{G}$ is the set of pairwise products of the regressors in $\mathcal{G}^r$ and in $\mathcal{G}^s$.

Now define separate regression matrices for r and s:

$$G^r \triangleq \begin{pmatrix} g_1^r(R_1) & \dots & g_{v_r}^r(R_1) \\ \vdots & \ddots & \vdots \\ g_1^r(R_n) & \dots & g_{v_r}^r(R_n) \end{pmatrix} \quad \text{and} \quad G^s \triangleq \begin{pmatrix} g_1^s(s_1) & \dots & g_{v_s}^s(s_1) \\ \vdots & \ddots & \vdots \\ g_1^s(s_q) & \dots & g_{v_s}^s(s_q) \end{pmatrix}. \tag{25}$$

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, and so only has to be computed once. The crucial result is that in an emulator such as (24), the regression matrix G from (6) can be factorised as

$$G = G^s \otimes G^r. \tag{26}$$

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

4.2 Computation

In an outer-product emulator, the Kronecker factorisations of G and W are conformable. This means there are algebraic simplifications for the terms combining G and W.

Consider, to start with, the term

$$G^T W^{-1} = (G^s \otimes G^r)^T (W^{-s} \otimes W^{-r}) \equiv H^s \otimes H^r \tag{27}$$

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

$$G^T W^{-1} G = H^s G^s \otimes H^r G^r \tag{28a}$$

$$G^T W^{-1} W' = (G^s \otimes G^r)^T (I_q \otimes W^{-r} W^{r\prime}) = (G^s)^T \otimes H^r W^{r\prime} \tag{28b}$$

$$G^T W^{-1} c = \operatorname{vec}\{H^r C (H^s)^T\}. \tag{28c}$$

Note that in (28a), $H^s G^s \equiv (G^s)^T W^{-s} G^s$, and similarly for $H^r G^r$. Starting from the Cholesky decomposition of W^s, Q^s say, it is better to compute

(28a) as

$$H^s G^s = \{(Q^s)^{-T} G^s\}^T \{(Q^s)^{-T} G^s\}, \qquad (28a')$$

and similarly for $H^r G^r$, involving one back-substitution and one symmetric cross-product.

As an illustration of the further benefits the come from an outer-product emulator, consider the cost of computing $G^T W^{-1} c$ without and with the factorisation of G. Without the factorisation this cost is

$$2n^2 q + 2q^2 n + 2vnq$$

flops, i.e. the same as before plus the matrix product $G^T(W^{-1}c)$. With the factorisation of G the cost is

$$2v_r nq + 2v_r q v_s + 2n^2 v_r + 2q^2 v_s$$

flops, including the cost of computing H^r and H^s (third and fourth terms). For the modest emulator, setting $v_r = v_s = 4$, these two costs are 34 Kflops and 11 Kflops (and H^r will be reused in the calculation of $G^T W^{-1} W'$).

4.3 Relation to the emulator of Conti and O'Hagan (2007)

This subsection explores the relationship between the outer-product emulator and the multivariate emulator of Conti and O'Hagan (2007, hereafter CoH). Although superficially different, the CoH emulator is a special case of the outer-product emulator, and thus all of the computational benefits outlined in this section transfer directly.

We return to the arbitrary finite collection of model evaluations $\boldsymbol{R}$ introduced at the start of section 2, but now expressed for the restrictions of the outer-product emulator. Written in matrix form, the outer-product emulator is

$$Y = G^r B (G^s)^T + E \qquad (29)$$

where $\beta \equiv \operatorname{vec} B$ where B is a $v_r \times v_s$ matrix of uncertain regression coefficients, and $e \equiv \operatorname{vec} E$ where E is the $m \times q$ residual matrix in (7). CoH propose an apparently different multivariate model, namely

$$Y = G^r D + E \qquad (30)$$

where D is a $v_r \times q$ matrix of uncertain regression coefficients. Clearly, (30) is a special case of (29), with $G^s = I_q$: identify s_j with j, and then define the regressors in $\mathcal{G}^s$ as $s_k(s_j) = \delta_{kj}$ for $k = 1, \dots, q$, where δ_{kj} is the Kronecker delta. The two emulators are equivalent in general, since the mean and variance of D can always be specified consistently with the mean and variance of B and the matrix G^s, but not in the particular case proposed by CoH, where a non-informative prior is used for D.

The CoH emulator illustrates two other features of the outer-product emulator. First, the regression coefficients in the CoH emulator all have well-defined units, because each model-output has its own column in D. The regression coefficients in the outer-product emulator have, in general, indeterminate units if there is more than one output-type, which makes it more difficult to specify a prior mean and variance (m and V). The special case of $G^s = I_q$ which makes the outer-product emulator into the CoH emulator illustrates how to circumvent this problem: a block-diagonal structure in G^s where the blocks are organised according to the type indicator in s_j. This is implemented by including a Kronecker delta in each of the components of $\mathcal{G}^s$, which selects a particular output-type. This block-diagonal structure also means that different output-types can have different sets of regressors. Having several output-types also affects the interpretation of the common variance multiplier τ in the variance of β and the residual. Where there are multiple output-types it is natural to treat the variance multiplier as unitless and $\kappa^s(\cdot)$ as a correlation function, and ensure that V and W^s are scaled appropriately.

Second, the two emulators differ in how they treat of the output domain $\mathcal{S}$. In CoH this is simply the index set $\{1, \dots, q\}$. Therefore there can be no prior structure across the output components other than that specified in the prior variance of D and the prior for W^s (with CoH's non-informative prior for $\{D, W^s\}$ there is no prior structure at all). This is unattractive if part of $\mathcal{S}$ is a metric space, e.g. if the s_j index different spatial locations or times. The outer-product emulator, by allowing this type of structure to be representing explicitly in the prior through the choice of $\mathcal{G}^s$, can borrow strength across Y, which will be important when the number of evaluations is small. A referee has pointed out that a metric space in $\mathcal{S}$ also allows us to predict 'new' model-output values for any r at output indices $s', s'', \dots$ which do not correspond to any of the actual model-outputs $s_1, \dots, s_q$. This would involve a straightforward generalisation of the calculations given here.

5 Illustration

This illustration demonstrates the efficiency of the outer-product emulator over a range of values for n, the number of evaluations in the ensemble, and q, the number of model-outputs. 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 outer-product emulator will also be more accurate, with less potential for numerical errors from finite precision arithmetic.

The object is to predict ten further evaluations (i.e., $n' = 10$); in all cases

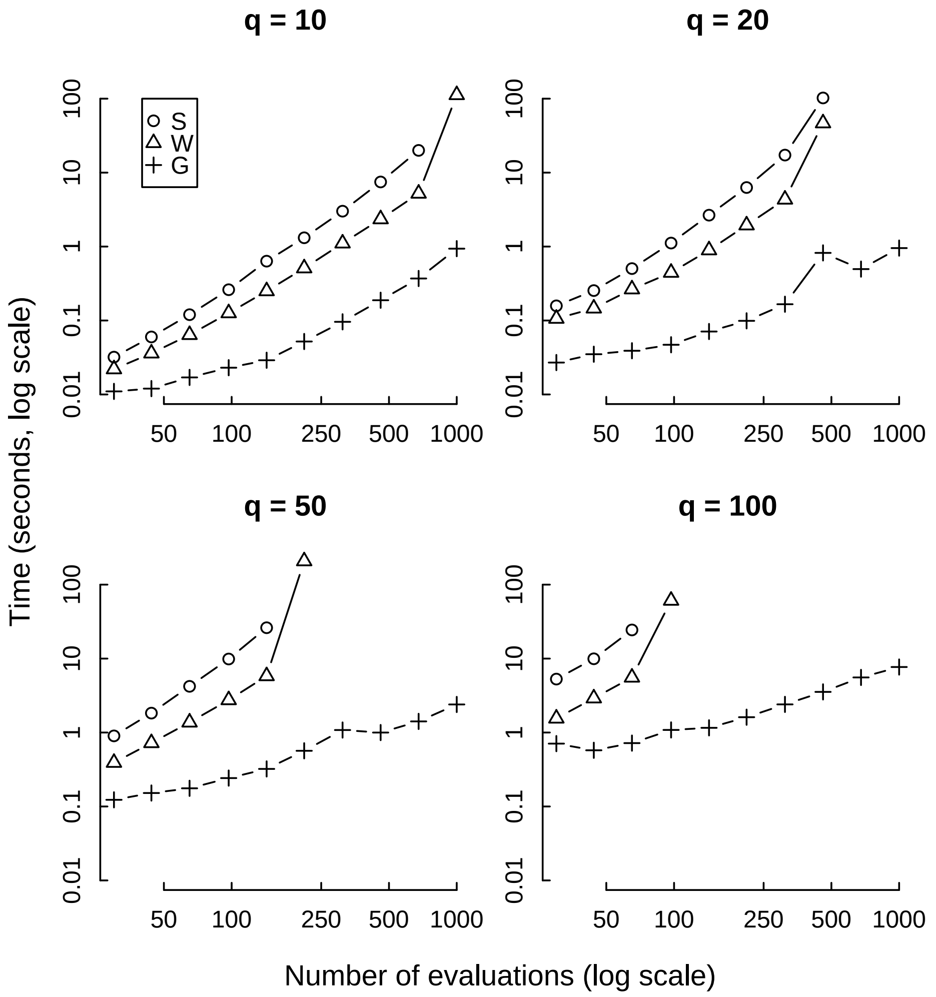

Figure 1: Timings for the three methods of constructing the emulator and predicting $n' = 10$ new evaluations, for varying n and q, with $v_r = 7$, $v_s = 4$. Method S, direct inversion of S (section 2.3); Method W, SMW inversion of S with a separable residual covariance function (section 3.3); Method G, outer-product emulator (section 4.2). The timings are total elapsed time in seconds. Where timings are not shown for a given n, it is because there was not enough memory to complete the calculation.

$v = 28$, with $v_r = 7$ and $v_s = 4$. Three calculations are considered. First, the direct inversion of S, following the calculations in (11) in section 2.3 (method S). Second, the construction of S^{-1} using the SMW formula and the additional simplifications that arise from a separable residual covariance function, as described in section 3.3 (method W). Third, the further additional simplifications that arise in an outer-product emulator, as described in section 4.2 (method G).

The calculations are implemented in the statistical computing environment R (R Development Core Team, 2004). 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 results are given in Figure 1, which speaks for itself: the performance of the outer-product emulator is really astounding. It computes in about one second a prediction that neither of the other two approaches could compute at all (bottom right panel), and easily handles 100,000 components ($q = 100$, $n = 1000$). But it is also interesting to note that while separability of the residual covariance function confers a reasonable speed improvement over direct inversion of S, it adds very little in terms of the size of the dataset that can be handled. This suggests that the effective size of the calculation is determined mainly by the combined G and W terms, rather than just the W terms.

In terms of accuracy, all three methods gave the same results to a tolerance of 10^{-7} (using the R function `all.equal`).

6 Discussion

An outer-product emulator has three features. First, the underlying model has regular outputs, i.e. 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 covariance function in the emulator separates into the product of a term in the model-input values and a term in the model-output index. This separability is a very natural choice for constructing the joint covariance function, and is a weaker choice than is typically made in scalar emulator construction, where the covariance function for the model-input values is itself 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 pairwise product of a set of regressors in the model-inputs and a set of regressors in the model-outputs.

As illustrated in section 5, the outer-product emulator can 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-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 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 is possible because the regression matrix never has to be explicitly evaluated.

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 matrix for the model-output index 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 pairwise product of space regressors and time regressors, then exactly the same Kronecker product approach could be used to factor the regression matrix G^s and the residual 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 emulator could also be treated as separable in time and angle, permitting the inclusion of many more evaluations, 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, Tony O'Hagan and the JCGS referees for their very perceptive comments, 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,

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

or, in the standard parameterisation of the multivariate t,

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

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

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:

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

where

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

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

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

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, 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 approximations of space-time covariance matrices. *Environmetrics*, **18**(7), 681–695.

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.

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, 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.