Small Sample Designs for Complex High-Dimensional Models Based on Fast Approximations

Jonathan Cumming and Michael Goldstein

Department of Mathematical Sciences, Durham University, U.K.

April 23, 2008

Abstract

Computer simulators are commonly used in many areas of science to investigate complex physical systems of interest. The output of such simulators is often expensive and time-consuming to obtain, thus limiting the amount of information we can gather about their behaviour. In some cases, an approximate simulator is also available, which can be evaluated in a fraction of the time and which shares many qualitative features with the original simulator. Since the approximate simulator is cheap to evaluate, large numbers of evaluations can be performed and its behaviour can be investigated thoroughly. We propose a technique to construct efficient small sample designs for runs of the full simulator by exploiting the information gained from the approximate simulator via Bayes linear methods. The methodology is illustrated with an example concerning a computer simulation of a hydrocarbon reservoir.

Keywords: Computer experiments; Experimental design; Bayes linear methods; Multi-level emulation; Hydrocarbon reservoir.

1 Introduction

Computer simulators are used by scientists to gain insight into the behaviour of complex physical systems. Example application areas include complex electronic circuits, fluid flow in underground oil reservoirs, the potential global effects of climate change, and the formation of entire galaxies. The simulator implements the physical laws thought to underlie the behaviour of the system with mathematical equations. Given a collection of input values, it then solves the equations numerically, producing a high-dimensional collection of outputs, such as a number of time series. The simulator is a deterministic model and therefore produces the same output values when re-evaluated for the same set of inputs.

Evaluating complex simulators for a given choice of input values can take a prohibitively long time and be extremely costly. This can severely limit the number of evaluations of the simulator that can be made within a fixed budget. Consequently, there is much uncertainty about the output of the simulator. To represent this uncertainty, we build a stochastic representation of the simulator, known as an *emulator* by using the available model evaluations and appropriate prior knowledge (Sacks, Welch, Mitchell, and Wynn, 1989; Currin, Mitchell, Morris, and Ylvisaker, 1991; Craig, Goldstein, Rougier, and Seheult, 2001; Santner, Williams, and Notz, 2003; O'Hagan, 2006).

In some cases, approximate versions of the simulator for the same system are available, which can be evaluated for a fraction of the time and cost of the original simulator, albeit with a lower accuracy. We refer to the approximate version as the *coarse* simulator, and the original simulator as the *fine* simulator to represent this difference in accuracy. Since the coarse and fine simulators model the same system, we expect that they will share some of the same qualitative features and behaviours in relation to the simulator inputs. Therefore, we can use many evaluations of the coarse simulator to supplement the sparse evaluations of the fine simulator and so reduce our uncertainty about the fine simulator. When we have two versions of the simulator, then we construct a multilevel emulator (Craig, Goldstein, Seheult, and Smith, 1998; Kennedy and

O'Hagan, 2000; Qian and Wu, 2008).

Since we may only make a relatively small number of evaluations of the fine simulator, these evaluations must be carefully chosen. For functions with high dimensional output spaces, each function evaluation is informative for each of a large number of output functions. Therefore, the problem of choosing a small collection of input values which is highly informative for all of the outputs is very challenging. The purpose of this paper is to describe a tractable strategy for making such small sample designs for high dimensional functional output. The strategy is based on the decomposition of the design problem into sub-problems based on the assessment of a version of a conditional independence representation for the structure of the design.

We begin in Section 2 by establishing some basic notation and terminology and outlining our general design and analysis strategy for a multilevel computer model, using coarse simulator information. Section 3 describes our approach to emulation, and introduces our framework for linking two computer simulators. Section 4 concerns the structure of the design problem and our basic method for constructing a design, by exploiting the active variable structure of the emulators. Section 5 looks at the specific task of designing for, and performing, emulator tuning, calibration and forecasting. We illustrate our results by application to a computer simulation of a hydrocarbon reservoir in Section 6, and make concluding comments in Section 7.

2 Preliminaries

Learning about a complex high-dimensional function with many outputs from only a limited number of evaluations is a very challenging problem. In this paper, we develop a methodology for designing a small number of runs at appropriate locations by using approximate versions of the computer simulator and by exploiting structural relationships between model inputs.

A computer simulator can be treated as a deterministic function $F(\cdot)$ which models the behaviour of a physical system $\boldsymbol{y}$. The inputs to the simulator, $\boldsymbol{x} = (x_1, \ldots, x_m)$, comprise a vector representing the aspects of the computer model which must be quantified before the simulator can be evaluated. The output of the simulator is $F(\boldsymbol{x})$, a vector representing the resulting behaviour of the physical system given that particular choice of $\boldsymbol{x}$. We describe our uncertainty about the value of the simulator, for each input choice, by means of the stochastic emulator, $f(\boldsymbol{x})$.

For the purposes of this article, we assume that we have two versions of the computer model - the coarse model, F_0, and the fine model, F_1. We suppose that the coarse model is sufficiently fast that we may make as many evaluations as we please, at effectively no cost, so that we may construct a detailed emulator of F_0. Thus we obtain a coarse emulator, $f_0(\boldsymbol{x})$, which we use to to construct an informed prior specification for the emulator, $f_1(\boldsymbol{x})$, of the fine simulator. We then use the fine emulator to determine informative choices of input values for evaluating the fine simulator.

The general strategy proceeds in four stages: emulation, tuning, calibration, and forecasting. The emulation stage requires the construction of the coarse emulator and the development of a corresponding prior fine emulator. The tuning stage uses a small-sample design to assess differences between the two simulators and their implications for a multilevel emulation. The calibration stage uses an additional small design to update the prior fine emulator, and use this updated emulator to perform history matching and reduce the size of the input space. Finally, we update the fine emulator by a further small sample design with the intention of using this emulator in an appropriate final-stage analysis, such as forecasting.

3 Emulation

In this paper, we will construct informative designs for second order emulators of F_1. Such emulators require the specification of the mean function for $F_1(\boldsymbol{x})$ and the covariance function over $(F_1(\boldsymbol{x}), F_1(\boldsymbol{x}'))$ pairs. We may choose this formulation in a particular application because we consider that a Bayes linear uncertainty analysis is appropriate for the problem – see, for example, Goldstein and Rougier (2006); or for a general overview of the Bayes linear approach see Goldstein and Wooff (2007). Alternatively, we may intend to carry out a full Bayes analysis, but find that the Bayes calculations required to identify an informative design for $F_1(\boldsymbol{x})$ are so complex that the approach is computationally intractable. Therefore, we might choose a second order design as a pragmatic approximation to the full design calculations, even if we intend to carry out a full Bayes analysis when we have evaluated the fine simulator at each chosen design value.

3.1 Screening

Let us assume that we have a large batch of runs, $\boldsymbol{F}_0$, of the coarse simulator over an appropriate Latin hypercube design, $\boldsymbol{X}_0$, in the inputs (McKay, Beckman, and Conover, 1979; Santner et al., 2003). Emulation and design are computationally intensive processes, so it is often infeasible to study all outputs from the model when the number of outputs, q, is large. The first step in the emulation process is to screen the outputs to determine which components of the computer simulator output will be emulated, and we then focus our subsequent analysis only on this reduced subset. We favour identifying subsets of outputs rather than, for example, linear combinations since we aim to understand, model and emulate the physical system itself. Additionally, the design methods of Section 4 depend on having emulators which are driven by only a small set of inputs, which may not be the case when considering combinations of outputs due to the increased complexity of the relationships between the inputs and the transformed outputs.

Since a design which is judged efficient for one output is generally efficient for any additional variables which are strongly correlated with this output, the correlation structure between the outputs will be important to the screening mechanism. In order to identify the reduced subset of outputs, S, to emulate we adopt the principal variable selection method of Cumming and Wooff (2007), which operates by scoring each output, y_i, by the value $H_i = \sum_{j=1}^{q} \mathrm{Corr}\left[y_i, y_j\right]^2$, and then selecting the output which maximises this statistic. Subsequent outputs are chosen using the partial correlation of the remaining variables given those already chosen to eliminate the effects of the selected outputs from subsequent analysis. We calculate this partial correlation by first partitioning the correlation matrix $\boldsymbol{R} = \mathrm{Corr}\left[\boldsymbol{y}\right]$ into the block form

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \boldsymbol{R}_{21} & \boldsymbol{R}_{22} \end{pmatrix},$$

where $\boldsymbol{R}_{11}$ is the matrix of correlations of the identified principal variables, $\boldsymbol{R}_{22}$ is the sub-matrix of correlations of the remaining variables, and $\boldsymbol{R}_{12}$ and $\boldsymbol{R}_{21}$ contain the correlations between the two groups. Given a non-trivial set of PVs, the matrix over which we calculate the H_i is then given by

$$\boldsymbol{R}_{22\cdot 1} = \boldsymbol{R}_{22} - \boldsymbol{R}_{21}\boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12}.$$

The process then iterates until sufficient variables are chosen that the partial variance of each remaining output is small. In general, we obtain large values of H_i when output y_i has, on average, high loadings on important principal components of the correlation matrix and thus corresponds to a structurally important variable. Designs which are informative for all the principal variables will tend to be informative for the remaining variables, through the joint correlation structure.

3.2 Emulating the coarse model

As $F(\boldsymbol{x})$ is multivariate, we will build a collection of univariate emulators. We now describe how to construct a second order emulator for a single output, $y_j \in S$. Our aim is to use this information to construct an emulator $f_0(\boldsymbol{x})$ to express our beliefs about the relationship between F_0 and its model inputs $\boldsymbol{x}$. We express the emulator for the jth component of F_0 as a combination of global and local model effects as follows:

$$f_{0j}(\boldsymbol{x}) = h_{0j}(\boldsymbol{x}_{Aj}) + \epsilon_{0j}(\boldsymbol{x}_{Aj}) + \delta_{0j}(\boldsymbol{x}). \qquad (1)$$

where $h_{0j}(\boldsymbol{x}_{Aj})$ is a global trend function in a subset of the inputs, $\boldsymbol{x}_{Aj}$; $\epsilon_{0j}(\boldsymbol{x}_{Aj})$ is a correlated residual process over the same subset; and $\delta_{0j}(\boldsymbol{x})$ is an uncorrelated nugget residual over all inputs. All three components are assumed to be uncorrelated with each other. For notational simplicity we shall drop the index j for the remainder of the discussion. We shall now consider the nature of $\boldsymbol{x}_A$, $h_0(\boldsymbol{x}_A)$, $\epsilon_0(\boldsymbol{x}_A)$ and $\delta_0(\boldsymbol{x})$ in further detail.

For any particular component of the coarse simulator, we often find that much of the systematic behaviour of $F_0(\boldsymbol{x})$ is driven by a small subset of inputs, known as the *active variables*, $\boldsymbol{x}_A$ (Craig et al., 2001; Goldstein and Rougier, 2007). Since these inputs are so important in determining the value of $F_0(\boldsymbol{x})$, the identification of $\boldsymbol{x}_A$ is a crucial task. Different outputs will have different sets of active inputs. Therefore some model effects will be more important to design for than others, and these effects will change across outputs. In order to determine $\boldsymbol{x}_A$ for a given output, we can either elicit information from an expert about appropriate choices of active inputs or, in the absence of expert knowledge, we could use a data-driven approach. For example, we can use our simulator runs to regress the output of $F_0(\boldsymbol{x})$ onto a linear model of main effects in all inputs and then stepwise delete all unimportant terms to leave a model in a reduced subset of inputs that we take to be $\boldsymbol{x}_A$. The collection of active variables across all outputs, $B_A = \cup_i \boldsymbol{x}_{Ai}$, is of particular importance when we consider designing for the collection of univariate emulators.

In the emulator, we represent the global variation in $F_0(\boldsymbol{x})$ by a trend function in the active variables, $h_0(\boldsymbol{x}_A)$. Since the number of runs of the fine simulator will be limited and the coverage of input space will be poor, the global trend will principally determine the value of the emulator as any design will only be weakly informative for the residual process at input points away from the chosen fine simulator runs. We express the emulator trend as a linear combination of appropriate basis functions in the active variables with unknown coefficients

$$h_0(\boldsymbol{x}_A) = \sum_{i=1}^{p} \beta_{0i} g_i(\boldsymbol{x}_A). \tag{2}$$

For example, we may consider the $g_i(\cdot)$ to be simple monomial terms in the active variables, the choice of which could be elicited from an expert or determined by a backward stepwise selection approach starting with all feasible terms in the active variables. Our method is not tied to any specific form of $g_i(\cdot)$, so we could use any alternative form such as orthogonal polynomials, or more general basis function such as those found via generic screening methods (Welch, Buck, Sacks, Wynn, Mitchell, and Morris, 1992). To determine appropriate values for the means and variances of the β_{0i}, we could elicit expert prior values and then adjust those prior values by the coarse model runs via the appropriate Bayes linear update (Goldstein and Wooff, 2007). Without available expert knowledge and since we have a large number of coarse function evaluations, we can simply fit the trend function via, for example, least squares to obtain appropriate estimates.

The residual process $\epsilon_0(\boldsymbol{x}_A)$ is a weakly stationary process which describes the portion of the difference between $F_0(\boldsymbol{x})$ and $h_0(\boldsymbol{x}_A)$ which is dependent on $\boldsymbol{x}_A$ but is not captured by h_0. We allow ϵ_0 to be correlated across the input space as it represents additional systematic variation in F_0 that is unaccounted for by h_0, such as additional higher-order model terms in $\boldsymbol{x}_A$. We therefore specify a covariance function over all $(\epsilon_0(\boldsymbol{x}_A), \epsilon_0(\boldsymbol{x}'_A))$ pairs, which has the form $\text{Cov}\left[\epsilon_0(\boldsymbol{x}_A), \epsilon_0(\boldsymbol{x}'_A)\right] = (1-\kappa_\delta)\sigma^2 R(\boldsymbol{x}_A, \boldsymbol{x}'_A; \theta)$ where σ^2 is the variance of the emulator residuals, κ_δ controls the relative size of the contribution of the nugget effect, θ is the correlation length parameter, and $R(\cdot,\cdot)$ is an appropriate correlation function. In our case, we use the Gaussian correlation function $R(\boldsymbol{x}_A, \boldsymbol{x}'_A; \theta) = \exp(-\theta||\boldsymbol{x}_A - \boldsymbol{x}'_A||^2)$. We assess the parameters of this correlation function, for example by extracting the emulator trend's residuals and then applying variogram techniques (Cressie, 1991) to determine values for θ, κ_δ, and σ^2. There is an extensive literature on appropriate choices for the correlation function, for example see Santner et al. (2003, Chapter 2). The design approach that we shall describe applies equally to all choices of correlation form.

The final term $\delta_0(\boldsymbol{x})$ is a nugget residual with variance $\kappa_\delta\sigma^2$ which describes all the remaining variation in F_0 that cannot be explained by $\boldsymbol{x}_A$ via the trend, h_0, or the structured residuals, ϵ_0, such as variation due to the effect of inactive variables or potential interactions between active and inactive variables. The information contained in δ_0 is weak compared to that of h_0 or ϵ_0, so we therefore choose to simply treat δ_0 as unstructured random noise. This is reasonable, particularly at the design stages, as there is no discernible structure within the remaining variables that we can exploit to guide the design and, furthermore, κ_δ is typically small.

From (1) and (2) we now write the emulator for a particular component of $F_0(\boldsymbol{x})$ as

$$f_0(\boldsymbol{x}) = \sum_{i=0}^{p} \beta_{0i} g_i(\boldsymbol{x}_A) + r_0(\boldsymbol{x}), \tag{3}$$

where we set $g_0 = 1$ to include a constant term in the global trend, and we define $r_0(\boldsymbol{x})$ as the sum of the residual terms

$$r_0(\boldsymbol{x}) = \epsilon_0(\boldsymbol{x}_A) + \delta_0(\boldsymbol{x}). \tag{4}$$

Since we have a large batch of runs available on F_0, we may eliminate most of the uncertainty in the values of the coefficients $\boldsymbol{\beta}_0$ and any remaining uncertainty will be of a considerably smaller magnitude than that of the uncertainties attached to our fine emulator. We therefore make the simplifying assumption that $\boldsymbol{\beta}_0$ is known. Thus, for any $\boldsymbol{x}$ with $F_0(\boldsymbol{x})$ known, i.e. evaluated, we may also consider $r_0(\boldsymbol{x})$ as known. Under this assumption, we can, without loss of generality, absorb the values of $\boldsymbol{\beta}_0$ into the basis functions themselves. This assumption greatly simplifies our description of the subsequent analysis by eliminating any variation in the model coefficients from the calculations. However, if we are only allowed a medium number of runs on the coarse simulator, then we will add additional variance terms to our subsequent calculations to express our uncertainty about each coefficient value. As our design calculations are constructed from the joint variance structure, this will not affect our general design methodology.

By treating $\boldsymbol{\beta}_0$ as known and absorbing its values into $g_i(\boldsymbol{x}_A)$, we can write our coarse emulator as

$$f_0(\boldsymbol{x}) = \sum_{i=0}^{p} g_i(\boldsymbol{x}_A) + r_0(\boldsymbol{x}). \tag{5}$$

3.3 Linking the coarse and fine models

We suppose that the coarse simulator is sufficiently informative for the fine simulator that it provides the basis for building an informative prior distribution. We now develop an appropriate form for $f_1(\boldsymbol{x})$, the emulator for a component of the fine simulator $F_1(\boldsymbol{x})$. If we consider that each of the terms in the global trend of the emulator represents a qualitative physical effect, then these effects may change differentially when the model changes resolution. Therefore, we allow the contribution of each basis function in $h_0(\boldsymbol{x}_A)$ to the fine emulator to be scaled individually via its own parameter. Thus we use the following form for the fine emulator

$$f_1(\boldsymbol{x}) = \sum_{i=0}^{p} \rho_i g_i(\boldsymbol{x}_A) + \rho_{p+1} r_0(\boldsymbol{x}) + r_1(\boldsymbol{x}), \tag{6}$$

where we build the fine emulator on the same set of active variables and basis functions as the coarse emulator, giving an identical structure to the global trend component. However, the relative magnitudes of the elements can differ from the coarse emulator in a way controlled by the multipliers ρ_i, which are independent of $r_0(\boldsymbol{x})$ and $r_1(\boldsymbol{x})$. Additionally, $r_1(\boldsymbol{x})$ is a new residual process unique to the fine emulator which absorbs any additional structure in the active variables or any additional effects attributable to variables not previously deemed active.

An alternative form for $f_1(\boldsymbol{x})$ is that of Kennedy and O'Hagan (2000), where we link the fine emulator directly to the coarse simulator

$$f_1(\boldsymbol{x}) = \rho F_0(\boldsymbol{x}) + r_1(\boldsymbol{x}), \tag{7}$$

where ρ is a single scaling factor whose value is to be determined. This construction is a special case of (6) obtained by setting $\rho_i = \rho$, and $\text{Corr}\,[\rho_i, \rho_j] = 1$ $\forall i, j$. However, the emulator form (6) allows for more prior flexibility by not forcing a single multiplier across the components of the fine emulator. This single multiplier approach is generalised in Qian, Seepersad, Joseph, Allen, and Wu (2006), and Qian and Wu (2008) by introducing a dependence of ρ on $\boldsymbol{x}$ such that

$$f_1(\boldsymbol{x}) = \rho(\boldsymbol{x}) F_0(\boldsymbol{x}) + r_1(\boldsymbol{x}), \tag{8}$$

where the form of $\rho(\boldsymbol{x})$ is a linear relationship or a Gaussian process, thus allowing the scaling factor to change throughout the input space. Unlike (7), this formulation does not directly correspond to our framework. However, if $\rho(\boldsymbol{x})$ was expressed as a linear regression of basis functions in $\boldsymbol{x}_A$ with unknown coefficients, then the emulator form would not be dissimilar to that of (6), and so our design methodology would be applicable.

When considering the relationship between the two models, it is preferable to consider expressions in the model differences rather than the coarse and fine emulators directly We write this difference between the models as

$$\begin{aligned} d(\boldsymbol{x}) &= f_1(\boldsymbol{x}) - f_0(\boldsymbol{x}) \\ &= \sum_{i=0}^{p} \rho_i^* g_i(\boldsymbol{x}_A) + \rho_{p+1}^* r_0(\boldsymbol{x}) + r_1(\boldsymbol{x}), \end{aligned} \tag{9}$$

where $\rho_i^* = \rho_i - 1$. Thus ρ_0^* is the constant 'offset' of the fine model from the coarse, and ρ_{p+1}^* governs the magnitude of the contribution of the coarse residuals.

In order to obtain a useable prior fine emulator from equation (9), we must make appropriate belief specifications for the multipliers $\boldsymbol{\rho}^* = (\rho_0^*, \ldots, \rho_{p+1}^*)$ and the residual process $r_1(\boldsymbol{x})$. In general, our uncertainty judgements about all of these quantities will be problem-specific. However, our design methodology will apply equally to all specifications of this general form. We describe here the simplest informative form that these judgements might take. First, we choose the expectation and variance of $r_1(\boldsymbol{x})$ to be

$$\mathrm{E}\left[r_1(\boldsymbol{x})\right] = 0, \tag{10}$$

$$\mathrm{Var}\left[r_1(\boldsymbol{x})\right] = u. \tag{11}$$

Next, we assign prior expectation

$$\mathrm{E}\left[\boldsymbol{\rho}^*\right] = 0, \tag{12}$$

which is equivalent to $\mathrm{E}\left[\boldsymbol{\rho}\right] = 1$ and corresponds to a judgement that there are no systematic biases between the models known *a priori*. We choose to specify the covariance matrix of $\boldsymbol{\rho}^*$ simply via two constants, v and c such that

$$\begin{aligned} \mathrm{Var}\left[\rho_i^*\right] &= v && (13) \\ \mathrm{Cov}\left[\rho_i^*, \rho_j^*\right] &= c, \ i \neq j. && (14) \end{aligned}$$

Whilst this is a simple form for $\mathrm{Var}\left[\boldsymbol{\rho}^*\right]$, by changing the value of c we can move from prior beliefs that the differences between the two emulators are independent across components of the global trend, to a model of perfect correlation having only a single effective multiplier corresponding to equation (7). Of course, we will use a more complex belief specification if the application requires.

The belief specifications in (13) and (14) induce the following prior expectations for $d(\boldsymbol{x})$ for which $F_0(\boldsymbol{x})$ is known, i.e. $\boldsymbol{x} \in \boldsymbol{X}_0$, namely

$$\begin{aligned} \mathrm{E}\left[d(\boldsymbol{x})\right] &= 0, && (15) \\ \mathrm{Var}\left[d(\boldsymbol{x})\right] &= \boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{V} \boldsymbol{g}(\boldsymbol{x}) + u, && (16) \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{g}(\boldsymbol{x}) &= (g_0(\boldsymbol{x}), g_1(\boldsymbol{x}), \ldots, g_p(\boldsymbol{x}), g_{p+1}(\boldsymbol{x})) \\ &= (g_0, g_1(\boldsymbol{x}_A), \ldots, g_p(\boldsymbol{x}_A), r_0(\boldsymbol{x}))\,, && (17) \end{aligned}$$

and $\boldsymbol{V} = \mathrm{Var}\left[\boldsymbol{\rho}^*\right]$. We can expand the variance expression given in (16) into the form

$$\mathrm{Var}\left[d(\boldsymbol{x})\right] = v s(\boldsymbol{x}) + c t(\boldsymbol{x}) + u, \tag{18}$$

where $s(\boldsymbol{x}) = \sum_{i=0}^{p+1} g_i(\boldsymbol{x})^2$ and $t(\boldsymbol{x}) = \sum_{i \neq j} g_i(\boldsymbol{x}) g_j(\boldsymbol{x})$. Thus by specifying the prior values v, c, and u, we can describe the behaviour of the difference between the two models. Our beliefs about these parameters will be updated as we make evaluations of the fine simulator.

3.4 Updating the fine emulator

We have now obtained an emulator of our coarse computer model, and specified a second-order belief framework to link the coarse emulator to the fine emulator. The next stage of analysis is to design runs for the fine computer model by using the information contained within our prior fine emulator – this is discussed in Sections 4 and 5.

Having obtained an appropriate design over the simulator inputs, we then evaluate the fine simulator at these design points. Since the coarse model, F_0, has a negligible associated evaluation cost, we also run this model at the same design points in order to obtain the difference values, $d(\boldsymbol{x})$, for each $\boldsymbol{x}$ in the design. We then update the prior fine emulator, $f_1(\boldsymbol{x})$, by the collection of simulator differences. As we are treating $\boldsymbol{\beta}_0$ as known, we may also consider the values of $r_0(\boldsymbol{x})$ as known for each design point thus simplifying the update calculations.

Since our beliefs about $F_1(\boldsymbol{x})$ are expressed via a second order emulator, $f_1(\boldsymbol{x})$, an appropriate manner in which to update these beliefs is via Bayes linear adjustment (for a full description of the Bayes linear approach, see Goldstein and Wooff, 2007). Having observed a random vector D, then the adjusted expectation and variance for the random vector B are given by

$$\begin{aligned} \mathrm{E}_D[B] &= \mathrm{E}[B] + \mathrm{Cov}[B,D]\,\mathrm{Var}[D]^{-1}(D - \mathrm{E}[D]), \qquad (19) \\ \mathrm{Var}_D[B] &= \mathrm{Var}[B] - \mathrm{Cov}[B,D]\,\mathrm{Var}[D]^{-1}\,\mathrm{Cov}[D,B]. \qquad (20) \end{aligned}$$

We write the differences between the outputs of the two simulators as $\boldsymbol{D} = \boldsymbol{F}_1 - \boldsymbol{F}_0$, where $\boldsymbol{F}_0$ and $\boldsymbol{F}_1$ are the results of evaluating the coarse and fine simulators over the design. Applying the Bayes linear adjustment formulae, we obtain the following general expressions for the adjusted expectation and variance of $f_1(\boldsymbol{x})$ given the simulator differences, $\boldsymbol{D}$

$$\begin{aligned} \mathrm{E}_{\boldsymbol{D}}[f_1(\boldsymbol{x})] &= \boldsymbol{g}(\boldsymbol{x})^T \mathrm{E}_{\boldsymbol{D}}[\boldsymbol{\beta}_0] + \mathrm{E}_{\boldsymbol{D}}[r_0(\boldsymbol{x})] + \mathrm{E}_{\boldsymbol{D}}[d(\boldsymbol{x})], \qquad (21) \\ \mathrm{Var}_{\boldsymbol{D}}[f_1(\boldsymbol{x})] &= \boldsymbol{g}(\boldsymbol{x})^T \mathrm{Var}_{\boldsymbol{D}}[\boldsymbol{\beta}_0]\,\boldsymbol{g}(\boldsymbol{x}) \\ &\quad + \mathrm{Var}_{\boldsymbol{D}}[r_0(\boldsymbol{x})] + \mathrm{Var}_{\boldsymbol{D}}[d(\boldsymbol{x})]. \qquad (22) \end{aligned}$$

As we are supposing that we have enough runs on the coarse simulator so that $\boldsymbol{\beta}_0$ can be treated as known, then at design points, $\boldsymbol{x}$, at which we have already evaluated the simulator the uncertainty in $\boldsymbol{\beta}_0$ and in $r_0(\boldsymbol{x})$ disappears and these expressions reduce to

$$\mathrm{E}_{\boldsymbol{D}}\left[f_1(\boldsymbol{x})\right] = F_0(\boldsymbol{x}) + \mathrm{E}_{\boldsymbol{D}}\left[d(\boldsymbol{x})\right], \tag{23}$$

$$\mathrm{Var}_{\boldsymbol{D}}\left[f_1(\boldsymbol{x})\right] = \mathrm{Var}_{\boldsymbol{D}}\left[d(\boldsymbol{x})\right]. \tag{24}$$

Therefore, in this case, to find the adjusted expectation and variance of $f_1(\boldsymbol{x})$ we need only calculate $\mathrm{E}_{\boldsymbol{D}}\left[d(\boldsymbol{x})\right]$ and $\mathrm{Var}_{\boldsymbol{D}}\left[d(\boldsymbol{x})\right]$ which we obtain from application of (19) and (20). The components of those expressions either correspond directly to prior belief statements or are determined by using the specifications and judgements detailed in Sections 3.2 and 3.3. If there still remains some uncertainty attached to the coefficients $\boldsymbol{\beta}_0$, then we would instead adjust the coarse emulator, $f_0(\boldsymbol{x})$, by the additional coarse model evaluations in order to obtain estimated values for $r_0(\boldsymbol{x})$ at each of the design points.

4 The structure of the design problem

Since the fine computer simulator, F_1, is costly to evaluate, practical constraints may permit only a small number of runs. We must make as full use of the designed model evaluations as possible by fitting *all* of the emulators using the *same* batch of runs. By exploiting the active variable structure of the emulators to break the design problem into several smaller problems, we can construct efficient small sample designs for this purpose.

4.1 Decomposing the design problem

Through screening of the outputs, we have now emulated the reduced collection of selected outputs, S. For example, suppose that we have a coarse simulator $F(\boldsymbol{x})$ with two outputs, y_a and y_b say, with emulators $f_a(\boldsymbol{x})$ and $f_b(\boldsymbol{x})$ and we have emulators $f_a(\boldsymbol{x}) = h_a(x_1, x_2, x_3, x_4) + r_a(\boldsymbol{x})$ and $f_b(\boldsymbol{x}) = h_b(x_1, x_2, x_5, x_6) + r_b(\boldsymbol{x})$. Thus $\{x_1, x_2, x_3, x_4\}$ and $\{x_1, x_2, x_5, x_6\}$ are the active variable sets for f_a and f_b respectively.

To design runs for F_1 in the absence of any structural information, we would design over the entire space of active inputs, i.e. $\{x_1, x_2, x_3, x_4, x_5, x_6\}$. However, the structure exposed by the sets of active variables shows that the design decomposes into the smaller active subspaces. For the example, this suggests designing over two 4-dimensional spaces rather than one 6-dimensional space. Reducing the dimensionality of the design space in this manner is beneficial when we have a restricted number of runs as we can obtain better coverage of lower-dimensional spaces given a fixed design size, and we can more easily search over lower-dimensional spaces to identify efficient designs.

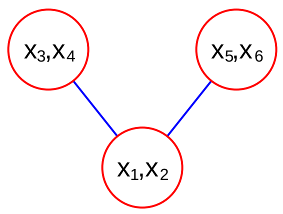
By further examination of the active sets $\{x_1, x_2, x_3, x_4\}$ and $\{x_1, x_2, x_5, x_6\}$, we see that the inputs $\{x_1, x_2\}$ are active in both emulators, whilst $\{x_3, x_4\}$ and $\{x_5, x_6\}$ are unique to f_a and f_b respectively. This means that the inputs $\{x_1, x_2\}$ may have interactions with any of the other four inputs to the computer model, but $\{x_3, x_4\}$ and $\{x_5, x_6\}$ have their space of possible interactions separated. Therefore, for any suggested set of design choices for $\{x_1, x_2\}$, we can consider the efficient selection of design choices for $\{x_3, x_4\}$ as a separate problem to that of the efficient selection for $\{x_5, x_6\}$.

We now define some notation and terminology for these different sets of input variables. First, we define B_A and B_U as the sets of active and inactive inputs across all outputs. Using the terminology of Bates, Buck, Riccomagno, and Wynn (1996), the set B_A is then further decomposed into a set of *border* inputs, B_0, which can potentially interact freely with any other active input variables, and a number of sets of *block* variables, B_i for $i = 1, \ldots, b$, which can potentially interact only with inputs within the same block and with the inputs in the border. Thus the sets B_0, B_i, and B_U are disjoint and exhaustive and thus partition the set of input variables. In the case of our example, we have $B_0 = \{x_1, x_2\}$, $B_1 = \{x_3, x_4\}$, $B_2 = \{x_5, x_6\}$ and $B_U = \emptyset$. These relationships can be depicted in a graph such as that in Figure 4.1(b), where input variables are connected if they appear in the same active set.

Of course, with such a small example it is trivial to identify which variables form the border and which constitute the blocks. With larger-scale problems, involving tens or hundreds of variables, making such an identification by eye be-

$$\begin{aligned} f_a(\boldsymbol{x}) &= h_a(x_1, x_2, x_3, x_4) + r_a(\boldsymbol{x}) \\ f_b(\boldsymbol{x}) &= h_b(x_1, x_2, x_5, x_6) + r_b(\boldsymbol{x}) \end{aligned}$$

(a) Emulators



(b) Interaction graph of inputs

$$\left(\begin{array}{cc|cc|cc} x_1 & 1 & 1 & 1 & 1 & 1 \\ 1 & x_2 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & x_3 & 1 & 0 & 0 \\ 1 & 1 & 1 & x_4 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & x_5 & 1 \\ 1 & 1 & 0 & 0 & 1 & x_6 \end{array}\right)$$

(c) Matrix representation

Figure 1: Decomposition of active variable structure within the emulators

comes substantially more complex. However, methods such as the algorithm of Zečevič and Šiljak (1994) can determine the necessary partitions automatically. The algorithm operates on the adjacency matrix of the graph associated with the active variables and permutes the columns and rows of the matrix until it obtains a bordered block diagonal structure (see Figure 1(c)) – the variable partitions into border and blocks can then be directly read off the permuted matrix. For large sets of inputs, block sizes may contain tens or hundreds of variables and we may repeat the decomposition algorithm on each block to further break it into a border and sub-blocks, and so forth.

4.2 Generating the design

Our aim is to design a set of n runs of the fine simulator by selecting input choices $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ at which to evaluate the fine simulator, $F_1(\boldsymbol{x})$. We partition the inputs into sets of active, B_A, and inactive inputs, B_U, via the methods described in Section 3.2 and obtain a border-block partition over B_A as described above. We then generate the design by combining an informed design over the active variables, B_A, with an appropriate space-filling design over the inactive inputs, B_U. We compare alternative choices of design for B_A with respect to a design criterion $C(\cdot)$. The choice of $C(\cdot)$ depends upon the purpose of the design and the future goals of the analysis, and each criterion is the sum of the expected gain in information, appropriately defined, for each selected output on the fine simulator. Different design criteria and their formulation are discussed in detail in Section 5.

The design for the active variables, B_A, is constructed by exploiting the border-block structure discussed above. By partitioning the active variable space in this way, we can construct the larger design by combining smaller sub-designs over the input subsets determined by the border and blocks. In the example, we decomposed the input space into three subspaces $\{x_1, x_2\}$, $\{x_3, x_4\}$ and $\{x_5, x_6\}$. Therefore, we will construct an appropriate design for each of these 2-variable spaces and combine them to form our final design. In order for this method to work, we need to be able to construct larger designs by augmenting an existing design with additional columns. For this purpose, Latin hypercubes are useful since they can be augmented in such a way and yet still preserve their properties.

To begin the process, we construct an initial design for the border variables. In our example, this is the set $B_0 = \{x_1, x_2\}$. Once we have fixed our design in the border inputs, we can consider the designs for the each of the sets of block variables independently. For our initial design, we generate a large number of possible Latin hypercube designs over B_0 and then choose the design which maximises the minimum interpoint distance.

Having fixed a border design, we next design over each block, B_i. Since we can design over each B_i separately once the design over B_0 is fixed, the order in which we consider the B_i is unimportant. Given a design for B_0, any design over B_i will provide sufficient information to determine the expected information, according to criterion $C(\cdot)$, for the selected outputs corresponding to the active inputs in $B_i \cup B_0$. In the example, having fixed a design over $B_0 = \{x_1, x_2\}$ then for any choice of design over $B_1 = \{x_3, x_4\}$ we can determine the gain in information for $f_a(\boldsymbol{x})$. Therefore, the value of the design over B_i can be evaluated with respect to criterion $C(\cdot)$. Adopting a similar strategy as with the border design, for each block B_i we generate a number of Latin hypercube candidate designs over B_i, each of which is then combined with our chosen design for B_0 and evaluated with respect to $C(\cdot)$. We then choose the design which optimises this criterion value. This process is then repeated for each of the remaining blocks.

Having designed over all the active inputs, we then iteratively refine the design to improve its overall performance on the design criterion. For each of the sub-designs over B_0 and the B_i, we generate a large number of new potential candidate sub-designs by the above methods. We then consider the effect of replacing the existing sub-design with any of the candidates. If the replacement results in an improvement in performance with respect to $C(\cdot)$, then we accept the change. We move through all of the design subspaces, each time seeking preferable sub-designs, with each iteration attempting to swap out the sub-designs in pursuit of a better overall design. The iterative refinement stops when our search procedures result in little or no further improvement in $C(\cdot)$. If each block has been further reduced into a sub-border and sub-blocks, then the design optimisation for each block is also structured according to this procedure.

5 Tuning, calibration and forecasting

In Section 2, we introduced the four stages of our general analysis: emulation, tuning, calibration and forecasting. We now consider the tuning, calibration and forecasting stages in more detail. Depending on the nature of the problem, the emphasis we place on these stages will vary. The more uncertain that we are that the fine and the coarse simulators obey useful relationships, then the larger must be our budget for simulator evaluations in the tuning phase. The larger the size of the input space, and the more extensive the quantity of potential calibration data, then the greater will be the value of carrying out a careful calibration stage to reduce the size of the input space that we must consider. This is particularly important if some key outputs are hard to emulate even on the coarse simulator, over the original input space. The final stage depends on the principle objectives of the analysis. We choose forecasting, simply as this is a common purpose for model analysis. In all cases, our objective is the same – to transfer the information that we have gathered from the coarse simulator into a meaningful prior emulator on the fine simulator using as few evaluations of the fine simulator as possible. This economy is intended to allow as many evaluations as possible for exploring the features of the behaviour of the fine simulator which are not mirrored in the coarse simulator.

5.1 Tuning

We must connect the emulator of the coarse simulator (about which we now know a great deal) to the emulator of the fine simulator (about which we know very little). The purpose of the tuning stage is to obtain some preliminary information about the change in behaviour which occurs when we move from the coarse simulator to the fine. The tuning stage uses runs of both simulators over an appropriate design to obtain a collection of model differences. These differences are then used to tune our prior beliefs about the multilevel framework parameters of Section 3.3, and hence our prior beliefs about $f_1(\boldsymbol{x})$. Additionally, the differences are used in a data analysis stage where we examine the changes in simulator means and variances as we change resolution to determine whether a multilevel emulation is reasonable or appropriate for each output. This second task is important, but difficult to construct small sample designs for and hence our choice of design for the tuning stage relates to the first task.

5.1.1 Tuning beliefs about multilevel parameters

The formal purpose of the tuning stage is to learn about the multilevel parameters u, v, and c as defined in (11), (13), and (14) for each of the selected outputs. When designing the tuning runs, we design for all outputs that we wish to study in later stages of the analysis, to ensure we have good tuning information for each. We therefore require a design criterion appropriate to tuning beliefs about the multilevel parameters, such as minimising a function of their variance. Suppose we choose n design points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, which we evaluate on both F_1 and F_0. We can then calculate the simulator differences $\{d_1, \ldots, d_n\}$ where $d_i = F_1(\boldsymbol{x}_i) - F_0(\boldsymbol{x}_i)$, for each of the selected outputs. Defining the squared differences as $w_i = d_i^2$, then we have

$$\mathrm{E}\left[w_i\right] = v s_i + c t_i + u, \tag{25}$$

where $s_i = \sum_{j=0}^{p+1} g_j(\boldsymbol{x}_i)^2$ and $t_i = \sum_{j \neq k} g_j(\boldsymbol{x}_i) g_k(\boldsymbol{x}_i)$ which corresponds directly to (18).

Two approaches to tackling the tuning problem are to use either weighted least squares or perform a Bayes linear variance adjustment. Since these two methods are quite different, each is associated with its own particular design criterion.

Weighted Least Squares Tuning Given the observed values of w_i and their corresponding input values, then tuning our beliefs about the multilevel parameters reduces to the linear regression problem of (25) in three unknowns with unknown coefficients $\boldsymbol{v} = (u, c, v)$. We can therefore obtain estimates for $\boldsymbol{v}$ via weighted least squares, weighting each observation by $1/\mathrm{Var}\left[w_i\right]$ and thus obtain weighted least squares estimates $\hat{\boldsymbol{v}}$ for the tuning parameters. We therefore seek a design that will reduce as much as possible the variance in our estimates for $\boldsymbol{v}$, giving a criterion of the form

$$C_{\mathrm{WLS}} = \mathrm{tr}\left(\mathrm{Var}\left[\hat{\boldsymbol{v}}\right]\right) = \mathrm{tr}\left(\boldsymbol{A}\boldsymbol{W}\boldsymbol{A}^T\right), \tag{26}$$

averaged over all outputs, where $\hat{\boldsymbol{v}}$ are the estimates of $\boldsymbol{v}$, $\boldsymbol{W}$ is the diagonal matrix of weights, $\boldsymbol{A} = (\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^T)^{-1}\boldsymbol{V}\boldsymbol{W}$, and $\boldsymbol{V}$ is the model matrix with ith row equal to $(1, s_i, t_i)$.

Bayes Linear Tuning If we made a second order prior specification for $\boldsymbol{v}$ then we can use the observations w_i to perform a Bayes linear adjustment of the prior values for each of the variance terms (Goldstein and Wooff, 2007, Chapter 8).

An appropriate criterion is then the resolution of the Bayes linear adjustment (Goldstein and Wooff, 2007) of the $\boldsymbol{v}$ by the observed squared differences w_i. The resolution is a scale-free measure which quantifies the magnitude of the effect of an adjustment in reducing our uncertainties; its value lies in the interval $[0, 1]$ where larger values indicate a greater proportion of the uncertainties have been resolved. The resolution of the adjustment of $\boldsymbol{X}$ by $\boldsymbol{D}$ is defined as

$$R_{\boldsymbol{D}}(\boldsymbol{X}) = \frac{1}{r_{\boldsymbol{X}}} \mathrm{tr}\left(\mathrm{Var}\left[\boldsymbol{X}\right]^{-1} \mathrm{Cov}\left[\boldsymbol{X}, \boldsymbol{D}\right] \mathrm{Var}\left[\boldsymbol{D}\right]^{-1} \mathrm{Cov}\left[\boldsymbol{D}, \boldsymbol{X}\right]\right) \tag{27}$$

where $r_{\boldsymbol{X}}$ is the rank of the matrix $\mathrm{Var}\left[\boldsymbol{X}\right]$. Thus, our design criterion is

$$C_{\mathrm{BL}} = R_{\boldsymbol{w}}(\boldsymbol{v}), \tag{28}$$

averaged over all chosen outputs, where $\boldsymbol{w}$ is the vector of the w_i.

5.1.2 Tuning data analysis

In addition to tuning our beliefs about $\boldsymbol{v}$, the model differences also provide us with information about how the behaviour of the simulator outputs changes as we move from the coarse simulator to the fine. Since we have few simulator runs on the fine simulator, it will be hard to draw detailed conclusions. However a simple comparison of the outputs of the two simulators may prove insightful.

In particular, we seek outputs whose variances change substantially when we move between the coarse and fine simulators. These outputs will not lend themselves to a simple multilevel emulation treatment since their behaviour changes across the two simulators, making it difficult to use $F_0(\boldsymbol{x})$ to construct informative designs for $F_1(\boldsymbol{x})$. Any such variables will be screened out of the subsequent analysis. If emulators for those quantities are required then they will have to be built individually on the separate simulators. We carry out a further data analysis after each batch of evaluations of the fine simulator is obtained, to assess further the appropriateness of the prior specification.

5.2 Calibration and forecasting

5.2.1 Calibration

Calibration is used to reduce the size of the input space, by eliminating input parameter combinations which lead to simulator output that is far from the observed behaviour of the physical system. Using the methods of Craig, Goldstein, Seheult, and Smith (1997), given observational data, $\boldsymbol{z}$, we can evaluate the "implausibility" of a particular input parameter combination, $\boldsymbol{x}$. We say that $\boldsymbol{x}$ is implausible if we judge that it is unlikely that the value of $F_1(\boldsymbol{x})$ will be sufficiently close to the observed data, $\boldsymbol{z}$, to constitute an acceptable match to the actual physical system, as assessed using

$$\mathcal{I}(\boldsymbol{x}) = \frac{(\mathrm{E}_{\boldsymbol{D}}\left[F_1(\boldsymbol{x})\right] - \boldsymbol{z})^2}{\mathrm{Var}_{\boldsymbol{D}}\left[F_1(\boldsymbol{x}) - \boldsymbol{z}\right]}. \tag{29}$$

$\mathcal{I}(\boldsymbol{x})$ can be calculated using the emulator of $F_1(\boldsymbol{x})$. Large values of $\mathcal{I}(\boldsymbol{x})$ identify regions of the parameter space for which good emulation is unlikely to be important.

In order to identify and eliminate regions of implausible matches, we construct a grid over the input space and evaluate the implausibility function (29) at each point in that grid. When we have identified and removed regions of high implausibility, we re-focus our analysis by re-fitting our emulators over this reduced region. Thus the goal of a design for calibration is to reduce the variance associated with the fine emulator of those outputs which correspond to historical data in order that the calibration stage may be effective in reducing the parameter space, and so improve our ability to emulate outputs which will be of interest at the forecasting stage.

5.2.2 Forecasting

There are many approaches to forecasting using computer models, including Bayes linear (Craig et al., 2001; Goldstein and Rougier, 2006) or full Bayes methods (Currin et al., 1991; Oakley and O'Hagan, 2002; O'Hagan, 2006). Each approach starts with a careful emulation of the functional output corresponding to the quantities that are to be forecast. Irrespective of the method used in forecasting, the goal of the design for the forecasting stage is to reduce as much uncertainty as possible about the fine emulator, for each output component which is relevant to the forecast.

5.2.3 Design criterion

For the calibration and forecasting stages of the analysis, we seek to augment the tuning design that we have already obtained with additional design points that will be appropriate for learning as much as possible about the corresponding collection of emulators. At the calibration stage, we will design for all selected outputs for which we have observational data. For the forecasting stage, we will design for those outputs which are important to us to forecast. We seek a design which has the greatest effect in reducing uncertainty about $F_1(\boldsymbol{x})$. We can assess our uncertainty in $F_1(\boldsymbol{x})$ by the adjusted variance of the fine emulator given the simulator evaluations. This variance does not depend on the values of the model evaluations, and follows from (24). For a given choice of inputs, $\boldsymbol{x} \in \boldsymbol{X}_0$ at which $F_0(\boldsymbol{x})$ is known we write

$$\begin{aligned}\mathrm{Var}_{\boldsymbol{D}}\left[f_1(\boldsymbol{x})\right] &= \boldsymbol{g}(\boldsymbol{x})^T \mathrm{Var}_{\boldsymbol{D}}\left[\boldsymbol{\rho}^*\right] \boldsymbol{g}(\boldsymbol{x}) \\ &\quad + \mathrm{Var}_{\boldsymbol{D}}\left[r_1(\boldsymbol{x})\right] \\ &\quad + 2\boldsymbol{g}^T \mathrm{Cov}_{\boldsymbol{D}}\left[\boldsymbol{\rho}^*, r_1(\boldsymbol{x})\right], \end{aligned} \tag{30}$$

where $\boldsymbol{g}(\boldsymbol{x})$ is defined in (17), and $\boldsymbol{D}$ is the observed simulator differences. For small-sample designs, it will not be possible to reduce much of the uncertainty associated with $r_1(\boldsymbol{x})$ over most of the input space and most small-sample designs will not differ greatly in their performance at this task. Therefore, we focus our attention on variance reduction for $\boldsymbol{\rho}^*$, and design using a criterion based on

$$\mathrm{Var}_{\boldsymbol{D}}\left[f_1(\boldsymbol{x})\right] \simeq \boldsymbol{g}(\boldsymbol{x})^T \mathrm{Var}_{\boldsymbol{D}}\left[\boldsymbol{\rho}^*\right] \boldsymbol{g}(\boldsymbol{x}). \tag{31}$$

We want to reduce uncertainty in $F_1(\boldsymbol{x})$ across the whole range of the inputs to allow us to calibrate and forecast from the model. Since we are concerned with the variation across the whole input space, we choose a grid of input points which covers the space and evaluate equation (31) over this collection of points. The grid we use for this is the original coarse simulator design, $\boldsymbol{X}_0$, as at these locations $r_0(\boldsymbol{x})$ is a known function. Furthermore, as $\boldsymbol{X}_0$ is a Latin hypercube there will be a single value of $\boldsymbol{x}$ corresponding to any particular choice for $\boldsymbol{x}_A$, and so we obtain a unique value for $r_0(\boldsymbol{x})$ for our calculations. Thus we get an adjusted variance matrix which we collapse into a scalar for use as a design criterion using the trace

$$\begin{aligned} C_{\mathrm{CF}} = \mathrm{tr}\left(\mathrm{Var}_{\boldsymbol{D}}\left[f_1(\boldsymbol{X})\right]\right) &\simeq \mathrm{tr}\left(\boldsymbol{G}^T \mathrm{Var}_{\boldsymbol{D}}\left[\rho^*\right] \boldsymbol{G}\right) \\ &= \mathrm{tr}\left(\mathrm{Var}_{\boldsymbol{D}}\left[\rho^*\right] \boldsymbol{G}^*\right), \end{aligned} \tag{32}$$

where $\boldsymbol{X}$ is the grid of evaluation points, $\boldsymbol{G}$ is the matrix of basis functions corresponding to those input parameters, and $\boldsymbol{G}^* = \boldsymbol{G}\boldsymbol{G}^T$. We favour designs which give the greatest reduction in this criterion.

The criterion (32) is defined univariately for each selected output. To obtain an aggregate score over the collection of all outputs, we sum the design criterion values for each individual output. As an extension, we could consider using a weighting scheme to weight the outputs differentially, to allow for different outputs to have different contributions to the overall combined criterion value. Possible weightings could include the principal variable loadings H_i to reflect the intrinsic importance of the outputs and their representativeness of outputs not included in the design calculation. Other choices could include measures of the quality of available observational data for use in the calibration stage, or utility scores to measure the value of accurate forecasts for each output variable at the forecast stage.

5.3 Overview

A summary of the stages involved in our approach to small sample design for a multi-level model is given in Table 1. The first stage involves the construction of the coarse emulator, $f_0(\boldsymbol{x})$. Since the coarse simulator is cheap to evaluate we perform a large batch of simulator runs. We then use this information to

Table 1: Summary of the strategy for generation of small-sample designs for $F_1(\boldsymbol{x})$

	Stage	Step	Description	Relevant Section
1	Emulation	1	Intensive sampling of $F_0(\boldsymbol{x})$	
		2	Screen outputs	3.1
		3	Construct coarse emulator $f_0(\boldsymbol{x})$ for each selected output	3.2
		4	Build a prior fine emulator $f_1(\boldsymbol{x})$	3.3
2	Tuning	1	Obtain a border-block partition for the input variables to the emulators	4.1
		2	Jointly design a small number of tuning runs for the selected outputs and evaluate on both $F_0(\boldsymbol{x})$ and $F_1(\boldsymbol{x})$	4.2, 5.1
		3	Tune prior judgements about multilevel framework; preliminary data analysis of model differences; screen out infeasible outputs	5.1
3	Calibration	1	Use border-block partition to design runs for outputs in the screening set that are relevant for calibration and evaluate on both $F_0(\boldsymbol{x})$ and $F_1(\boldsymbol{x})$	4.1, 4.2, 5.2
		2	Update each fine emulator after further data analysis of the model differences	3.4
		3	Calibrate the input space by matching the emulators with observed data; use this calibration to reduce the size of the input space; If the input space is substantially reduced, then re-emulate the coarse simulator within the reduced space	5.2
4	Forecasting	1	Use border-block partition to design runs for outputs that are relevant for forecasting and evaluate on both $F_0(\boldsymbol{x})$ and $F_1(\boldsymbol{x})$	4.1, 4.2, 5.2
		2	Update each fine emulator after further data analysis of the model differences	3.4
		3	Further evaluation and updating of the fine simulator to refine $f_1(\boldsymbol{x})$	3.4
		4	Forecast	5.2

identify the most informative outputs, and to determine which input parameters are most important for describing the behaviour of each simulator output – the *active variables*. Each output is then emulated univariately, and used to construct a prior emulator for that quantity on the fine simulator.

For each of the tuning, calibration and forecasting stages, we require a design for the fine simulator to obtain appropriate batches of evaluations. Given a collection of outputs to be designed for and their associated sets of active variables, we partition the overall active variable set into a *border-block* structure.This allows us to decompose the overall design problem into a number of smaller independent designs which can be constructed iteratively. We then construct these sub-designs to optimise an appropriate criterion. These sub-designs are then combined to the final composite design.

The next three stages of the strategy progressively refine our prior emulator for $F_1(\boldsymbol{x})$. Each stage requires the construction and evaluation of an additional design of simulator runs. The goal of the tuning stage is to investigate the orders of magnitude of the differences between the computer simulators and to ascertain the implications for the relationship between the two emulators, and hence our beliefs about $F_1(\boldsymbol{x})$. Each stage also includes a data analysis on the model differences. The calibration stage tackles the problem of an input space being too large for us to construct a useful small-sample emulator over the whole space. To address this problem, we apply history matching methods which use observational data to reduce the input space. The final stage of the analysis is to design and evaluate simulator runs appropriate to the ultimate goal of the emulation, for example forecasting. We update the fine emulator $f_1(\boldsymbol{x})$ by these model runs and use this updated emulator in any subsequent calculations.

6 Example: Hydrocarbon reservoir model

We illustrate our approach with a simulation of a hydrocarbon reservoir provided for us by Energy SciTech Limited. The model is based on a $48 \times 26 \times 25$ grid, with each grid cell representing a region of subterranean rock with particular geological properties and containing varying proportions of oil, water and gas. The reservoir contains four producing wells which, during the course of the simulation, pump fluids out of the reservoir under certain fixed operating conditions. Similarly there is a single injection well which injects gas into the reservoir to maintain pressure and production levels at the extraction wells.

The outputs of the model are time series of aspects of the well behaviour, including quantities such as pressures, production rates of oil, water and gas, cumulative production levels of oil, water, and gas, and ratio quantities including water cut and gas-oil ratio. For the purposes of this example, we focus on a single time point approximately eighteen months into the operation of the simulation. At this point there are a total of 65 outputs for consideration.

The inputs to the model are a collection of twenty scalar multipliers which rescale various geological properties throughout the reservoir. These multipliers affect quantities including permeabilities, porosity, transmissibilities of the geological faults, critical oil saturation properties, and properties of the reservoir's aquifer.

We have structured our account according to the corresponding stages of Table 1, preserving the numbering in that table..

6.1 Emulation

6.1.1 Sampling of $F_0(\boldsymbol{x})$

To obtain a fast approximation we coarsened the vertical gridding of the model to just two layers, and we used the 25-layer grid for the full simulator. The evaluation time for the coarse simulator was approximately 30 seconds, compared with 35 minutes for the fine simulator. The coarse model was evaluated over a 1500 point Latin hypercube in all inputs to provide us with our initial batch of coarse model runs.

6.1.2 Screening the outputs

The first task is to screen the simulator outputs in order to reduce the output dimension. Applying principal variable selection methods over the coarse simulator evaluations, we identified those variables which accounted for the majority of the variation in the model runs. As there was substantial correlation among the outputs, the principal variable screening was quite effective and the output collection was well-represented on a relatively small subset of the original outputs. Overall, we considered that we could obtain a very good representation of all 65 outputs using a subset of 15, as the proportion of variation explained by our chosen principal variables was at least 0.94 for each remaining output.

The names of the chosen 15 variables are given in the first column of Table 2. The prefixes 'W1', 'W2', 'W3', 'W4' indicate that the variable corresponds to correspondingly numbered production well, whereas 'G1' corresponds to the single gas injector. The quantities measured at each of those wells are given by the variable suffix: 'pre' and 'bhp' correspond to two measures of pressure; 'wct' is the water-cut; 'gor' is the gas-oil ratio; variables such as 'wattot' and 'oilrt' correspond to the total water production, and the rate of oil production respectively; and finally 'gasinjtot' is the total amount of gas injected.

6.1.3 Coarse simulator emulation

The reduced subset of outputs was emulated as discussed in Section 3. Prior to emulation, the designs were scaled so all inputs took the range $[-1, 1]$ and outputs were scaled to have mean 0 and variance 1. The active variables for each emulator were determined via backward selection over all main effects until a satisfactory representation was obtained using three to six input variables. The terms in the final trend component were determined by a stepwise selection from these main effects to include quadratic, cubic and pairwise interaction terms where appropriate. Adding any more active variables to the emulators did not provide any substantial improvement in the R^2 of the OLS fit.

Table 2: Emulation summary for each model output

Output	Active variables	Emulator trend R^2	No. Model Terms	Classification
W4.pre	k_x, k_y, P	0.90	10	$(*)$
W1.oilrt	k_z, P, F_{N1}	0.52	11	$(*)$
W3.bhp	k_x, k_y, k_z, P, F_K	0.75	15	$(-)$
W2.wattot	k_x, k_y, k_z, P, F_{R2}	0.77	18	$(+)$
W3.wct	$k_x, k_y, k_z, P, F_K, F_{R2}$	0.71	24	$(-)$
W2.gor	k_x, k_y, P	0.86	11	$(-)$
W4.wct	k_x, P, F_W	0.85	12	$(+)$
G1.gasinjtot	k_x, k_y, k_z, P	0.93	12	$(*)$
W2.liqrt	k_x, P, F_K	0.70	11	$(*)$
W3.gor	k_x, P, F_K	0.76	9	$(*)$
W2.watrt	k_x, k_y, P	0.70	12	$(+)$
W4.wattot	k_x, k_y, P, F_Y	0.87	15	$(+)$
W4.gor	k_x, k_y, P	0.87	11	$(-)$
W1.oiltot	k_x, k_y, P	0.81	8	$(*)$
W4.oilrt	k_x, P, F_K	0.73	10	$(*)$

A summary of the emulation is given in Table 2. The adequacy of the fit varies between the outputs, but most appear to have a moderate to good level of R^2. One variable in particular fared relatively poorly, namely well 1 oil production rate (W1.oilrt). Further investigation showed that, due to the nature of this variable, we were unable to obtain a better fit using any number or combination of available inputs. The emulation revealed that, of the 20 inputs, only 9 were identified as being active for the outputs under consideration. Those variables deemed inactive included all multipliers pertaining to the aquifer, three multipliers for fault transmissibilities and all multipliers for oil saturation properties.

6.1.4 Prior fine emulation

In order to design for the tuning runs, we construct prior fine emulators of the form given in (6). Therefore, we need to make specifications for the prior expectations of the multilevel parameters v, c, and u (as defined in (11), (14) and (15)). At this stage we have no knowledge about how the two simulators may differ, so we make simple pragmatic choices. For v, we assign the prior expectation to be $\mathrm{E}[v] = 1/4$, which when combined $\mathrm{E}[\rho] = 1$ corresponds to a prior belief that the model coefficients are unlikely to change sign between simulators. We assign a small covariance between the ρ^* by setting $\mathrm{E}[c] = 1/8$, and finally in the absence of any other information, we consider that $\mathrm{Var}[r_1(\boldsymbol{x})] = \mathrm{Var}[r_0(\boldsymbol{x})]$ resulting in $\mathrm{E}[u] = \mathrm{Var}[r_0(\boldsymbol{x})]$.

Since this is an illustrative example rather than an actual case study investigated in real time, a batch of 200 fine model runs were available to allow us to check the validity of the prior emulator form (6). By fitting the coarse emulators to these fine simulator runs, we observed that the coefficients in the model trends do indeed change differentially rather than scaling uniformly (see for example Table 3) indicating that the extra prior flexibility provided by (6) is necessary. It should be noted that the problem of model choice, i.e. determining the active variables and the form of the $g_i(\cdot)$, requires a substantially larger number of simulator runs than for estimation of the coefficients, when the form of model is given. Therefore, we require a large batch of available simulator

Table 3: Coefficients for the emulator trend of Well 2 Liquid Rate fitted to the coarse model runs, and to 200 fine model runs

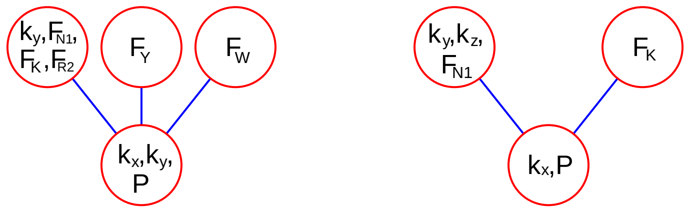
Coefficient	Coarse	Fine
(Intercept)	761.9	-2573.6
k_x	-1689.5	-3183.6
P	2492.5	9524.4
F_K	-109.3	-201.6
k_x^2	1620.9	4869.1
P^2	-3320.3	-6557.1
F_K^2	197.2	517.3
k_x^3	-548.8	-1872.7
P^3	1259.8	1475.1
F_K^3	-107.4	-357.1
$k_x : P$	58.5	-870.3
R^2	0.700	0.836

evaluations to determine the form of the emulators, but far fewer are required to learn about the $\boldsymbol{\beta}$. This is a crucial benefit of the multilevel approach as the coarse simulator provides us with ample information to determine the form of the emulator, whilst the estimation of the $\boldsymbol{\beta}_1$ coefficients can be performed from a smaller batch of carefully chosen fine simulator runs.

6.2 Tuning

6.2.1 Border block partition

Using the active variable information gained from fitting the emulators of the coarse model, we now obtain the border-block partition of the inputs. Applying the factorisation algorithm as discussed in Section 4, we obtain the partition depicted in Figure 2(a). The structure of the emulators is such that there is a three-variable border containing the variables k_x, k_y and P, corresponding to permeability in the x and y direction, and reservoir porosity. The remaining in-



(a) Initial screening (b) Second screening

Figure 2: Border block structure of the input variables (a) after the initial screening, and (b) after the second screening following the tuning runs

puts fall into a larger block containing four variables, and two blocks containing a single variable each. Since the sizes of these subsets are small relative to the number of overall active inputs, there are clear advantages to decomposing the design problem in this way.

6.2.2 Design and evaluate tuning runs

Given the border block partition and the prior fine emulator we now construct a small design for tuning runs. We chose to construct a design of size 6, and followed the general design procedure discussed in Section 4. This produced a design, which yielded the value of 1.128 for the criterion C_{WLS} from (26), where we have used the WLS method for tuning rather than the Bayes linear approach as we are unfamiliar with the physical issues involved in coarsening reservoir simulators so that we do not start with informed prior judgements. The chosen design is presented in Table 4. We are free to make any choices (for example a Latin hypercube) in the remaining input variables. These six parameter combinations were then evaluated on both the coarse and fine computer simulators.

Table 4: The design generated for for the tuning runs

k_x	k_y	k_z	P	F_Y	F_W	F_{N1}	F_K	F_{R2}
0.84	-0.50	-0.63	-0.61	0.17	0.96	-0.61	-0.88	-0.01
0.47	-0.25	1.00	0.27	-0.99	0.32	-0.12	-0.62	0.32
0.28	0.86	0.14	-0.75	-0.66	-0.30	0.19	0.49	0.55
-0.78	-0.93	0.63	0.64	0.76	0.52	0.96	0.76	-0.66
-0.23	0.39	-0.27	1.00	-0.07	-0.62	-0.99	0.33	0.96
-0.59	0.22	-0.77	-0.32	0.47	-0.97	0.55	0.00	-0.95

6.2.3 Tune prior judgements

Using the differences between the two models' output over the tuning design, we are now able to tune our beliefs about the parameters $\{v, c, u\}$. To simplify this account, we obtained guideline estimates for these values via a weighted least squares fit of the model (25), weighting each of the model differences by its prior variance.

At this stage, we now examine our suggested tuned values for $\{v, c, u\}$, and perform preliminary data analysis of the model differences to investigate the behaviour of the two functions. This was particularly insightful as it revealed that the outputs could be divided into three groups. The first group (labelled $(*)$ in Table 2) had coarse-fine differences that were well-behaved and conformed reasonably well to the multilevel model. The second group (labelled $(+)$) consisted of variables which displayed substantially greater variation on the fine simulator than on the coarse simulator – interestingly this group contains outputs which are all concerned with the production of water from the reservoir. The final group (labelled $(-)$) behaved in an opposite manner, having far less variation on the fine simulator than on the coarse model.

For the next stage of the analysis, we chose to screen out the variables in the $(+)$ and $(-)$ groups, and focus on the remainder. The was because, for the eliminated variables, the coarse simulator does not appear to be substantially informative for the fine simulator, and so the multilevel approach does not appear useful at the preliminary design stage. These screened quantities will require further investigation in their own right once the main set of fine model runs have been performed. For the remaining quantities, the new choices for values of $\{v, c, u\}$ were not substantially different from our prior specifications, therefore we shall treat our original choices as acceptable prior values at the next stage. The number of outputs is now further reduced from 15 to 7.

6.3 Calibration and forecasting

The next stage in our approach is to design for and calibrate the model, reducing the size of the input space and re-focussing the analysis on the reduced region. However, from a design perspective, the construction methods are the same for each of the calibration and forecasting stages. Further, the calibration stage, whilst an important part of the general design process, involves a variety of extra considerations (for details of Bayes linear calibration see Craig et al. 1997), which would substantially extend and complicate this account. Therefore, for brevity, we merge the calibration and the forecasting stages, and move directly onto a single design stage where our objective is to reduce variance for all the variables labelled $(*)$ in Table 2.

6.3.1 Design and evaluate forecasting runs

The change in the collection of outputs due to the screening following the tuning stage impacts the composition of the collection of active inputs. We therefore re-perform the border-block partitioning to obtain the new structure shown in Figure 2(b) which now includes only six active inputs.

For the seven principal variables that we have retained, we now develop a design to learn as much as possible about the fine emulators. For each output we seek to to maximise the reduction in the variance via minimising C_{CF} from (32) where the matrix $\boldsymbol{G}$ is constructed from the initial design $\boldsymbol{X}_0$ on the coarse simulator. We construct this design as a combination of the tuning design as

Table 5: Values of design criterion (32) for the second stage design

Additional Design Size	Criterion Value	Additional Design Size	Criterion Value
0	281.783	14	158.555
2	265.926	16	152.096
4	230.370	18	128.495
6	184.194	20	124.495
8	177.374	22	122.691
10	173.225	24	121.306
12	160.174	26	119.383

given in Table 4 (since this has not yet been used to perform an update of the emulator), plus an additional design generated by the border-block methods of Section 4, evaluating the design criterion over the combination of the two designs. For this stage, we constructed several new designs of sizes varying from 2 to 26. To simplify the search process, we proceed by adding two new observations at each stage. This makes it easier to compare different designs and speeds up the iterative search algorithm. As the size of the problem increases, such simplifications will become important.

The value of the design criterion for the new designs is given in Table 5, and the first six points of this additional design are given in Table 6. The criterion value for the tuning design alone is 281.783. Adding extra points to the design and optimising for the appropriate criterion produces a noticeable improvement in design performance, with the performance increasing as the design size increases albeit with generally diminishing returns. As the size of the improvement in design quality tends to level off as the design size increases, if the design budget is very tight for this stage then we might settle for only 6 additional runs. However, if more runs were possible then we might perform a further one or two batches of 6 runs, supporting the formal design calculations with careful data analysis given the results of each batch of runs.

Table 6: The design for the final stage of the analysis

k_x	k_y	k_z	P	F_{N1}	F_K
0.74	0.73	0.46	0.24	0.44	-0.25
-0.36	-0.47	-0.97	-0.07	-0.26	-0.75
-0.80	-0.67	0.25	-0.98	-0.60	0.60
0.08	0.57	-0.36	0.59	0.78	-0.11
-0.91	-0.16	0.97	0.43	-0.79	0.33
0.61	0.16	-0.04	0.83	0.06	1.00

7 Discussion

Learning about complex high dimensional functions with many outputs from limited numbers of evaluations is a very challenging problem. We have presented a method to tackle this problem which exploits fast approximations to the simulator and uses valuable structural information about the model parameters. We have decomposed the problem into a number of stages each of which can be performed in different ways depending on the requirements of the problem. To give an overview of key steps in the process, we have illustrated the methodology with relatively simple belief specifications, design criteria and search methodology to demonstrate feasibility and tractability without becoming too involved in all of the multitude of practical and theoretical issues that will occur in any substantial modelling exercise. It is certainly the case that each of the stages that we have described above may be carried out in a more or less thorough way depending on how carefully we make our belief specifications, how focused we are on particular subsequent practical uses for the analysis, and how much time and computing power we are prepared to expend on seeking efficient designs. These are highly problem specific issues. The modular structure of our approach, and the computational simplifications within the Bayes linear formulation, give plenty of scope for upgrading any individual step which is particularly important in a specific problem. We view the framework that we have suggested as giving a sensible set of guidelines for organising the tasks that we should carry out when planning to use a coarse simulator as the basis for small sample designs, and our suggestions for implementing each stage supply a tractable starting baseline for carrying out each task.

We have restricted our attention to the role of the coarse simulator in creating informative designs. We therefore finish our account at the point at which we have sufficiently exploited information from the coarse simulator that we are ready to move on to the second design stage, namely capturing the information in the full simulator which cannot be approximated by information in the coarse version. We have emphasised the requirement for small sample designs for the first design stage, in our approach, as this releases the maximum budget for the second stage of the design process. This process starts with a diagnostic analysis as to the ways in which our representation breaks down, for example by "leave one out" diagnostics, more detailed comparisons of the covariance structure of predicted against observed outcomes or more open-ended investigations of the information contained in all of the simulator evaluations to that point. Predictive discrepancies, in combination with expert judgement, should form the basis of new exploratory designs which can be used to uncover systematic features that were represented in the first stage of the analysis simply as unstructured variation. The additional structure uncovered on the full simulator may be then investigated, where relevant, on the coarse approximation, leading to an iterative version of the design cycle that we have described.

8 Acknowledgements

This paper was produced with the support of the Basic Technology initiative as part of the "Managing Uncertainty for Complex Models" project. We are grateful to Energy SciTech Limited for the use of the reservoir simulator software and the formulation of the example. We are also grateful for the detailed comments of the referees, which have lead to improvements in our account.

References

Bates, R. A., Buck, R. J., Riccomagno, E., and Wynn, H. P. (1996), "Experimental Design and Observation for Large Systems," *Journal of the Royal Statistical Society, Series B*, 58, 77–94.

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

Craig, P. S., Goldstein, M., Seheult, A. H., and Smith, J. A. (1997), "Pressure matching for hydrocarbon reservoirs: a case study in the use of Bayes linear strategies for large computer experiments," in *Case Studies in Bayesian Statistics*, eds. Gatsonis, C., Hodges, J. S., Kass, R. E., McCulloch, R., Rossi, P., and Singpurwalla, N. D., New York: Springer-Verlag, vol. 3, pp. 36–93.

— (1998), "Constructing partial prior specifications for models of complex physical systems," *Applied Statistics*, 47, 37–53.

Cressie, N. (1991), *Statistics for Spatial Data*, Wiley.

Cumming, J. A. and Wooff, D. A. (2007), "Dimension reduction via principal variables," *Computational Statistics & Data Analsysis*, 52.

Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1991), "Bayesian Prediction of Deterministic Functions With Applications to the Design and Analysis of Computer Experiments," *Journal of the American Statistical Association*, 86, 953–963.

Goldstein, M. and Rougier, J. C. (2006), "Bayes Linear Calibrated Prediction for Complex Systems," *Journal of the American Statistical Association*, 101, 1132–1143.

— (2007), "Reified Bayesian Modelling and Inference for Physical Systems," *Journal of Statistical Planning and Inference*.

Goldstein, M. and Wooff, D. A. (2007), *Bayes Linear Statistics*, Wiley.

Kennedy, M. C. and O'Hagan, A. (2000), "Predicting the Output from a Complex Computer Code when Fast Approximations are Available," *Biometrika*, 87, 1–13.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979), "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, 21, 239–245.

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

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

Qian, Z., Seepersad, C. C., Joseph, V. R., Allen, J. K., and Wu, C. F. J. (2006), "Building surrogate models based on detailed and approximate simulations," *ASME Journal of Mechanical Design.*

Qian, Z. and Wu, C. F. J. (2008), "Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments," *Technometrics*, to appear.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989), "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409–435.

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

Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D. (1992), "Screening, Predicting, and Computer Experiments," *Technometrics*, 34, 15–25.

Zečevič, A. I. and Šiljak, D. D. (1994), "Balanced decomposition of sparse systems for multilevel parallel processing," *IEEE Trans. Circ. Syst. Fund. Theory Applic.*, 41, 220–233.