Multilevel Emulation

JAC, MG

3 October 2007

1 Overview

The document begins in Section 2 by describing the uncertainty framework that will be used to describe a general computer model and also the methods that can be used to link multiple models. In Section 3 the general task of constructing an emulator is described, in addition to detailed discussion of how to link two multilevel emulators. A brief overview of the currently available code and its functionality is presented in Section 4, and any open questions, unresolved problems and areas for future areas of research are given in Section 5. An extended example is given in Section 6 where we apply these methods to a computer model of a hydrocarbon reservoir and present the results. Finally, Appendix 7 contains a comprehensive list of notation used in this document.

2 Structuring our beliefs

2.1 Computer simulators

A computer simulator is a complex computer program which attempts to produce an accurate representation of a real physical system of interest. Let this physical system take the unknown value $y = (y_1, \ldots, y_q)^T \in \mathcal{Y}$. We may also have observations, z, of the system value which we write as:

$$z = y + e, \tag{1}$$

where e is a measurement error which is independent of y.

Let us assume that we have a computer simulation F of this physical system, which is a function taking a collection of covariates and model parameters as input, $x = (x_1, \ldots, x_p)^T \in \mathcal{X}$, and returning the output vector $F(x)$. The computer model is assumed to be deterministic and as such will always give the same output when evaluated on the same collection of inputs.

Using the *best input approach* [7], we assume that there is a 'best' model input, $x^* \in \mathcal{X}$, for which the model output at this input value separates the system from all other uncertain quantities, including the best input itself and the model. The value $F(x^*)$ does not exactly reproduce the physical system, but given the value $F(x^*)$ we can learn nothing more about the simulator by making additional evaluations. This judgement is expressed via the relationship

$$y = F(x^*) + \eta \tag{2}$$

where η is the discrepancy between the physical system and the evaluation of the simulator at x^*, and where $F(x^*)$ and η are independent.

2.2 Multiple simulators

Suppose that that the simulator F can be evaluated at a variety of different levels of complexity (or resolutions). For example, the computer model could be expanded to include additional or more sophisticated mathematics in order to model the underlying system thus resulting in a more accurate representation. Alternatively, if the computer model is run over a grid (such as with finite element analysis) the size of the grid cells could be reduced to increase model complexity and to enhance its accuracy. Other examples include adjusting the quality of the numerical solver used to solve the differential equations representing the system or varying the number of covariates in the model. By such means, we can obtain a number of different versions of the computer model, all of which we expect to preserve the qualitative relationships between inputs and outputs since they are all approximating the same system. These simulators will therefore be related to one another and exhibit an interesting covariance structure, the advantages and structure of which we will discuss later in this document.

Assume that we have L different versions of our computer model where $F_\ell(x)$ is the level ℓ computer simulator and higher values of ℓ indicate higher resolution simulators. Generally, any evaluations of F at higher resolutions will be more representative of the physical system, but this additional accuracy comes at the price of increased computation time. Conversely, simple versions of the simulator will be fast and cheap to evaluate but their precision will be low. The coarser simulators will be crude approximations to the real system, but have the advantage of requiring minimal computation time to evaluate. These properties are pivotal to the development of multilevel emulators – the coarse model is cheap but evaluations are available in large numbers and so we use the information from these many runs to make informed choices for our prior specification for the fine model and to carefully determine feasible sites for evaluating the more expensive fine simulator.

2.3 Beliefs about a single simulator

If we first consider a single computer model, then our goal is to produce a stochastic representation of our beliefs which link the model inputs x to the deterministic scalar simulator output $F(x)$ – we call this stochastic representation of the computer model the *emulator*, f, the of the simulator F. Irrespective of the resolution of the simulator, we express our prior beliefs about this relationship as an equation of the form

$$f(x) = h(x) + r(x) \tag{3}$$

where $h(x)$ is a trend function of the inputs (such as a linear regression), which captures the majority of the global aspects of the variation of $F(x)$ with x; and $r(x)$ represents all the remaining variation in $F(x)$ which is not captured by the trend. This gives the general form of the emulator for any computer model.

Typically, the trend component of the emulator is a linear combination of simple deterministic functions of the input variables, such that $h(x) = g(x)^T \beta$. For problems where the number of input variables is small, we include functions of all input variables into the systematic component and $r(x)$ is treated as a stationary stochastic process in x. However, when the number of inputs is large

or when our beliefs about the output $F(x)$ are dominated by a small subset of the inputs then we prefer to express $h(x)$ solely in terms of functions of a reduced subset of inputs – the *active variables* x_A. By restricting our focus to the active variables, we now further decompose the residual terms into the sum of two uncorrelated components $\varepsilon(x_A)$ and $\delta(x)$ [2, 3, 1]. The emulator is now expressed as

$$f(x) = h(x_A) + \varepsilon(x_A) + \delta(x). \tag{4}$$

The trend component, $h(x)$, is now a regression over the active variables only, and $\varepsilon(x_A)$ is a smooth function of the active inputs which describes the portion of the residual variation between $F(x)$ and $h(x_A)$ which is dependent on x_A but is not captured by h. Values of $\varepsilon(x_A)$ will be strongly correlated for values of x_A which are close in the input space. We therefore assume that this residual is spatial in nature and consider ε to be a stationary stochastic process with a particular covariance function. The ε is typically assumed to be a weakly stationary process with an assumed covariance function of the form $\mathsf{Cov}\left[\varepsilon(x_A), \varepsilon(x_A')\right] = (1-\kappa_\delta)\sigma^2 R(x_A - x_A'; \theta, \kappa_\delta)$ where θ is the correlation length parameter and κ_δ controls the size of the nugget effect, δ. Thus ε is correlated across $\mathcal{X}$ and can be thought to represent additional systematic variation that is unaccounted for by h such as additional higher-order terms in x_A which are not included in the regression surface.

The final term $\delta(x)$ is a nugget residual which describes all the remaining variation in F that cannot be explained by x_A alone. Unlike ε, we choose to approximate δ as unstructured and uncorrelated random noise unless we have reason to do otherwise. This gives $\mathsf{Cov}\left[\delta(x), \delta(x')\right] = \kappa_\delta \sigma^2$ when $x = x'$, and 0 otherwise.

2.4 Beliefs about multiple simulators

Let us assume that due to differences in resolution we have obtained two versions of the computer model - a simple fast version F_0 (the *coarse* model) and a more complex higher-resolution version F_1 (the *fine* model). Since the coarse model is quick to evaluate we can obtain a large number of runs and consequently construct a detailed (and accurate) emulator for F_0. We then wish to create a structural framework that will link the emulators of both F_1 and F_0, which would then enable us to transfer our beliefs about the comparatively well-known coarse model, to the less-understood fine model. Having discussed how we would formulate our beliefs about each simulator individually, we now turn to structuring our beliefs about the relationship between the simulators. Ultimately, our goal is to obtain a prior emulator for our fine computer model, which we can then update using a batch of evaluations of the fine simulator.

Let us assume that we can emulate both of our simulators directly. From (4) we have

$$f_0(x) = h_0(x_A) + \varepsilon_0(x_A) + \delta_0(x), \tag{5}$$

$$f_1(x) = h_1(x_A) + \varepsilon_1(x_A) + \delta_1(x), \tag{6}$$

where we suppose that both f_0 and f_1 share the same set of active variables thus giving identical structure for both h_0 and h_1. Since both emulators contain

similarly structured elements it is fairly straightforward to link the two emulators by linking the individual components themselves. Thus we will create a model which relates $h_1(x_A)$ to $h_0(x_A)$, and similarly for ε_1 and ε_0.

As we increase the complexity of the computer model it is reasonable to assume that we introduce extra complexity and structure into the global trend component of the emulator. If we construct our emulator trend in terms of regressions of simple functions of the active inputs, then to represent this extra complexity we could introduce additional functions in the original set of active variables, or functions in variables which were not previously deemed to be active. This leads to a 2-level emulator construction of the form

$$f_0(x) = g_0(x_A)^T \beta_0 \qquad\qquad\qquad +\varepsilon_0(x_A) + \delta_0(x), \qquad (7)$$
$$f_1(x) = g_1(x_A)^T \beta_1 + g_{1+}(x)^T \beta_{1+} \qquad +\varepsilon_1(x_A) + \delta_1(x), \qquad (8)$$

where $g_+(x)$ and β_{0+} are the additional regression functions of the inputs present in the fine emulator, and their corresponding coefficients . If we include these additional regression terms in the fine simulator then we can allow the nuggets of the two emulators to be unconnected and only impose links on the trend and the structured residual components.

2.4.1 Multi-parameter autoregressive model

A simple mechanism for relating the components of the two emulators is to impose a recursive/autoregressive relationship. This method directly relates the coefficients and residuals of the fine emulator to those of the coarse emulator and requires the specification of only a small number of quantities. Consequently, this method of relating the simulators is advantageous if runs of the fine model are in comparatively short supply, which is a common situation when evaluation of F_1 is expensive.

A key link between the two simulators of different resolutions will be via covariances between the trends h_1 and h_0. Since the trend represents the majority of the variation of the simulator over the input variables, then by specifying covariances over these quantities we will be able to express our judgements about the global systematic relationships between the two models. We make the assumption that we express these two trend components in terms of the same set of active variables and basis functions $g(x_A)$, but we allow the regression coefficients to differ between the two models and we may also allow the fine model to have additional terms not present in the coarse model as mentioned above. We impose the autoregressive relationship on the two sets of coefficients, and so we express the individual elements of β_1 as

$$\beta_{1i} = \rho_i \beta_{0i}, \qquad (9)$$

where the each fine coefficient β_{0i} is a multiple ρ_i of the coarse coefficient β_{0i}. We allow each coefficient to have it own multiplier which allows each coefficient to be adjusted individually. We assign ρ a prior expectation of 1 to reflect that we initially believe the trends are the same. We also set $2 \times \mathsf{sd}\,[\rho_i] = 1$ to reflect that we believe there to be only a small chance that ρ_i (and hence β_0) will reverse its sign on the fine model. Correlations between the ρ are set to zero *a priori*.
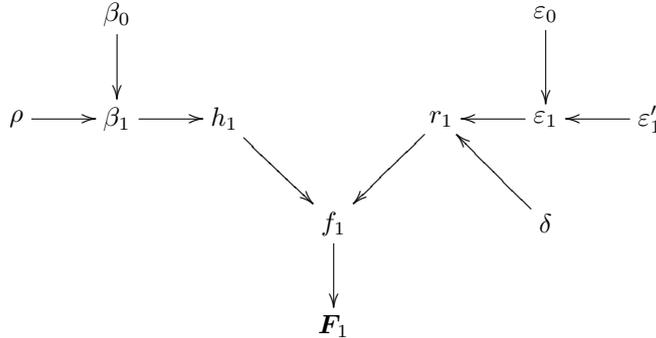
4

Figure 1: Graph of the linked 2-level model

The other important quantity which will be coupled between the two models is the residual term $\varepsilon(x)$. As with the coefficients, we may believe that the similarity between the residual terms arises due to the presence in the fine model of the coarse model's residual component. This common component would also be combined with a unique source of variation that is specific to the fine simulator. This leads to the following construction

$$\varepsilon_1(x_A) = \kappa_\varepsilon \varepsilon_0(x_A) + \sqrt{1 - \kappa_\varepsilon^2}\varepsilon_1'(x_A). \tag{10}$$

where ε_1' is the additional variation source in the fine residuals and the relative contributions of the two components to the sum is controlled by scalar multiplier κ_{ε_1}. Putting this all together gives us a mechanism for linking the emulators for the two different simulators; the picture for this linked formulation is shown in Figure 1.

2.4.2 Single-parameter autoregressive model

In the situation where the fine simulator runs are so expensive that we are unable to fit either of the previous models then we would require. The further simplification of the model would be to reduce the number of parameters controlling the relationship between the regression trends to a single autoregression multiplier. This is similar to the autoregressive model proposed by Kennedy and O'Hagan [8], though we propose only apply this relationship to the trend component of the emulator. This model is equivalent to defining the coefficients of the fine simulator as multiples of the coarse coefficients

$$\beta_1 = \rho\beta_0, \tag{11}$$

where ρ is an unknown scalar. This will allow for the trend of the coarse model to be subject to rotations to better suit the fine model, which whilst lacking the capacity for individually adjusting the coefficients as in the previous models is a reasonable approach when we are faced with a severely limited amount of data. Additionally, we still link the residuals as in (10) and may allow for additional terms in the trend to be introduced as in (8) if they are deemed to be important.

3 Constructing a multilevel emulator

Having established the uncertainty framework that we will use to describe multilevel emulation, we now describe the general strategy for applying these techniques to the 2-simulator problem. Given a coarse and fine simulator of the physical system, the methodology can be summarised in the following steps:

1. Emulate the coarse simulator

2. Construct a prior emulator for the fine model

3. Design and evaluate runs of the fine model

4. Update the fine emulator

5. History matching

3.1 Emulate the coarse simulator

Since the coarse simulator is quick and cheap to evaluate it is relatively easily to accumulate a large number of evaluations of this model. We may therefore believe that a Bayesian treatment of the coarse emulator may not be necessary as the results would be dominated by this large data set. We could therefore apply non-Bayesian techniques to obtain a simple lightweight emulator for the coarse model. Of course, there is nothing to stop us from specifying priors and updating them by the data if we have such information available and feel that this approach would be preferable. Having performed many runs of the coarse simulator, the task is then to use this to build a pragmatic representation of our judgements about F_0.

3.1.1 Designing for the coarse simulator

The coarse simulator is a cheap and quick approximation to the physical system. While it is less accurate in its representation of the real system under study, it has the important advantage of requiring only a small amount of computation time to evaluate thereby allowing for a relatively large number of runs. As we typically will be dealing with very high-dimensional input spaces, we easily suffer from the curse of dimensionality making it difficult to obtain an adequate coverage of the input space. Additionally, at this early stage we assume that we have little or no information about the potential effects that our many input variables may have on the response. We would therefore choose a space-filling design in order to spread the points as evenly as possible around the input space in order to try to capture the global aspects of the simulator's variation over $\mathcal{X}$. An obvious choice for a design at this stage would be a Latin hypercube.

We must first determine the number of runs we will make on the coarse simulator, n_0. Since it is relatively simple to construct a Latin hypercube we can construct a large number of such designs over the p input variables, and select that design which has the greatest value of the minimum distance between design points, i.e. a maximin Latin hypercube. It is beneficial to have good separation of design points, as when design points are very close together this will result in a singular variance matrix. We can then take the design $\boldsymbol{X}_0$ and evaluate the coarse simulator at each of the design points to obtain the coarse runs $\boldsymbol{F}_0$.

3.1.2 Determine which outputs to emulate

Since constructing an emulator is often a substantial endeavour it may be infeasible to attempt to emulate all outputs unless the number of outputs, q, is small enough that this would prove to be a manageable task. If q is sufficiently large that a complete emulation of all outputs is not a possibility, then we require a method for reducing the size of the problem. Since we are attempting to understand the physical system, it would appear to be more reasonable to focus on a subset of the outputs rather than, for example, linear combinations of the variables. Since all outputs are features of the same physical system given the inputs, it is a reasonable assumption that they will be correlated to one another to varying degrees so we could exploit this correlation structure to inform our selection of outputs.

An appropriate variable selection such as principal variables [5] could be used to identify a suitable subset of output variables which would account for a majority of the variation in the output space and would typically be strongly or moderately correlated to many of those remaining outputs. Focussing on these output variables would ensure that we are modelling the important quantities in terms of the amount of variance they contribute to the collection, and also those quantities which remained un-emulated could perhaps be obtained by considering the existing relationships between the active and inactive variables.

3.1.3 Identify the collection of active variables

The first task in building an emulator for a particular output variable is to identify the subset of input variables which are most important in explaining the global variation of F_0 as these will be the active variables for this emulator. The identification of the active variables is an important task as it is hoped that the regression surface in these terms will capture the majority of the global variation of both the coarse simulator and the fine simulator as well. In a Bayesian framework we would seek to elicit candidate active variables from the experts and likely combine this information with information from a selection of simulator evaluations.

As a simple non-Bayesian method, we could consider first fitting $F_0(x)$ on linear terms in x by using ordinary least squares, assuming an uncorrelated error structure. This approach is reasonable since we have a large sample size on a roughly orthogonal design with design points that are reasonably well-separated. We could then consider performing a stepwise variable selection over this model and retaining the first k variables which enter the model under forward selection, or the remaining k variables after a backward selection. This could provide us with a simple basis for the systematic component of the emulator.

A more detailed method for determining active variables is discussed in detail in [3], however their selection mechanism is primarily geared towards determining variables that would be informative for history matching using implausibility measures. It is clear that there is scope for using more sophisticated screening methods at this stage of the analysis.

3.1.4 Fit the regression surface over x_A

Having obtained our collection of active variables, we now need to fit an appropriate polynomial surface $\beta_0^T g(x_A)$ to F_0 which will hopefully capture the

majority of the global variation. We typically assume that the basis functions $g(x_A)$ comprise low-order monomial terms, pairwise interactions and a constant. Our goal is now to appropriately fit this linear model and obtain estimates for β_0. As usual, there is a wealth of possible methods for fitting such a model.

As with the selection of the active variables we may wish to fit $h(x_A)$ stepwise using ordinary least squares. However, in using OLS we are making the strong assumption that the model residuals $r_0(x)$ are uncorrelated. We have assumed by definition of $\varepsilon_0(x_A)$ that the residuals will be spatially correlated, which makes this assumption somewhat unreasonable. It may be the case that again with a large sample, orthogonal design and well-spaced design points that this is not unreasonable, nonetheless it may result in parameter and variance estimates which are not appropriate and potential problems if the residual and the regressors are not orthogonal. These models are trivial to fit in R using `step` and `lm`.

As an alternative to OLS for obtaining the regression component of the emulator, we might consider using stepwise fitting by generalised least squares (GLS). Generalised least squares is similar in spirit to OLS however it relaxes the assumption of uncorrelated residuals and allows them to have a covariance of a particular pre-specified form. However, it does require knowledge of the form of the correlation function before we fit the model. Functions such as `gls` in `nlme` in R can fit GLS regression models and also automatically estimate the correlation parameter (θ) and nugget size (κ_δ) at the same time by maximum likelihood (and so will suffer with the same problems as any MLEs). GLS methods will struggle however when fitting models to a large amount of data and so may be inappropriate when dealing with the coarse simulator as we will often have a large batch of evaluations at this stage.

If we have available prior information for the mean and variance of the β_0, then we will at this stage perform the Bayes Linear update using the available coarse runs. This gives $\mathsf{E}_0[\beta_0]$ and $\mathsf{Var}_0[\beta_0]$, where we write $\mathsf{E}_0[\cdot]$ as an abbreviation for $\mathsf{E}_{F_0}[\cdot]$ and the equivalent for the variance. However, if we are obtaining parameter estimates via frequentist methods then we treat our estimated coefficients and variances as the posterior values for β_0.

3.1.5 Update the residuals

Once we have captured the global trend in the data via our determination of $h(\cdot)$, it remains for us now to address the model for the local variation of the residuals. Rather than being treated as deterministic like the trend component, the residuals are treated as an unknown stochastic process. A fully-Bayesian approach would be to assume that the residuals followed a stationary Gaussian random process [9, 10, 6], whereas a Bayes Linear approach would assume a weakly stationary process. Since we are dealing with model residuals we assume a constant prior mean of zero, and specify a covariance function such as $\mathsf{Cov}\,[x, x'] = \sigma^2(1 - \kappa_\delta)\exp(-\theta|x - x'|^p)$ where σ^2 is the variance of the emulator residuals, κ_δ is a scalar in $[0, 1]$ which governs the magnitude of the nugget residual, and θ is the correlation length parameter.

However, in order to evaluate this covariance function we require values for the hyperparameters θ, σ^2, and κ_δ. Obtaining these values can be a complex problem, particularly in a full-Bayes analysis as we will be required to specify and update priors for these quantities. Since we are predominantly employing

frequentist methods to determine the parameters of the coarse emulator, we adopt a similar tactic in this case and estimate the hyperparameters via an appropriate method and plug in those estimates in subsequent calculations. Two possible methods for estimating these quantities are a GLS fit or a variogram-based method. As mentioned above, GLS is closely related to OLS though without the independent residual assumption. The GLS model is usually fitted via maximum likelihood and not only can estimate the hyperparameters, but also adjust the model coefficients to more favourable values however the method suffers from poor performance with large data sets in addition to the usual issues associated with finding the maximum of a likelihood surface. The variogram-based method [4] seeks to fit a variogram function to the data, and since the variogram and correlation functions are intimately related we can obtain correlation parameters from the fitted function. This method can cope with large amounts of data, but unlike GLS does not adjust the regression coefficients Having estimated the hyperparameters of the covariance function, we plug-in these values and update the prior process by the observed coarse simulator residuals and then use this conditioned process as the posterior residual process.

3.2 Construct a prior emulator for the fine model

Having constructed an emulator for the coarse computer model, we now wish to combine the information from the emulator with the structure linking the emulators to obtain a prior specification for the fine model (which has been updated by the coarse model runs). In the Bayesian case we will have specified priors for our key parameters $(\beta_0, \varepsilon_0, \delta_0)$ and these priors will have been consequently updated by the model runs, $\boldsymbol{F}_0$, giving us adjusted belief specifications for these quantities. If we did not use priors for these quantities then, in the absence of other information, we may believe that the estimated means and variances for $\beta_0, \varepsilon_0, \delta_0$ obtained from the fitting of the emulator constitute a reasonable belief specification given the coarse runs.

However, having a belief specification for the components of the coarse emulator is not sufficient to determine expectations and variances of the components of the fine emulator. To obtain these quantities we must combine our updated beliefs about the coarse emulator with the structure linking the two emulators and beliefs about the quantities controlling that connection. We have defined the fine emulator in (8) to be of the form

$$f_1(x) = g_1(x_A)^T \beta_1 + g_{1+}(x)^T \beta_{1+} + \varepsilon_1(x_A) + \delta_1(x).$$

Taking adjusted expectation and variances given the coarse model runs gives

$$\mathsf{E}_0[f_1(x)] = g_1(x_A)^T \mathsf{E}_0[\beta_1] + g_{1+}(x)^T \mathsf{E}[\beta_{1+}] + \mathsf{E}_0[\varepsilon_1(x_A)] + \mathsf{E}[\delta_1(x)], \quad (12)$$

$$\mathsf{Var}_0[f_1(x)] = g_1(x_A)^T \mathsf{Var}_0[\beta_1] g_1(x_A) + g_{1+}(x)^T \mathsf{Var}[\beta_{1+}] g_{1+}(x)$$
$$+ \mathsf{Var}_0[\varepsilon_1(x_A)] + \mathsf{Var}[\delta_1(x)]. \quad (13)$$

3.2.1 Priors for β_1

First let us consider the coefficients of the regression component of the emulator. We link the coefficients of the two emulators using the method in (9). We assume

9

that ρ and β_0 are independent of one another, and further that ρ is independent of $\boldsymbol{F}_0$. We can then obtain the following expression for the expectation of β_1.

$$
\begin{aligned}
\mathsf{E}_0[\beta_{1i}] &= \mathsf{E}_0[\rho_i \beta_{0i}] \\
&= \mathsf{E}\left[\rho_i\right] \mathsf{E}_0[\beta_{0i}]
\end{aligned} \tag{14}
$$

This expectation is then a function of our prior expectations of ρ and the expectation for the coarse coefficients after adjustment by the coarse model runs. We will use the priors for ρ that were introduced in Section 2.

Next we must obtain an expression for the variance of the fine coefficients. Expressing this as the covariance between any pair of elements of the coefficient vector

$$
\begin{aligned}
\mathsf{Cov}_0\left[\beta_{1i}, \beta_{1j}\right] &= \mathsf{Cov}_0\left[\rho_i \beta_{0i}, \rho_j \beta_{0j}\right] \\
&= \mathsf{E}_0\left[\rho_i \rho_j \beta_{0i} \beta_{0j}\right] - \mathsf{E}_0[\rho_i \beta_{0i}] \mathsf{E}_0\left[\rho_j \beta_{0j}\right] \\
&= \mathsf{E}\left[\rho_i \rho_j\right] \mathsf{E}_0\left[\beta_{0i} \beta_{0j}\right] - \mathsf{E}\left[\rho_i\right] \mathsf{E}_0[\beta_{0i}] \mathsf{E}\left[\rho_j\right] \mathsf{E}_0\left[\beta_{0j}\right] \\
&= \mathsf{E}\left[\rho_i \rho_j\right] \mathsf{E}_0\left[\beta_{0i} \beta_{0j}\right] - \mathsf{E}\left[\rho_i \rho_j\right] \mathsf{E}_0[\beta_{0i}] \mathsf{E}_0\left[\beta_{0j}\right] \\
&\quad + \mathsf{E}\left[\rho_i \rho_j\right] \mathsf{E}_0[\beta_{0i}] \mathsf{E}_0\left[\beta_{0j}\right] - \mathsf{E}\left[\rho_i\right] \mathsf{E}\left[\rho_j\right] \mathsf{E}_0[\beta_{0i}] \mathsf{E}_0\left[\beta_{0j}\right] \\
&= \mathsf{E}\left[\rho_i \rho_j\right] \mathsf{Cov}_0\left[\beta_{0i}, \beta_{0j}\right] + \mathsf{E}_0[\beta_{0i}] \mathsf{E}_0\left[\beta_{0j}\right] \mathsf{Cov}\left[\rho_i, \rho_j\right] \\
&= \left(\mathsf{Cov}\left[\rho_i, \rho_j\right] + \mathsf{E}\left[\rho_i\right] \mathsf{E}\left[\rho_j\right]\right) \mathsf{Cov}_0\left[\beta_{0i}, \beta_{0j}\right] \\
&\quad + \mathsf{E}_0[\beta_{0i}] \mathsf{E}_0\left[\beta_{0j}\right] \mathsf{Cov}\left[\rho_i, \rho_j\right] \\
&= \mathsf{Cov}\left[\rho_i, \rho_j\right] \mathsf{Cov}_0\left[\beta_{0i}, \beta_{0j}\right] \\
&\quad + \mathsf{E}\left[\rho_i\right] \mathsf{E}\left[\rho_j\right] \mathsf{Cov}_0\left[\beta_{0i}, \beta_{0j}\right] \\
&\quad + \mathsf{E}_0[\beta_{0i}] \mathsf{E}_0\left[\beta_{0j}\right] \mathsf{Cov}\left[\rho_i, \rho_j\right]
\end{aligned} \tag{15}
$$

Thus we can obtain an expression for $\mathsf{Var}_0[\beta_1]$ in terms of the prior expectations and variances for ρ and the updated beliefs about β_0 given the coarse runs.

3.2.2 Priors for ε_1

Using the relationship (10) and assuming κ_ε is a constant and that ε_0 and ε_1' are independent, the prior expectations for ε_1 are as follows

$$
\mathsf{E}_0[\varepsilon_1] = \kappa_\varepsilon \mathsf{E}_0[\varepsilon_0] + \sqrt{1 - \kappa_\varepsilon^2} \mathsf{E}\left[\varepsilon_1'\right] \tag{16}
$$

$$
\mathsf{Var}_0[\varepsilon_1] = \kappa_\varepsilon^2 \mathsf{Var}_0[\varepsilon_0] + (1 - \kappa_\varepsilon^2) \mathsf{Var}\left[\varepsilon_1'\right] \tag{17}
$$

We assume that ε_1' is another weakly stationary process with prior mean zero, and prior variance given by $\mathsf{Var}\left[\varepsilon_1'\right] = \sigma^2(1 - \kappa_\delta) R(x_A - x_A'; \theta)$. The values of σ, κ_δ and θ obtained from fitting the coarse model are also used here.

3.2.3 Priors for δ_1

We do not link the nugget residual across the two simulators, therefore we set $\mathsf{E}\left[\delta_1\right] = 0$, $\mathsf{Var}\left[\delta_1\right] = \kappa_\delta \sigma^2 \boldsymbol{I}$, using the same values for σ and κ_δ as were used on the coarse simulator.

3.2.4 Priors for β_{1+}

How we handle the presence of additional terms in the regression is still a matter for further investigation.

Substituting the expectations and variances for β_1 and ε_1 given in (14), (15), (16) and (17) back into the expressions (12) and (13) we can now obtain a belief specification for the fine simulator which has been adjusted by the coarse model runs. The next task is to carefully select a batch of runs of the fine simulator and then use these values to further update our beliefs about the fine emulator.

3.3 Design and evaluate runs of the fine model

Up to this stage we have performed a large number of runs of the coarse simulator, constructed an emulator for the coarse model, specified beliefs linking coarse and fine emulators and used all this information to construct a prior emulator for the fine computer model. We have now exhausted our source of information available in the coarse model and so we now focus on the fine emulator itself. We have constructed a prior emulator for this model, and so we now need to gather some fine simulator runs in order to update our emulator to more accurately reflect the behaviour of the fine model. Given that the fine model is expensive to evaluate we will be restricted to performing only a relatively small number of runs of the model and therefore we would wish to carefully select the locations at which we run to model to ensure that none of the runs are wasted.

An important point to consider when choosing the location for the fine model runs is that when we allow the trend to be defined in terms of a small subset of active variables (as in (4)), then only those variables will be driving the global variation in the emulators. Hence, we would like to ensure that our prospective design exhibits properties such as orthogonality, and a good separation of design points within this active subspace of inputs.

However, whilst the active variables are the most important for our emulators the simulator itself is a function of all inputs and therefore for any design of fine runs we must also specify values for the inactive variables. One option is to hold all inactive variables at a fixed central values, however this will not help us to determine whether the fine model exhibits any additional systematic variation in the inactive variables that was not present in the coarse emulator. Hence, it would also be desirable to ensure that the design points in the set of active inputs give a good spread of values in the trend components of the emulators. In other words we want to maximise some function of $\mathsf{Var}\left[\beta^T g(x_A)\right]$ for all the outputs. Obtaining a good spread of values in our emulator outputs should correspond to having a good spread of values in the simulator space once we perform the model runs themselves which should make it easier to capture the global trend in the outputs.

In order to consider a method for selecting such a design we clearly wish to focus on the global trends for each of our outputs. We could arrange our trend coefficients into a matrix, B, where each row corresponded to our coefficients for a particular output and non-zero entries indicate the presence of particular terms in the regression model. If we similarly construct a model matrix, X, then our goal would be to choose a design which maximises a particular function of the matrix $\mathsf{Var}\left[BX\right]$. Obvious candidates for the choice of function are the determinant (the "generalised variance"), and the trace. The trace would be

the simplest as it would require only the calculation and summation of the variances of each emulator trend.

As a first pass for a method to choose a suitable design we could follow the following algorithm:

- Generate many Latin hypercubes over all inputs (even inactive), since they are still Latin over the active vars and they are relatively trivial to obtain

- Select a subset of these designs, perhaps those which are best under minimax separation in the AVs. These will be our feasible set.

- Evaluate the appropriate criteria function of $\mathsf{Var}\,[BX]$ for each design in the feasible set. Choose the best.

Obviously this will not return an optimal design, however the design will be sensible and obtained simply and inexpensively. Having obtained a suitable design for the fine model runs, $\boldsymbol{X}_1$, we can now evaluate the fine simulator at each of these design points and obtain the matrix of fine model evaluations $\boldsymbol{F}_1$.

3.4 Update the fine emulator

We have expressed our beliefs about the fine simulator as in (12) and (13), which when written in matrix form gives

$$\mathsf{E}_0[\boldsymbol{F}_1] = \boldsymbol{G}^T \mathsf{E}_0[\beta_1] + \boldsymbol{G}_+^T \mathsf{E}\,[\beta_{1+}] + \mathsf{E}_0[\varepsilon_1(\boldsymbol{X}_A)] + \mathsf{E}\,[\delta_1(\boldsymbol{X})]\,, \qquad (18)$$

$$\mathsf{Var}_0[\boldsymbol{F}_1] = \boldsymbol{G}^T \mathsf{Var}_0[\beta_1]\,\boldsymbol{G}^T + \boldsymbol{G}_+^T \mathsf{Var}\,[\beta_{1+}]\,\boldsymbol{G}_+ \\ + \mathsf{Var}_0[\varepsilon_1(\boldsymbol{X}_A)] + \mathsf{Var}\,[\delta_1(\boldsymbol{X})]\,, \qquad (19)$$

where $\boldsymbol{X}$ is the matrix of all inputs at the design points, $\boldsymbol{X}_A$ is the matrix of active variables, $\boldsymbol{G}$ is the model matrix for the main regression $\boldsymbol{G} = g_1(\boldsymbol{X}_A)$ and $\boldsymbol{G}_+$ is the model matrix for the additional regression terms $\boldsymbol{G}_+ = g_{1+}(\boldsymbol{X}_A)$.

By combining the prior beliefs about the fine simulator elements given in Section 3.2 with the design points at which the fine simulator was evaluated we can now determine values for the expectations and variances of the fine model runs. We can update the fine simulator quantities β_1 and ε_1' by our fine simulator runs once we have determined the covariance between these quantities and the fine simulator runs.

Alternatively, we may wish to evaluate the both the coarse and the fine simulator at the points in the fine design, $\boldsymbol{X}_1$, and calculate the simulator differences $D_{01}(x) = F_1(x) - F_0(x)$. We would then obtain the matrix of differences between the model runs

$$\boldsymbol{D}_{01} = \boldsymbol{F}_1 - \boldsymbol{F}_0, \qquad (20)$$

which we could use to update the fine emulator instead of using the fine model runs. In order to perform the update we would require expectations and vari-

ances of the simulator differences

$$
\begin{aligned}
\mathsf{E}_0[\boldsymbol{D}_{01}] &= \boldsymbol{G}^T\left(\mathsf{E}_0[\beta_1] - \mathsf{E}_0[\beta_0]\right) + \boldsymbol{G}_+^T\mathsf{E}\left[\beta_{1+}\right] + \mathsf{E}_0[\varepsilon_1(\boldsymbol{X}_A) - \varepsilon_0(\boldsymbol{X}_A)] \\
&\quad + \mathsf{E}\left[\delta_1(\boldsymbol{X}) - \delta_0(\boldsymbol{X})\right], \\
&= \boldsymbol{G}^T 0\boldsymbol{G} + ? + \left(\kappa_\varepsilon\mathsf{E}_0[\varepsilon_0(\boldsymbol{X}_A)] + \sqrt{1 - \kappa_\varepsilon^2}\,\mathsf{E}_0[\varepsilon_1'(\boldsymbol{X}_A)]\right) - \mathsf{E}_0[\varepsilon_0(\boldsymbol{X}_A)] \\
&= (\kappa_\varepsilon - 1)\mathsf{E}_0[\varepsilon_0(\boldsymbol{X}_A)], \qquad\qquad\qquad\qquad\qquad (21) \\
\mathsf{Var}_0[\boldsymbol{D}_{01}] &= \boldsymbol{G}^T\mathsf{Var}_0[\beta_1 - \beta_0]\,\boldsymbol{G}^T + \boldsymbol{G}_+^T\mathsf{Var}\left[\beta_{1+}\right]\boldsymbol{G}_+ \\
&\quad + \mathsf{Var}_0[\varepsilon_1(\boldsymbol{X}_A) - \varepsilon_0(\boldsymbol{X}_A)] + \mathsf{Var}\left[\delta_1(\boldsymbol{X})\right] + \mathsf{Var}\left[\delta_0(\boldsymbol{X})\right], \\
&= \boldsymbol{G}^T\left(\mathsf{Var}_0[\beta_1] + \mathsf{Var}_0[\beta_0] - 2\mathsf{Cov}_0[\beta_1, \beta_0]\right)\boldsymbol{G}^T + ? \\
&\quad + \mathsf{Var}_0[\varepsilon_1(\boldsymbol{X}_A)] + \mathsf{Var}_0[\varepsilon_0(\boldsymbol{X}_A)] - 2\mathsf{Cov}_0[\varepsilon_1(\boldsymbol{X}_A), \varepsilon_0(\boldsymbol{X}_A)] \\
&\quad + \mathsf{Var}\left[\delta_1(\boldsymbol{X})\right] + \mathsf{Var}\left[\delta_0(\boldsymbol{X})\right],
\end{aligned}
$$

where we can also use the results that $\mathsf{Cov}_0[\beta_{1_i}, \beta_{0_i}] = \mathsf{E}\left[\rho_i\right]\mathsf{Cov}_0\left[\beta_{0_i}, \beta_{0_j}\right]$ and $\mathsf{Cov}_0[\varepsilon_1(\boldsymbol{X}_A), \varepsilon_0(\boldsymbol{X}_A)] = \kappa_\varepsilon\mathsf{Var}_0[\varepsilon_0(\boldsymbol{X}_A)]$.

3.4.1 Updating β_1

Applying the standard Bayes Linear update formulæ for the update of β_1 we obtain

$$
\begin{aligned}
\mathsf{E}_{01}[\beta_1] &= \mathsf{E}_0[\beta_1] + \mathsf{Cov}_0[\beta_1, \boldsymbol{F}_1]\,\mathsf{Var}_0[\boldsymbol{F}_1]^{-1}\left(\boldsymbol{F}_1 - \mathsf{E}_0[\boldsymbol{F}_1]\right) \\
\mathsf{Var}_{01}[\beta_1] &= \mathsf{Var}_0[\beta_1] - \mathsf{Cov}_0[\beta_1, \boldsymbol{F}_1]\,\mathsf{Var}_0[\boldsymbol{F}_1]^{-1}\,\mathsf{Cov}_0[\boldsymbol{F}_1, \beta_1],
\end{aligned}
$$

where we write $\mathsf{E}_{01}[\cdot]$ as the expectation first adjusted by $\boldsymbol{F}_0$ and now by $\boldsymbol{F}_1$. If we are updating by the simulator differences then we merely replace $\boldsymbol{F}_1$ with $\boldsymbol{D}_{01}$ and use the appropriate expectations and variances as defined above. The only additional quantity required in order to perform this update is $\mathsf{Cov}_0[\beta_1, \boldsymbol{F}_1]$ (or $\mathsf{Cov}_0[\beta_1, \boldsymbol{D}_{01}]$).

$$
\begin{aligned}
\mathsf{Cov}_0[\beta_1, \boldsymbol{F}_1] &= \mathsf{Cov}_0\left[\beta_1, \boldsymbol{G}^T\beta_1 + \boldsymbol{G}_+^T\beta_{1+} + \varepsilon_1(\boldsymbol{X}_A) + \delta_1(\boldsymbol{X})\right] \\
&= \mathsf{Cov}_0\left[\beta_1, \boldsymbol{G}^T\beta_1\right] \\
&= \mathsf{Var}_0[\beta_1]\,\boldsymbol{G} \\
\mathsf{Cov}_0[\beta_1, \boldsymbol{D}_{01}] &= \mathsf{Cov}_0[\beta_1, \boldsymbol{F}_1 - \boldsymbol{F}_0] \\
&= \mathsf{Cov}_0[\beta_1, \boldsymbol{F}_1] - \mathsf{Cov}_0[\beta_1, \boldsymbol{F}_0] \\
\mathsf{Cov}_0[\beta_{1_i}, \boldsymbol{F}_0] &= \mathsf{Cov}_0\left[\beta_{1_i}, \boldsymbol{G}^T\beta_0 + \varepsilon_0(\boldsymbol{X}_A) + \delta_0(\boldsymbol{X})\right] \\
&= \mathsf{Cov}_0\left[\beta_{1_i}, \boldsymbol{G}^T\beta_0\right] \\
&= \mathsf{Cov}_0[\rho_i\beta_{0_i}, \beta_0]\,\boldsymbol{G} \\
&= \mathsf{E}\left[\rho_i\right]\mathsf{Cov}_0[\beta_{0_i}, \beta_0]\,\boldsymbol{G}
\end{aligned}
$$

We now have sufficient information to perform the Bayes Linear update by the model runs and obtain adjusted moments $\mathsf{E}_1[\beta_1]$ and $\mathsf{Var}_1[\beta_1]$. If we wish to update by the model differences then we make the appropriate substitutions in the update formula.

3.4.2 Updating the residuals

Given our updated estimates for ρ we can obtain values for the fine coefficients. We can use these coefficients to determine the trend component of the fine emulator and hence obtain values for the residuals of the fine runs which can be used to update the ε_1' (and/or determine the presence of new trend components). The residual update is a current focus of research.

3.4.3 Diagnostics

Now we have obtained an emulator for the fine simulator is the appropriate time to perform diagnostics to evaluate the emulators adequacyand to check for the presence of additional terms and active variables. Determing and applying appropriate diagnostics is an area of future research.

3.5 History matching

We have now obtained a batch of runs of the fine simulator and an updated statistical model for the simulator itself. If our goal is to learn about the possible true input configuration for the physical system given our emulator, then we can now perform a history match to see whether we can eliminate any regions of $\mathcal{X}$ as infeasible. If we find there are regions of input space that are clearly implausible for matching our observed data, then we can construct a new design for additional runs of the fine simulator in the non-infeasible (?) areas of parameter space. The fine emulator can then be updated by these additional runs, and so we *refocus* our emulator on this reduced subspace. This process can then be iterated until we have narrowed the space of possible input choices to a sufficient level.

The process of history matching seeks to identify regions of input values or a single value for which $F(x)$ is acceptably close to y. The Bayes Linear strategy evaluates candidate simulator input values via *implausibility*, large values of which indicate that that particular input point is an infeasible history match. By evaluating implausibilities on a grid throughout the input space we can identify and exclude regions which are poor matches to the observed system history, and so we can progressively reduce the space of acceptable history matches.

The implausibility measure used to assess the legitimacy of a potential history match is the [reciprocal] squared coefficient of variation on our updated fine emulator:

$$\mathsf{impl}(x) = \mathrm{CV}^2(F_1(x) - y) = \frac{\mathsf{E}\left[F_1(x) - z\right]^2}{\mathsf{Var}\left[F_1(x) - z\right]}. \tag{22}$$

The numerator in this expression simply becomes $(\mathsf{E}\left[F_1(x)\right] - z)^2$ which can be calculated from evaluations of the fine emulator; and the denominator is expressed as $\mathsf{Var}\left[F_1(x) - z\right] = \mathsf{Var}\left[F_1(x)\right] + \mathsf{Var}\left[\eta\right] + \mathsf{Var}\left[e\right]$. Thus implausibility can be calculated from only the evaluated expectations and variations of the adjusted fine emulator, the data values, $\mathsf{Var}\left[\eta\right]$ and $\mathsf{Var}\left[e\right]$.

Large values of $\mathsf{impl}(x)$ for a particular choice of input values indicate that it is implausible that $F_1(x)$ is close to y. Conversely, small values of $\mathsf{impl}(x)$ indicate either that the adjusted variance of $(F_1(x) - y)$ is large or that the probability of a match is high. Extending implausibility to history matching

with multiple output variables is detailed in [3], however a simple method of combining $\mathsf{impl}(x)$ for many responses is to calculate

$$\mathcal{I}(x) = \max_j |\mathsf{impl}_j(x)| \qquad (23)$$

which is a conservative combination as it seeks only to rule in inputs that are acceptable to all output variables. An additional advantage of this method of combination is that the implausibilities can be calculated through individual univariate calculations.

4 Current Code

Currently, `R` code is available to perform the following functions:

- Design – Construct simple and maximin Latin hypercubes as prospective designs for the coarse model

- Select responses – choose a subset of k variables via Principal Variables (and also potentially any of a variety of other methods)

- Select active variables – select a subset of input variables for a given output via forward and backward stepwise selection

- Fit emulator trend – fit an emulator trend given the active variables using stepwise regression

- Fit emulator residuals – fit the emulator residuals using either GLS or variogram methods

- Update the fine emulator – obtain priors for β_1 and update using the fine model runs

- Evaluate the emulator – evaluate the posterior emulator at a given location

- Implausibility calculations – calculate implausibility over a set of input points, project onto 2-D plane and produce implausibility plots, maximise implausibility over multiple output variables.

5 Unanswered questions and outstanding issues

- Updating the residuals

- Diagnostics – is the coarse emulator reasonable for the coarse simulator? Is it a reasonable prior for the fine model? Is the fine emulator ok for the fine simulator?

- Univariate vs. multivariate emulators – currently only univariate. Multivariate emulator would require tensors and fancy maths. Implications for the links between models, eg ρ

- Extension to more than 2 resolutions

- Additional regression terms – evidence of additional systematic effects perhaps suggested by big residual variances (post-update), poor R2 for the model (could just be a crappy model)

- Comparison with Kennedy & O'Hagan

- Emulating the difference – rather than emulating the two simulators, emulate the difference between the two - advantages when simulators are hard to emulate, but the difference is well-behaved, can then easily estimate the fine simulator by combining a coarse model run with a difference emulator evaluation

- Shared latent component model for linking emulators – as below. would require relatively large pool of coarse and fine runs to estimate the many parameters

$$\beta_0 = \Lambda_0 \beta + \xi_0,$$
$$\beta_1 = \Lambda_1 \beta + \xi_1,$$
$$\varepsilon_0 = \kappa_{\varepsilon_0} \varepsilon + \sqrt{(1 - \kappa_{\varepsilon_0}^2)} \varepsilon_0',$$
$$\varepsilon_1 = \kappa_{\varepsilon_1} \varepsilon + \sqrt{(1 - \kappa_{\varepsilon_1}^2)} \varepsilon_1'$$

- Spatio-temporal effects – diagnostics for assessing spatial effects over inputs. Temporal effects will probably require a multivariate treatment of the emulator.

6 Example

6.1 Model Overview

A hydrocarbon reservoir is a three-dimensional region of porous rock (such as sandstone or limestone) within which hydrocarbons (i.e. oil and gas) can be found. This region of porous rock is capped with a layer of impermeable rock which traps the hydrocarbons in place, thus creating the reservoir. The hydrocarbons are usually found with water in the reservoir. The complex geology of the reservoir and the location and operating conditions of any wells are two fundamental components in simulating reservoir performance.

The Daisy reservoir is modelled on a $48 \times 26 \times 50$ grid. Not all grid cells in the grid are 'active', i.e. the grid is not a perfect cuboid. The model runs from the notional date of 1 January 2003 until 1 February 2009 reporting output every month. The reservoir contains a total of eight different geological faults. There are four production wells and five injection wells. Four of the injectors inject water and one injects gas; four of the injectors are not activated until after five years of production. The reservoir grid with wells and faults is shown in Figure 2 where colour is used to denote depth only.

There are a total of 20 input parameters that can be easily varied. Each of these parameters has been assigned a most-likely value and associated minimum and maximum possible values by the reservoir expert. In some cases these input parameters correspond directly to model properties such as aquifer porosity. In

Figure 2: The Daisy hydrocarbon reservoir

Param	ML	Min	Max	Description
kx_mult	1.0	0.6	1.2	x-permeability multiplier
ky_mult	1.0	0.6	1.2	y-permeability multiplier
kz_mult	1.0	0.6	1.2	z-permeability multiplier
P_mult	1.0	0.6	1.2	Porosity multiplier
FMUL_Y	0.01	0.0	1.0	Fault 'Y' transmissibility multiplier
FMUL_W	0.01	0.0	1.0	Fault 'W' transmissibility multiplier
FMUL_N1	0.01	0.0	1.0	Fault 'N1' transmissibility multiplier
FMUL_N2	0.01	0.0	1.0	Fault 'N2' transmissibility multiplier
FMUL_K	0.01	0.0	1.0	Fault 'K' transmissibility multiplier
FMUL_R1	0.01	0.0	1.0	Fault 'R1' transmissibility multiplier
FMUL_R2	0.01	0.0	1.0	Fault 'R2' transmissibility multiplier
FMUL_R3	0.01	0.0	1.0	Fault 'R3' transmissibility multiplier
swcr	0.23	0.2	0.3	Critical water saturation?
sowcr	0.179	0.14	0.2	Critical oil-water saturation?
sgcr	0.1	0.08	0.12	Critical gas saturation?
sogcr	0.007	0.05	0.07	Critical oil-gas saturation?
swco	0.231	0.2	0.25	Another water saturation...
AQK	100	50	500	Aquifer permeability
AQP	0.1	0.05	0.25	Aquifer porosity
AQR	2400	2000	20000	Aquifer radius

Table 1: Inputs to the Daisy reservoir simulation

Name	Description
bhp	Bottom-hole pressure
thp	Tubing-head pressure
pre	Pressure
oilrt	Oil produced this month
gasrt	Gas produced this month
waterrt	Water produced this month
oiltot	Cumulative oil produced
gastot	Cumulative gas produced
watertot	Cumulative water produced
wc	Water cut
gor	Gas-oil ratio

Table 2: Outputs for each well in the Daisy reservoir simulation

other cases these parameters are actually multipliers that scale other model properties such as permeability. The inputs are given in Table 1.

The output of the model is obtained monthly for every well in the reservoir running from 1 Jan 2003 to 1 Feb 2009. The reported variables are summarised in Table 2 for producing wells. For injectors similar output is given only reporting amount of oil/gas/water injected rather than produced, and omitting water cut and gas-oil ratio. This gives 80 output variables per run, with 73 observations each. Clearly the totals can be simply derived from the monthly productions and so one or other of these sets of variables could be ignored.

The coarse computer model is obtained by coarsening the model grid vertically so it consists of only 2 layers. The fine model is a similarly-obtained 25-layer model. The full 50-layer simulation is treated as "reality" since no real history data is available for this model. The 50-layer model was run at a particular input configuration (the 'most-likely' values as determined by the reservoir engineer) and combined with a small amount of random noise in order to generate observational data.

6.2   Coarse Emulator Construction Results

In summary form for now since we are focussing on the ML-link rather than emulation itself

- Total of 121 outputs observed at 73 time points.

- All inputs transformed to $[-1, 1]$

- Run coarse model over a 1500 point maximin Latin hypercube in all 20 inputs

- Huge number of outputs. Restrict to a single time point and consider all outputs for all wells at that point.

- Choose time point $T = 1096$. Halfway through the simulation, at the end of first phase (no active water injectors). Some vars have zero standard deviation – these are removed. 67 remaining output variables.

Figure 3: Correlation plot of hydrocarbon reservoir outputs at t=1096

- Output variables are reasonably highly correlated (see Figure 3), suggests principal variables should be effective

- Select only 2 PVs - 'Well 4 pre' and 'Well1 oilrt'. These account for 85.58491% of variation in the output space.

- Select 3 AVs for these variables - 'Well 4 pre' chooses kx_mult, ky_mult, P_mult. 'Well 1 oilrt' chooses kz_mult, P_mult, FMUL_N1.

- Fit the residual process by variograms
  Well4.pre: $\sigma = 0.3282644$, $\kappa_\delta = 0.05602681$, $\theta = 11.90904$;
  Well1.oilrt: $\sigma = 0.7222866$, $\kappa_\delta = 0.06282002$, $\theta = 5.970722$

- Multiple R^2 of the combined trend and conditioned residual process are
  – Well4.pre: 0.899, Well1.oilrt: 0.525. The fit for the first is quite good, but the fit for the second is somewhat less impressive. Problematic?

6.3 Multilevel Emulation Results

Having obtained an emulator of the coarse model by using 1500 runs, we can obtain estimates for $\mathsf{E}_0[\beta_0]$ and $\mathsf{Var}_0[\beta_0]$. We also use the coarse model runs to obtain numerical values for the hyperparameters σ, κ_δ, and θ. We now seek to use this coarse emulator as a basis for emulating the fine computer model using the formulation described in Section 2.4.1.

For the purposes of this illustration we shall focus on only one output variable, *W4.pre*. The emulator for this output contains three active variables, *kx_mult*, *ky_mult*, *P_mult*; of which *P_mult* is the dominant variable driving the majority of the variation in *W4.pre*. We shall now perform 50 runs of the fine simulator and use these to construct an multilevel emulator for the fine model, the adequacy of the emulator will then be assessed by comparing its performance with an additional 200 fine model runs. These 200 fine model runs are shown by the black points in Figure 4.

19

Figure 4: Scatterplot of 200 runs of the fine simulator with evaluations of the coarse and multilevel emulators

We take the standard priors for ρ as being $\mathsf{E}\left[\rho_i\right] = 1$, $\mathsf{Var}\left[\rho_i\right] = 1$, $\mathsf{Cor}\left[\rho_i, \rho_j\right] = 0$. We shall construct a fine emulator using both a multi-parameter autoregressive model and a single-parameter model. The first column contains the coarse simulator coefficients, followed by the coefficients obtained from the multiple and single parameter autoregressive multilevel emulators respectively. The final column contains the least squares estimates for the regression of *W4.pre* for comparison. The final row of the table gives the mean squared error of evaluations of the respective models at the 200 validation points. The emulator estimates are also displayed in Figure 4 where the single-parameter emulator estimates are coloured green, and the multiple-parameter emulator estimates coloured red.

The coarse emulator is clearly a poor fit for the fine simulator runs as it has the highest of the mean squared errors and the coarse emulator evaluations (blue) display a notable positive bias in Figure 4. The two multilevel emulators both exceed the performance of the coarse emulator, however it is the multiple-parameter method (red) which performs best with an MSE of 0.230 compared to the value of 0.520 for the single parameter model. Both emulators still exhibit a positive bias above the fine simulator points.

A further study was performed to assess whether the comparative performance of the two methods was affected by the number of evaluations used in the update. For this experiment, multilevel emulators were repeatedly constructed using both methods updating by 10, 25, 50, 100, 150 and 200 of the fine simulator runs. The performance was then assessed on the 50 remaining fine simulator runs. The results of this particular trial are presented in Table 4 and clearly

	Coarse LSQ	Fine Single	Fine Multiple	Fine LSQ
(Intercept)	0.269	-0.111	-0.180	-0.925
kx_mult	0.247	0.275	0.271	0.310
ky_mult	0.142	0.208	0.210	0.358
P_mult	1.419	1.716	1.715	1.758
kx_mult^2	-0.074	-0.010	-0.058	0.065
ky_mult^2	-0.049	-0.123	-0.175	-0.154
P_mult^2	-0.675	-0.416	-0.460	-0.044
P_mult^3	0.257	-0.135	-0.138	-0.254
kx_mult:P_mult	-0.140	-0.195	-0.181	-0.001
ky_mult:P_mult	-0.078	-0.080	-0.088	0.006
MSE	1.041	0.366	0.496	

Table 3: Coefficients and mean squared errors of the various emulators of *W4.pre*

Num Runs	Coarse	1/4	1/2	3/4
2	1.004	0.9212	0.9392	0.9587
5	1.004	0.8282	0.8577	0.8903
10	1.004	0.6925	0.7340	0.7828
25	1.004	0.5099	0.5629	0.6355
50	1.004	0.3600	0.4056	0.4762
100	1.004	0.2314	0.2596	0.3119
150	1.004	0.1817	0.2007	0.2394
200	1.004	0.1568	0.1705	0.2007

Table 4: Mean squared errors for the multilevel emulators when updated by a varying number of fine simulator runs and varying the contribution of the single multiplier component to the variance of ρ

show that the multiple parameter method is preferable to the single parameter method in all tested cases.

References

[1] P S Craig, M Goldstein, J C Rougier, and A H Seheult. Bayesian forecasting ofr complex systems using computer simulators. *Journal of the American Statistical Association*, 96(454):717–729, 2001.

[2] P S Craig, M Goldstein, A H Seheult, and J A Smith. Bayes linear strategies for history matching of hydrocarbon reservoirs. In J M Bernardo, J O Berger, A P Dawid, and A F M Smith, editors, *Bayesian Statistics 5*, pages 69–95. Clarendon Press, Oxford, UK, 1996.

[3] P S Craig, M Goldstein, A H Seheult, and J A Smith. 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*, volume 3, pages 36–93. Springer-Verlag, New York, 1997.

[4] N Cressie. *Statistics for Spatial Data*. Wiley, 1991.

[5] J A Cumming and D A Wooff. Principal variables. *Computational Statistics & Data Analsysis*, 52(3), 2007.

[6] C Currin, T Mitchell, M Morris, and D Ylvisaker. Bayesian prediction of deterministic functions with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991.

[7] M Goldstein and J C Rougier. Bayes linear calibrated prediction for complex systems. *Journal of the American Statistical Association*, 101(475):1132–1143, 2006.

[8] M C Kennedy and A O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

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

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

7 Notation

y	The 'true' value of the physical system, $y \in \mathcal{Y}$. Has dimensionality q.
q	The number of outputs from the computer model.
z	Observational data on y. See (1)
e	The measurement error incurred when observing y.
$F(\cdot)$	The deterministic computer model, $F : \mathcal{X} \to \mathcal{Y}$.
$f(\cdot)$	The emulator of the computer model – a stochastic representation of $F(\cdot)$.
x	The collection of inputs to the computer model, $x \in \mathcal{X}$. Has dimensionality p.
x^*	The 'best input' value which separates the system from all other uncertain quantities.
x_A	The collection of active inputs, $x_A \subseteq x$.
η	Model discrepancy.
F_1, F_0	The fine and coarse versions of the computer model.
$\boldsymbol{X}_0, \boldsymbol{X}_1$	The matrix of design points for runs of the coarse (fine) simulator.
$\boldsymbol{F}_0, \boldsymbol{F}_1$	The results of evaluating the coarse (fine) simulator at each design point in $\boldsymbol{X}_0$ ($\boldsymbol{X}_1$).
$h(\cdot)$	The systematic component of the emulator; a regression surface in the (active) inputs which captures the majority of the variation.
$r(\cdot)$	The 'residual' component of the emulator; the remaining variation which is not contained in $h(x)$.
$\varepsilon(x_A)$	The component of residual variation that is attributable to the x_A; a smooth function in x_A.
$\delta(x)$	The remaining variation in $F(x)$ which cannot be explained by x_A alone.
$g(x)$	A vector of simple functions of x
β	The coefficients in the polynomial representation of $h(\cdot)$
$R(x, x'; \theta, \kappa_\delta)$	The correlation function of the ε.
θ	Additional parameter(s) required by the correlation function. For the Gaussian correlation, θ is a scalar representing the correlation length.
κ_δ	Scalar controlling the size of the nugget effect. $\kappa_\delta \in [0, 1]$.
σ	The residual variance of $r(\cdot)$.
g_{1+}, β_{1+}	Additional regression functions and their coefficients present in the fine emulator, but not present in the coarse emulator.
Λ_0, Λ_1	Matrix of constants specifying the relationship between β_0 (β_1) and β.
ξ_0, ξ_1	Source of simulator specific variation in β_0 (β_1)
$\varepsilon_0', \varepsilon_1'$	Additional source of simulator specific variation in ε_0 (ε_1)
$\kappa_{\varepsilon_0}, \kappa_{\varepsilon_1}$	Scalar controlling the magnitude of the contribution of ε_0' (ε_1') to ε_0 (ε_1).
$D_{01}(x)$	The difference between the two simulators, $D_{01} = F_1 - F_0$.
$d(x)$	The emulator of the simulator difference, D_{01}.