Technical description of EssRisk

Table of Contents

Introduction....1
Proxy models....2
Workflow....2
Deterministic and probabilistic....3
Sensitivity....3
Linear Regression....4
  Choosing terms....8
  Criteria....8
  Methods for choosing terms....10
Proxy models....11
Restricted Likelihood (REML)....12
Bayesian model averaging....14
Restricted Likelihood (REML) with Bayesian averaging....15
Multicollinearity and clustering....17
Likelihood function....17
Projection operators....19
Matern correlation functions....20
Genetic algorithms....21
  How is the initial population generated?....22
  How is the next generation created?....22
  How do we know which members of the old population to choose?....23
  How is the old population replaced?....23
  How do we know we have converged?....23
Monte Carlo....23
  Simple Monte Carlo....24
  Markov Chain Monte Carlo....24
    Metropolis Hastings....25
    General remarks....25
    Random walk....26
    DREAM....26
    Hamiltonian methods....26
Synergistic method....28
Importance sampling and effective sample size....28
Sensitivity....29
Bayesian Optimisation....31

Introduction

EssRisk is a software product designed to perform calibration, forecasting and decision making under uncertainty. It was originally developed for use by oil and gas reservoir engineers, but is applicable to modelling in many other industries, such as pricing, home heating, and process plants.

EssRisk uses proxy models to accelerate simulation Bayesian calibration and uncertainty analysis. The workflow involves design of experiments, model updating, and sampling from S curves using DREAM, and Hamiltonian MCMC methods. Linear regression model selection employs information criteria with correction factors, balancing fit and complexity.

It is particularly suited where the model is expensive computationally, and surrogate models are used to accelerate the process.

The emphasis is on a practical tool which creates validated probabilistic ensembles of simulation results. It is the result of many decades of research, and the concepts have been applied to hundreds of studies.

Other documents describe background and case studies.

Proxy models

The approach in EssRisk is to have multiple proxy models, each of which emulated a point of interest in a simulation output. The proxy models themselves provide an exact representation of the simulation output for each run. The point of interest may be a forecast point, or a history point where we want to match history and the simulator.

The proxy model is a kriging (or Gaussian process) model, which exactly replicates the design points, and includes the polynomial average model. The correlation function is a Matern function, and the hyper parameters are optimised using a combination of evolutionary algorithms (GA) and the BOBYQA optimisation methods. The hyper parameters include a correlation direction, two Matern function parameters, and two correlation lengths (along and orthogonal to the projection direction). This hyper parameter optimisation is performed independently for each estimator point.

Workflow

The workflow is designed for optimal performance, but is moreover designed to create an equiprobable ensemble of full simulation runs which corresponds to the probability distribution defined by the proxy functions.

The workflow consists of :

- Import deck and define parameters and ranges. The simulation deck is imported, and the modifiers to be varied, and their ranges and prior distributions, are defined.
- Calculate and run scoping runs. The initial design consists of a Latin Hypercube Sampling method. This has become a conventional experimental design technique when using proxy models. It has the advantage that the number of design points does not depend on the number of modifiers. Each modifier is sampled across its complete range.
- Import history and define history match and prediction points. The history data is imported, and individual points to be matched, or for which prediction uncertainty is desired, are defined. Each of these points will have an individual proxy model.
- Update (or build) the proxy models. The proxy model is built from the existing design points. First a polynomial set of models is constructed. The models are selected using a unique Akaike/Bayesian Information Criterion, which include a correction factor and a Bayesian prior. A set of polynomial models is created using a combination of Genetic Algorithm (GA) and Markov Chain Monte Carlo methods. This set of models is then averaged into a single polynomial model, where the coefficients and variances are averaged. The variance of the beta coefficients is included in the overall variance calculation.
- Calculate the S curve. Once the proxy models have been constructed, sampling over the likelihood space is performed using Markov Chain Monte Carlo methods (MCMC). The methods used include Hamiltonian (NUTS), DREAM, and random walk (RWM).

- Sample the S curve. The S curve is sampled in order to generate the modifier values for the next run(s). If there is no prediction, the next runs are chosen by random sampling of the samples. If there is prediction, S curves are generated for each prediction point, and these S curves are sampled using a combination of random sampling and round robin sampling of S curve bins. There are special methods for round robin sampling when there is more than one prediction point (i.e. multiple round robins).

Deterministic and probabilistic

One of the pitfalls of existing proxy-based forecast uncertainty methods is that there can be a credibility gap. The S curve generated by the proxy model does not appear to correspond to the ensemble of simulation runs, and it can be difficult to find any simulation run which corresponds, for example, to the P10. This mismatch can either be due to a poor proxy model, poor Monte Carlo techniques, or a bias in the simulations. If the simulations are generated from some process which is designed to optimise a history match (either using a proxy model or from genetic programming methods) then the ensemble will inevitably be biased. One may, indeed, find all simulation runs above the P50 calculated from the proxy model.

It is therefore vital to synergise the proxy model and simulation model S curves, and this is done by sampling from the proxy S curve.

As each simulation run finishes, the proxy model is updated with the new information and a new MCMC calculation is performed. Once the proxy-generated S curve is ready, the next simulation run is generated by sampling from the S curve.

In this way, the simulation runs and the proxy model synergise in their respective S curves, and the proxy model can be seen to be merely an aid to generating a valid equiprobable set of simulation runs. For those sceptical of proxy models, the proxy model can become invisible during peer reviews. Moreover, the S curve does not depend on the proxy model being a close match to the simulation model. The deterministic simulation results form the final solution.

The sampling from the proxy S curve requires care. If there is a single prediction point, then a random sampling (from the Y axis) can be performed, but it is preferred to do a stratified sampling, using a randomised round robin technique.

If there is a multitude of prediction points, a simple round robin becomes infeasible. However, a multi round robin is used, whereby each prediction point has its own round robin, and there is a random permutation of prediction points. For each sample, one can predict what samples of the other prediction points it corresponds to, and the round robins are gradually filled until they are all complete and the process restarts.

In general, with multiple prediction points, it is not possible to generate simulation runs which form an ensemble which corresponds to every prediction S curve, but this is the nature of prediction uncertainty – tomorrow will not be a P10 for temperature, precipitation, sunshine and snow.

Sensitivity

There are two very different kinds of sensitivity analyses.

The first is to examine the sensitivity of simulation output to changes in input parameters. This can be displayed in tornado charts or spider plots. The sensitivities can be generated either directly from simulation runs, or from a proxy model. The ranges of the parameters directly affects the sensitivity, and these are normally determined beforehand by the asset team. This kind of study is commonly used to filter out the uncertain parameters into a smaller subset. It is also sometimes used at the end of a history match study to perform a sensitivity analysis around the best history match model.

These kinds of sensitivity analyses are not without merit, and they can be used as a graphical communication tool, but should be used with great caution. Filtering parameters is not so important with modern software tools, which can handle many hundreds of parameters, and sensitivity studies around a best history match are not statistically valid.

The second kind of sensitivity study is to examine how the team's beliefs influence the S curve. How do individual measurement data affect the S curve, how do beliefs on tolerances affect the curve? As we gather more data, the width of the S curve falls, and as we hold stronger beliefs on the quality of data measurements and the quality of the model, again the width decreases.

With a proxy model, these kinds of sensitivities can be calculated easily, and the opportunity arises for interactive tools which allow engineers and decision makers to examine, on a graphical user interface, how risk changes with different beliefs. The exposure of the direct correspondence between a prior beliefs and prediction uncertainty may be uncomfortable for some – not only does the S curve represent a multitude of possible predictions, but there is a multitude of possible S curves. The pivotal role of the engineer's judgement is clear.

Linear Regression

In the regression model there are n cases and p independent variables (or regressors). Each case has a measured response, and we are trying to fit the responses with the regressors, as a linear function of the coefficients. The regressors are assumed to be i.i.d. (independent and identically distributed random variables).

In linear regression we have a set of n responses

$$Y_i, x_i, \ \textit{for} \ \ 1 \le i \le n \tag{1}$$

where Y_i *is a scalar* $\wedge$ *each* x_i *is a d dimensional vector*, with values

$$x_{ij}, \ \textit{for} \ \ 1 \le i \le n, \ 1 \le j \le d \tag{2}$$

We form an equation which is linear in the β coefficients, and linear in a set p of functions

$$f_j(x_i) \ \textit{for} \ \ 1 \le j \le p \tag{3}$$

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j f_j(x_i) + \varepsilon_i \tag{4}$$

where ε_i is assumed to be an error term with zero mean and normal distribution.

Having found the coefficients β we may then predict the values for $y(x)$ for any x.

$$y(x) = \beta_0 + \sum_{j=1}^{p} \beta_j f_j(x) \tag{5}$$

The nature and number of functions p depends on the model being applied. In the simplest case, we use all terms linear in x and we have:

$$f_j(x_i) = x_{ij}, \wedge \quad d = p \tag{6}$$

We may choose not to use all the dimensions of x, and the regression is based on a subset which is found to be the most influential.

So for example we may have

$$f_1(x_i)=x_{i1}, f_2(x_i)=x_{i3}, f_3(x_i)=x_{i6}, \cdots \tag{7}$$

where we have chosen terms $x_1, x_3, x_6, \cdots$ to be in the model.

In general we then have a problem of choosing the possible functions, choosing which functions to use in the model, and calculating the β coefficients.

In our case, we have a possible total set of functions which represent (a) terms linear in x, $f_k(x_i)=x_{il}$, x (b) quadratic terms in x, $f_k(x_i)=x_{il}^2$ (c) cross terms in x, $f_k(x_i)=x_{il}x_{\mathrm{im}}$. The possible number of terms, including the intercept, is then $\left(1 + 2d + d*\frac{d-1}{2}\right)$. More complex functions could be included, for example exponential, but the number of possible functions can easily explode. Non-linear regression can use non-linear functions of β, for example $f_k(x_i)=\exp(\beta_k x_{il})$, but this greatly increases the difficulty in finding values of β as non-linear optimisation methods are required, and much of the subsequent formalism is invalid. In our approach, the proxy model takes account of these non-linear behaviours through alternative methods.

We then have a set of coefficients

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_p \end{pmatrix} \tag{8}$$

and errors

$$\varepsilon = \begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_p \end{pmatrix} \tag{9}$$

responses and functions

$$Y = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ \vdots \\ Y_n \end{pmatrix} \quad X = \begin{pmatrix} 1 & f_1(x_1) & f_2(x_1) & f_3(x_1) & \dots & f_p(x_1) \\ 1 & f_1(x_2) & f_2(x_2) & f_3(x_2) & \cdots & f_p(x_2) \\ 1 & f_1(x_3) & f_2(x_3) & f_3(x_3) & \cdots & f_p(x_3) \\ 1 & f_1(x_4) & f_2(x_4) & f_3(x_4) & \cdots & f_p(x_4) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & f_1(x_n) & f_2(x_n) & f_3(x_n) & \cdots & f_p(x_n) \end{pmatrix} \tag{10}$$

which may be expressed in vector-matrix form

$$Y = X\beta + \varepsilon \tag{11}$$

$$y(x) = f(x)^T \beta \tag{12}$$

To solve for the β coefficients we use the least squares method

$$\min_{\beta} \sum_{i=1}^{n} e_i^2 \tag{13}$$

$$\min_{\beta} \sum_{i=1}^{n} \left(Y_i - \beta_0 - \sum_{j=1}^{p} \beta_j f_j(x_i) \right)^2 \tag{14}$$

In matrix notation we have

$$\min_{\beta} (Y - X\beta)^T (Y - X\beta) \tag{15}$$

which is found at the point where the derivative is zero

$$\frac{\partial}{\partial \beta} \left[(Y - X\beta)^T (Y - X\beta) \right] = 0 \tag{16}$$

Giving values $\hat{\beta}$ where

$$X^T \left(Y - X\hat{\beta} \right) = 0 \tag{17}$$

$$\hat{\beta} = \left(X^T X \right)^{-1} X^T Y \tag{18}$$

and

$$\sum_{i=1}^{n} e_i = ¿ 0 \ ¿ \tag{19}$$

The above result depends on having an intercept coefficient β_0.

We may define various quantities of interest

Residual sum of squares

$$rss = \sum_{i=1}^{n} \left(Y_i - y\left(x_i\right) \right)^2 \tag{20}$$

Mean sum of squares

$$mss = \sum_{i=1}^{n} \left(Y_i - \overline{y} \right)^2 \quad where \quad \overline{y} = \frac{1}{n} \sum_{i=1}^{n} y\left(x_i\right) \tag{21}$$

Total sum of squares

$$tss = \sum_{i=1}^{n} \left(Y_i - \overline{Y} \right)^2 \quad where \quad \overline{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i \tag{22}$$

$$tss = mss + rss \tag{23}$$

R squared

$$R^2 = \frac{mss}{mss + rss} = 1 - \frac{rss}{mss + rss} \tag{24}$$

Adjusted R squared

$$R^2_{adj} = 1 - \frac{rss}{mss + rss} \times \frac{n - 1}{n - p - 1} \tag{25}$$

Estimate of variance sigma squared

$$\hat{\sigma}^2 = \frac{1}{n - p - 1} \sum_{i=1}^{n} \left(Y_i - y\left(x_i\right) \right)^2 \tag{26}$$

or

$$\hat{\sigma}^2 = \frac{rss}{n - p - 1} \tag{27}$$

In addition, the variance estimate of the β coefficient is given by

$$var\left(\hat{\beta}\right) = var\left(\left(X^T X\right)^{-1} X^T Y \right) = \left(X^T X\right)^{-1} X^T var\left(Y\right) X \left(X^T X\right)^{-1} = \hat{\sigma}^2 \left(X^T X\right)^{-1} \tag{28}$$

as

$$var\left(Y\right) = \hat{\sigma}^2 I \tag{29}$$

Choosing terms

Choosing the terms to use in the regression problem is a well known statistical problem, sometimes referred to as model building. It is a procedure often applied by a statistician together with their views on causation and a careful analysis of the data. In our situation, we need a fully automatic process, and we know (unless there is an error in our reservoir model, or compartments are completely sealed) that all parameters affect results, albeit they vary in their significance. The problem then becomes one of choosing the most influential terms.

There are two separate issues, the criteria used to rank the possible models, and separately the procedure to find the optimal model. We address first the issue of the criteria.

One of the dangers in this model building process is to avoid over fitting. When the number of cases is less than the total number of possible terms, it is easy to find a multitude of different models which provide a good fit to the data, but these will be very unreliable when predicting other data points. In addition, the number of terms used, and the quality of the fit, directly affects the estimate of uncertainty in the model (the sigma squared statistic). There are further issues which we will discuss later concerning multi collinearity and Bayesian model averaging.

Criteria

There are many different criteria which have been used in the past, but modern statisticians generally prefer to use approaches based on likelihood and information criteria. Even within these, there are many different flavours. We focus on the Akaike (AIC) and Bayesian (BIC) Information Criterion, although you will also find DIC, WAIC and many others.

The likelihood function is given by

$$L = \frac{1}{\left(2\pi\sigma^2\right)^{\frac{n}{2}}} e^{\frac{-(Y-X\beta)^T(Y-X\beta)}{2\sigma^2}} \tag{30}$$

with the log likelihood

$$\mathrm{LL} = -\frac{n}{2}\ln\left(2\pi\sigma^2\right) - \frac{(Y-X\beta)^T(Y-X\beta)}{2\sigma^2} \tag{31}$$

Minimising this wrt. to σ^2 and β gives $\hat{\sigma}^2$ and $\hat{\beta}$

$$\hat{\sigma}^2 = \frac{rss}{n} \tag{32}$$

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T Y \tag{33}$$

$$\mathrm{LL} = -\frac{n}{2}\ln\left(2\pi\right) - \frac{n}{2}\ln\left(\hat{\sigma}^2\right) - \frac{n}{2} \tag{34}$$

Then we can use the Akaike information criterion *AIC*

$$AIC = 2 \ll -2\left(p+2\right) \tag{35}$$

or the corrected form AIC_c , which attempts to avoid over fitting by penalising large values of p

$$AIC_c = 2 \ll -2(p+2)\frac{n}{n-p-3} \tag{36}$$

$$AIC_c = AIC - 2\frac{(p+2)(p+3)}{n-p-3} \tag{37}$$

This formulation does not include the number of degrees of freedom in the sigma squared, so an alternative is to use:

$$L = \frac{1}{\left(2\pi\sigma^2\right)^{\frac{(n-p-1)}{2}}} e^{\frac{-(Y-X\beta)^T(Y-X\beta)}{2\sigma^2}} \tag{38}$$

with the log likelihood

$$\text{LL} = -\frac{(n-p-1)}{2}\ln\left(2\pi\sigma^2\right) - \frac{(Y-X\beta)^T(Y-X\beta)}{2\sigma^2} \tag{39}$$

giving

$$\hat{\sigma}^2 = \frac{rss}{n-p-1} \tag{40}$$

$$\text{LL} = -\frac{(n-p-1)}{2}\ln\left(2\pi\hat{\sigma}^2\right) - \frac{n-p-1}{2} \tag{41}$$

with again

$$AIC = 2 \ll -2(p+2) \tag{42}$$

$$AIC_c = 2 \ll -2(p+2)\frac{n}{n-p-3} \tag{43}$$

$$AIC_c = AIC - 2\frac{(p+2)(p+3)}{n-p-3} \tag{44}$$

Some authors have proposed:

$$\hat{\sigma}^2 = \frac{rss}{n-p-3} \tag{45}$$

$$\text{LL} = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\left(\hat{\sigma}^2\right) - \frac{n-p-3}{2} \tag{46}$$

$$AIC_c = 2\mathrm{LL} - 2(p+2) \tag{47}$$

$$AIC_c = -n\ln(2\pi) - n\ln(\hat{\sigma}^2) - (n-p-3) - 2(p+2) \tag{48}$$

$$AIC_c = -n\ln(2\pi) - n\ln(\hat{\sigma}^2) - n - p - 1 = const. - n\ln(\hat{\sigma}^2) - p \tag{49}$$

Many prefer the Bayesian information criterion *BIC* which gives a larger penalty for large *n*

$$BIC = 2\mathrm{LL} - (p+2)\ln(n) \tag{50}$$

Prior beliefs can be included, for example beliefs on the relative number of linear and higher order terms, and this gives an adjustment factor:

$$-2p_l \ln(l-1) - 2p_u \ln(u-1) \tag{51}$$

where p_l, p_u are the actual number of linear and higher order terms respectively, and u, l are given by $l = N_l$, $u = N_u$ and N_l, N_u are the expected number of linear and higher order terms respectively.

In our method, we take a pragmatic approach, emphasising the importance of avoiding over fitting, and define *EIC*, using a combination of the above for AIC and BIC:

$$\mathrm{LL} = -\frac{n-p-1}{2}\ln(2\pi) - \frac{n-p-1}{2}\ln(\hat{\sigma}^2) - \frac{n-p-1}{2} \tag{52}$$

$$\hat{\sigma}^2 = \frac{rss}{n-p-1} \tag{53}$$

$$EIC = 2 \ll -(p+2)\ln(n) - 2\frac{(p+2)(p+3)}{n-p-3} - 2p_l \ln(l-1) - 2p_u \ln(u-1) \tag{54}$$

with suitable prior beliefs for u, l.

Methods for choosing terms

The best approach for using the above criteria for choosing the optimal subset is to evaluate all possible subsets. However, this is impractical for anything other than very small problems.

The use of evolutionary algorithms to find the best subset is of more recent interest. Naïve methods which simply use a mutation operator which involves a random flip (in or out) of a term have very slow convergence. In our approach, we use a crossover operator which greatly speeds up the computation.

The evolutionary algorithm approach is:

- *Start with a population of different subsets. These may be generated at random, using some permutation operator, and using random sizes of subsets, but all subject to p < n.*
- *Use mutation and cross over operators to generate a new population.*

- *Stop according to some convergence criterion.*

The gene which represents the subset is a simple bit array, which indicates whether a term is in or out of the subset. Because the array may be long, and there may be a large population, implementation needs to be done carefully to avoid memory issues.

The mutation operator can be a simple bit flip.

The crossover operator used consists of an AND/OR operator:

Given two parents P1 and P2, choose two children C1, C2 where C1 has all terms which are in both P1 and P2, and C2 has all terms which are either in P1 or P2.

C1 then becomes a small subset with terms common to P1 and P2, and C2 becomes a large subset with terms in either P1 or P2. This greatly accelerates the ability to build models.

One common rule for polynomial models is that higher order terms are only allowed if the linear term is in the subset. The above crossover method conforms naturally to this rule.

Proxy models

The proxy model extends the polynomial model to include a correlation function which depends on distance between two points $\mathscr{O}\left(\|z\|\right)$. This may be a Gaussian correlation, a Matern correlation, or others such as thin-plate spline or cubic spline.

There are many derivations of the proxy model, but we present the simplest version.

We extend the polynomial model to fit the data (note there is no error term)

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j f_j(x_i) + \sum_{k=1}^{n} \lambda_k \mathscr{O}\left(\|x_i - x_k\|\right) \tag{55}$$

and to predict values outside of the sample set

$$y(x) = \beta_0 + \sum_{j=1}^{p} \beta_j f_j(x) + \sum_{k=1}^{n} \lambda_k \mathscr{O}\left(\|x - x_k\|\right) \tag{56}$$

where

$$\Phi = \begin{pmatrix} \mathscr{O}\left(\|x_1 - x_1\|\right) & \mathscr{O}\left(\|x_1 - x_2\|\right) & \cdots \\ \mathscr{O}\left(\|x_2 - x_1\|\right) & \mathscr{O}\left(\|x_2 - x_2\|\right) & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{57}$$

or, assuming no nugget term,

$$\Phi = \begin{pmatrix} 1 & \mathscr{O}\left(\|x_1 - x_2\|\right) & \cdots \\ \mathscr{O}\left(\|x_2 - x_1\|\right) & 1 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{58}$$

and

$$\varphi(x)=\begin{pmatrix}\mathcal{O}\left(\left\|x_1-x\right\|\right)\\ \mathcal{O}\left(\left\|x_2-x\right\|\right)\\ \vdots\end{pmatrix} \tag{59}$$

We impose the condition that the model must exactly reproduce the sample points

$$Y=X\beta+\ \lambda\Phi \tag{60}$$

and an additional constraint

$$X^T\lambda=0 \tag{61}$$

which gives

$$\begin{pmatrix}Y\\0\end{pmatrix}=\begin{pmatrix}\Phi & X\\ X^T & 0\end{pmatrix}\begin{pmatrix}\lambda\\ \beta\end{pmatrix} \tag{62}$$

with solution

$$\lambda=\Phi^{-1}(Y-X\beta) \tag{63}$$

$$\beta=\ \left(X^T\Phi^{-1}X\right)^{-1}X^T\Phi^{-1}Y \tag{64}$$

$$E(y(x))=f(x)^T\beta+\varphi(x)\Phi^{-1}(Y-X\beta) \tag{65}$$

We use a modified form to include the variance in the beta coefficients

$$E(y(x))=f(x)^T\beta+\ \left(f(x)^T Var(\beta)X^T+\sigma^2\mathcal{O}(x)^T\right)\Phi^{-1}(Y-X\beta) \tag{66}$$

and we use for the variance

$$Var(y(x))\ =f(x)^T Var(\beta)f(x)+\ \sigma^2-\left(f(x)^T Var(\beta)X^T+\sigma^2\mathcal{O}(x)^T\right)\Phi^{-1}\left(f(x)^T Var(\beta)X^T+\sigma^2\mathcal{O}(x)^T\right)^T \tag{67}$$

For $Var(\beta)$ we use only the diagonal elements.

Restricted Likelihood (REML)

We have already seen the likelihood function for a polynomial model

$$L\ =\frac{1}{\left(2\pi\sigma^2\right)^{\frac{(n-p-1)}{2}}}e^{\frac{-(Y-X\beta)^T(Y-X\beta)}{2\sigma^2}} \tag{68}$$

$$\mathrm{LL} = -\frac{(n-p-1)}{2}\ln(2\pi\sigma^2) - \frac{(Y-X\beta)^T(Y-X\beta)}{2\sigma^2} \tag{69}$$

$$\hat{\sigma}^2 = \frac{rss}{n-p-1} \tag{70}$$

$$\mathrm{LL} = -\frac{(n-p-1)}{2}\ln(2\pi\hat{\sigma}^2) - \frac{n-p-1}{2} \tag{71}$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y \tag{72}$$

For a mixed model such as our proxy model, we need to optimize the hyperparameters by considering the restricted likelihood function REML

$$REML = \frac{|X^T X|^{\frac{1}{2}} |\phi|^{\frac{-1}{2}} |X^T \phi^{-1} X|^{\frac{-1}{2}}}{(2\pi\sigma^2)^{\frac{(n-p-1)}{2}}} e^{\frac{-(Y-X\beta)^T \phi^{-1}(Y-X\beta)}{2\sigma^2}} \tag{73}$$

or, using $V = \phi\sigma^2$

$$REML = \frac{|X^T X|^{\frac{1}{2}} |V|^{\frac{-1}{2}} |X^T V^{-1} X|^{\frac{-1}{2}}}{(2\pi)^{\frac{(n-p-1)}{2}}} e^{\frac{-(Y-X\beta)^T V^{-1}(Y-X\beta)}{2}} \tag{74}$$

The logarithm is given by

$$REMLL = -\frac{(n-p-1)}{2}\ln(2\pi\sigma^2) - \frac{1}{2}\ln|\phi| - \frac{1}{2}\ln|X^T \phi^{-1} X| + \frac{1}{2}\ln|X^T X| - \frac{(Y-X\beta)^T \phi^{-1}(Y-X\beta)}{2\sigma^2} \tag{75}$$

which, minimising wrt. σ^2 and β gives

$$\hat{\sigma}^2 = \frac{(Y-X\beta)^T \phi^{-1}(Y-X\beta)}{n-p-1} \tag{76}$$

$$\hat{\beta} = (X^T \phi^{-1} X)^{-1} X^T \phi^{-1} Y \tag{77}$$

Using

$$(Y-X\beta) = \left(\phi^{-1} - X(X^T \phi^{-1} X)^{-1} X^T \phi^{-1}\right) Y \tag{78}$$

we also have

$$(Y-X\beta)^T\phi^{-1}(Y-X\beta)=Y^T\left(\phi^{-1}-\phi^{-1}X\left(X^T\phi^{-1}X\right)^{-1}X^T\phi^{-1}\right)Y \tag{79}$$

Bayesian model averaging

There has been much recent interest in overcoming the problems arising out of the use of a single linear regressionmodel. In practice, there is likely to be a multitude of different models, each with a similar ability to fit the data, but with different predictions outside of the data samples. To overcome this, modern practitioners use Bayesian model averaging, and take a set of models M with corresponding weights w_M Each model fits the data. We then have:

$$Y=X_M\beta_M+\varepsilon_M \tag{80}$$

We then have for prediction

$$\hat{y}(x)=\sum_M w_M f_M(x)^T\beta_M \tag{81}$$

and in the mixed model context

$$\beta_M=\left(X_M^T\phi^{-1}X_M\right)^{-1}X_M^T\phi^{-1}Y \tag{82}$$

We may construct matrices $\tilde{f}(x)$ and $\acute{\beta}_M$ where $\tilde{f}(x)$ is the superset of all terms $f_M(x)$ and $\acute{\beta}_M$ is a vector of full dimension but with zeros where the corresponding term is not included in the model.

We may then express the model as

$$\hat{y}(x)=\tilde{f}(x)^T\sum_M w_M\acute{\beta}_M \tag{83}$$

and further simplify with

$$\tilde{\beta}=\sum_M w_M\acute{\beta}_M \tag{84}$$

so that we arrive back at the simple expressions

$$\hat{y}(x)=\tilde{f}(x)^T\tilde{\beta} \tag{85}$$

Of course, handling these supersets and subsets requires care in the implementation, with the use of suitable pointers for each model.

When looking at the variance, we need to include both within model variance and across model variance, so we have

$$var\left(y(x)\right)=\sum_M w_M\left[var\left(y(x)\right)_M+\left\{\hat{y}(x)_M-\ \hat{y}(x)\right\}^2\right] \tag{86}$$

$$var\left(y(x)\right)=\sum_M w_M\left[\tilde{f}(x)^T var(\beta)_M\tilde{f}(x)+\left\{f_M(x)^T\beta_M-\ \tilde{f}(x)^T\tilde{\beta}\right\}^2\right] \tag{87}$$

$$var\left(y(x)\right)=\sum_{M} w_M\left[\tilde{f}(x)^T var(\beta)_M \tilde{f}(x)+\left\{\tilde{f}(x)^T \acute{\beta}_M - \tilde{f}(x)^T \tilde{\beta}\right\}^2\right] \tag{88}$$

$$var\left(y(x)\right)=\tilde{f}(x)^T \sum_{M} w_M\left[var(\beta)_M+\left(\acute{\beta}_M - \tilde{\beta}\right)\left(\acute{\beta}_M - \tilde{\beta}\right)^T\right]\tilde{f}(x) \tag{89}$$

which can be expressed as

$$var\left(y(x)\right)=\tilde{f}(x)^T var\left(\breve{\beta}\right)\tilde{f}(x) \tag{90}$$

with a revised calculation for the variance of the beta coefficients

$$var\left(\breve{\beta}\right)=\sum_{M} w_M\left[var(\beta)_M+\left(\acute{\beta}_M - \tilde{\beta}\right)\left(\acute{\beta}_M - \tilde{\beta}\right)^T\right]_{diag} \tag{91}$$

Restricted Likelihood (REML) with Bayesian averaging

For the model averaged restricted likelihood function, we cannot average the normal REML, as each model has a different number of degrees of freedom. However, we may consider a set of models, where the index indicates the model number:

$$\overline{Y}=\begin{pmatrix} Y \\ Y \\ \vdots \end{pmatrix} \tag{92}$$

$$\overline{\beta}=\begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \end{pmatrix} \tag{93}$$

$$\overline{X}=\begin{pmatrix} X_1 & 0 & 0 \\ 0 & X_2 & 0 \\ 0 & 0 & \ddots \end{pmatrix} \tag{94}$$

$$\overline{X}^T\overline{X}=\begin{pmatrix} X_1^T X_1 & 0 & 0 \\ 0 & X_2^T X_2 & 0 \\ 0 & 0 & \ddots \end{pmatrix} \tag{95}$$

$$\overline{\phi}=\begin{pmatrix} w_1\phi & 0 & 0 \\ 0 & w_2\phi & 0 \\ 0 & 0 & \ddots \end{pmatrix} \tag{96}$$

$$\overline{X}^T \overline{\phi}^{-1} \overline{X} = \begin{pmatrix} \frac{X_1^T \phi^{-1} X_1}{w_1} & 0 & 0 \\ 0 & \frac{X_2^T \phi^{-1} X_2}{w_2} & 0 \\ 0 & 0 & \ddots \end{pmatrix} \tag{97}$$

We then have for the determinants

$$|\overline{\phi}| = \prod_M \left\{ w_M^n |\phi| \right\} \tag{98}$$

$$\left| \overline{X}^T \overline{\phi}^{-1} \overline{X} \right| = \prod_M \left\{ \frac{\left| X_M^T \phi^{-1} X_M \right|}{w_M{}^{p_M}} \right\} \tag{99}$$

$$\left| \overline{X}^T \overline{X} \right| = \prod_M \left| X_M^T X_M \right| \tag{100}$$

and we can construct the restricted likelihood

$$REMLL = -\frac{\ln\left(2\pi\sigma^2\right)}{2} \sum_M \left\{ w_M \left(n - p_M \right) \right\} - \frac{1}{2} \sum_M \left\{ \left(n - p_M \right) \ln\left(w_M \right) \right\} - \frac{1}{2} \sum_M \ln |\phi| - \frac{1}{2} \sum_M \ln \left| X_M^T \phi^{-1} X_M \right| + \frac{1}{2} \sum_M \tag{101}$$

When minimised, this gives

$$\hat{\sigma}^2 = \frac{\sum_M \left\{ w_M \left(Y - X_M \beta_M \right)^T \phi^{-1} \left(Y - X_M \beta_M \right) \right\}}{\sum_M \left\{ w_M \left(n - p_M \right) \right\}} \tag{102}$$

$$\hat{\beta}_M = \left(X_M^T \phi^{-1} X_M \right)^{-1} X_M^T \phi^{-1} Y \tag{103}$$

We can then derive

$$REMLL = -\frac{\ln\left(2\pi\hat{\sigma}^2\right)}{2} \sum_M \left\{ w_M \left(n - p_M \right) \right\} - \frac{1}{2} \sum_M \left\{ \left(n - p_M \right) \ln\left(w_M \right) \right\} - \frac{1}{2} \sum_M \ln |\phi| - \frac{1}{2} \sum_M \ln \left| X_M^T \phi^{-1} X_M \right| + \frac{1}{2} \sum_M \tag{104}$$

$$REMLL = Const. - \frac{\ln\left(2\pi\hat{\sigma}^2\right)}{2} \sum_M \left\{ w_M \left(n - p_M \right) \right\} - \frac{1}{2} \sum_M \ln |\phi| - \frac{1}{2} \sum_M \ln \left| X_M^T \phi^{-1} X_M \right| \tag{105}$$

Multicollinearity and clustering

In the introduction to linear regression, we state that the regressors are assumed to be i.i.d. (independent and identically distributed random variables). Unfortunately, this is rarely the case.

We have either the problem of clustering, where samples are in distinct clusters, or correlation between regressors, which is termed multicollinearity. In the extreme case, where two samples are identical, the number of samples is effectively one less than the actual number. Where two samples are 100% correlated, the same situation occurs. In either of these extreme cases, the relevant matrices become singular. In effect, two beta coefficients can take arbitrary values, so long as the sum is equal to a constant.

In our history matching situation, the initial scoping runs, if done with a well designed Latin Hypercube design, are not correlated. But, as we gather more history match results, the samples will become more and more correlated. This can, if not accounted for, lead to over fitting and too many polynomial terms enter the equations. The penalty functions used in the criterion for polynomial functions become too loose, as the effective number of samples is lower than the actual number.

To overcome this effect, various measures have been proposed .

One approach is to define a variance inflation factor (VIF). This may be more useful as a diagnostic rather than a cure. A better approach which has been proposed is to use, as an effective sample correction factor,

$$\text{effective sample size} = tr\left(X\left(X^T X\right)^{-1} X^T \right) \tag{106}$$

and this then replaces the actual sample size in the penalty function. However, this is expensive to calculate during the polynomial term selection algorithm. The above references present a variety of approaches to deal with this important topic. We use a very simple, practical method, but further investigation is necessary.

When performing Cholesky decomposition, we have

$$LD\,L^T = X^T X \tag{107}$$

We may then examine the diagonal matrix D and compare it with the diagonal elements of $X^T X$.

We then have an estimate

$$\text{effective sample size} = \frac{n}{p+1} \sum D_i / \left(X^T X \right)_{ii} \tag{108}$$

and this is used as the effective number of samples in the penalty functions.

Note that in this formulation

$$\text{effective sample size} \le \frac{np}{p+1} \tag{109}$$

Likelihood function

The likelihood function is a product of the individual history match point likelihoods. The individual likelihoods are the probability that the simulation run is within the tolerance for that point . Of course,

where we have done a simulation run, the probability is zero or one. However, we can use the proxy model to provide an estimate where we have not done a run.

$$Likelihood(x_s) = \prod_{i=1}^{n_h} L_i(x_s) \tag{110}$$

where n_h is the number of history match points and x_s is sample s

Gaussian (normal) density function

$$\varphi(\omega, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(\omega-\mu)^2}{2\sigma^2}} \tag{111}$$

Cumulative Gaussian (normal) distribution function

$$\vartheta(\alpha, \mu, \sigma) = \int_{-\infty}^{\alpha} \varphi(\omega, \mu, \sigma)\, d\omega \tag{112}$$

We have three choices for the likelihood function:

a)

$$L_i(h_i, tol_i, \mu_i(x), \sigma_i(x)) = \vartheta(h_i + tol_i, \mu_i(x), \sigma_i(x)) - \vartheta(h_i - tol_i, \mu_i(x), \sigma_i(x)) \tag{113}$$

$$\mu_i(x) = E(Y_i(x)) \tag{114}$$

$$\sigma_i^2(x) = Var(Y_i(x)) + tol_i^2 \tag{115}$$

b)

$$L_i(h_i, tol_i, \mu_i(x), \sigma_i(x)) = \vartheta(h_i + \sigma_i(x), \mu_i(x), \sigma_i(x)) - \vartheta(h_i - \sigma_i(x), \mu_i(x), \sigma_i(x)) \tag{116}$$

$$\sigma_i^2(x) = Var(Y_i(x)) + tol_i^2 \tag{117}$$

c)

$$L_i(h_i, tol_i, \mu_i(x), \sigma_i(x)) = \vartheta(h_i + tol_i, \mu_i(x), \sigma_i(x)) - \vartheta(h_i - tol_i, \mu_i(x), \sigma_i(x)) \tag{118}$$

$$\sigma_i^2(x) = Var(Y_i(x)) \tag{119}$$

Note

a) avoids difficulties with the non smoothness of (c)

b) encourages more exploration than (a)

c) is the strict definition of the probability of the estimate being within tolerance of the history. When Var(x) = 0 (i.e. at simulation points) it is 0 or 1. The simulation is either within or without. This can cause difficulties with the MCMC algorithms.

We have chosen to implement method b).

Projection operators

Where the correlation function is isotropic, we have, for a Gaussian correlation function with correlation length , and a vector x between two points:

$$\exp\left(\frac{-x^T x}{\theta^2}\right) \tag{120}$$

In our correlation function, we use two different correlation lengths, one along a vector and the other orthogonal to that vector. For a given unit vector u we may define

$$P = u u^T \tag{121}$$

with properties

$$P = P^T \tag{122}$$

$$PP = u u^T u u^T = u u^T = P \tag{123}$$

$$(I-P)(I-P) = I + PP - 2P = (I-P) \tag{124}$$

$$P(I-P) = P - PP = 0 \tag{125}$$

Given two inverse correlation lengths a and b, we may then transform x:

$$\bar{x} = a Px + b(I-P)x \tag{126}$$

$$\bar{x}^T \bar{x} = a^2 x^T PPx + b^2 x^T (I-P)(I-P)x \tag{127}$$

$$\bar{x}^T \bar{x} = a^2 x^T Px + b^2 x^T (I-P)x \tag{128}$$

$$\bar{x}^T \bar{x} = (a^2 - b^2) x^T Px + b^2 x^T x \tag{129}$$

$$x^T Px = (x^T u)^2 \tag{130}$$

and we then have a correlation function

$$\exp(-\bar{x}^T \bar{x}) \tag{131}$$

This transformation can be used for either Gaussian or Matérn correlation function.

Matern correlation functions

For a Gaussian, or normal, correlation function we have:

$$\phi_{i,j} = \exp\left(\frac{-(x_i - x_j)^T (x_i - x_j)}{\theta^2}\right) \tag{132}$$

where θ is the correlation length.

Matérn correlation functions are more flexible. For a separation distance d > 0 between locations of two observations, the Matérn correlation of the two observations is given by

$$\rho(d;\varsigma,\nu) = \frac{1}{2^{\nu-1}\Gamma(\nu)}\left(\frac{d}{\varsigma}\right)^{\nu} K_{\nu}\left(\frac{d}{\varsigma}\right) \tag{133}$$

where $K_{\nu}\left(\frac{d}{\varsigma}\right)$ is a modified Bessel function of the third kind of order ν. $\varsigma > 0$ is a range parameter, and $\nu > 0$ is a smoothness parameter.

When ν is of the form $(m + ½)$, $K_{\nu}(z)$ takes an explicit form and the Matérn correlation function is the product of $e^{\frac{-d}{\varsigma}}$ and a polynomial of degree m in d.

For example, with $\nu = ½$

$$K_{½}(z) = \sqrt{\frac{1}{2}\frac{\pi}{z}}\ e^{-z} \tag{134}$$

so that

$$\rho(d;\varsigma,\nu=½) = e^{\frac{-d}{\varsigma}} \tag{135}$$

since $\Gamma(½) = \sqrt{\pi}$, which is the familiar exponential correlation function.

Similarly

$$\rho\left(d;\varsigma,\nu=\frac{3}{2}\right) = e^{\frac{-d}{\varsigma}}\left(\frac{d}{\varsigma}+1\right) \tag{136}$$

$$\rho\left(d;\varsigma,\nu=\frac{5}{2}\right) = e^{\frac{-d}{\varsigma}}\left(\frac{1}{3}\left(\frac{d}{\varsigma}\right)^2+\frac{d}{\varsigma}+1\right) \tag{137}$$

In the limit as ν approaches infinity, with ς approaching 0 in such a way that $\beta = 2\nu^{½}\varsigma$ remains constant, the Matérn correlation function approaches the Gaussian correlation function

$$\rho(d;\beta) = e^{-\left(\frac{d}{\beta}\right)^2} \tag{138}$$

In our implementation, we use the simplified versions for $\nu = \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}, \frac{9}{2}, \frac{11}{2}, \infty$ and interpolate between them for performance.

Derivatives of Matérn functions may be derived, but in our work we apply derivatives to the interpolated values. Care needs to be taken to ensure there are no derivative discontinuities across discrete half integer values of ν.

An alternative formulation uses the parameter $\beta = 2\nu^{½}\varsigma$ as the range parameter. The advantage is that β is more independent of ν - ν determines the behaviour at the origin, and hence the smoothness, and β broadly determines how quickly the correlations decrease with distance, roughly independently of ν.

Overall, we have a Matérn function

$$Matern\left(\left(x_i - x_j\right),\ \alpha,\ \beta, \mu, \rho,\ \omega\right) \tag{139}$$

and the following parameters which need to be optimised:

$\mu = direction\ \ vector$

$\alpha = length\ \in\ direction\ \ \mu$

$\beta = length\ \ orthogonal\ ¿\ \ direction\ \ \ \mu$

$\rho = nugget$

$\omega = Matérn\ \ order$

The nugget term is a small term which reduces the correlation function at zero. It is useful where the data has some noise – either stochastic or chaotic.

The total number of terms to optimise is d + 4

Genetic algorithms

Genetic (or evolutionary) algorithms are, at a superficial level, very simple. The basic idea is:

1. Generate a possible population – each member of the population has some value of the parameters and an objective value
2. Generate new members of the population by performing some operation on the existing members of the population
3. Replace existing population with the new members
4. Repeat until converged

Within this very simple scheme there are various choices which can radically affect the efficiency of finding solutions, and indeed whether the global optimum is found at all.

The tuning of these approaches to particular problems is vital, and we below give an indication of what choices are available.

How is the initial population generated?

A Latin Hypercube design is one option, but there may be alternatives depending on the overall situation. Possibly the algorithm is being done repeatedly, or maybe there is some screening or intelligence used to generate a useful initial population.

How is the next generation created?

The next generation is generated by applying operators, where the term is used in a very general sense. This is a fertile field for imagination, and preferably the operators should be chosen to suit the particular problem. Standard operators are mutation, which operates on a single member of the population (which might simply be a random change in the parameter values, or flipping of Boolean values) or crossover (where two members of the population are taken and combined in some way to make another two members). Other common operators are differential evolution and particle swarm. These are often much more efficient than the basic mutation and crossover operators. Some operator may have some hyper parameters which can be adjusted, such as the random walk step length.

When dealing with scheduling problems, different operators are required, with some permutation operators to vary the schedule. These may need to be created especially for a given problem, where the schedule constraints are complex (e.g. scheduling a drilling rig).

An incomplete set of potential operators is :

Mutation operators

- Particle Swarm
- MPT
- Polynomial
- Power
- Uniform
- Uniform Upper Lower
- Non Uniform
- Whole Uniform
- Whole Uniform Upper Lower
- Whole Non Uniform
- Boundary

Crossover operators:

- SBX
- Trig
- Gradient
- Heuristic
- Multiple
- Laplace

- Differential Evolution

How do we know which members of the old population to choose?

All operators take one (mutation) or two (crossover) or more (e.g. differential evolution) members of the old population to generate the new population. There are different approaches to choosing which of the old members to use. It could be a simple random selection, or a weighted random selection (where members with a better value of the objective are chosen preferentially). The weighting could be based on ranking or objective value. The functional form of the weighting (how rapidly does it decay across the population?) is another factor to consider.

More sophisticated approaches include tournament and round robin selection, which tend to be better at maintaining diversity in the population. These are themselves subject to further parameter choices (how many members in a tournament?).

These choices greatly affect the rate of convergence and the maintenance of diversity. Unfortunately, these two objectives are in conflict – if 'good' parents are chosen, it will converge rapidly to possibly a local minimum. If 'poor' parents are chosen, it will converge more slowly but explore further.

How is the old population replaced?

In a generational scheme, the new population is created as a whole at the same time. The old population is then replaced, with possibly some members of the old population retained if the objectives are good. The number to retain is another parameter to adjust.

In a steady state scheme, the old population is continually updated for each new member of the population. There are again various options – a) replace the worst member of the old population with the new member if it is better b) replace the existing member of the old population with the new member if it is better. The replacement strategy has a big impact on the degree of diversity within the population.

How do we know we have converged?

The slow convergence rate of evolutionary algorithms, and the lack of any knowledge of derivatives or Hessians, makes this difficult. In practice, it may simply be a fixed limit on the number of function evaluations.

Monte Carlo

For a random variable x, and a probability density function $p(x)$ the probability of x falling in an arbitrary interval (a', b') is given by:

$$P\{a' \le x \le b'\} = \int_{a'}^{b'} p(x)\,dx \tag{140}$$

The cumulative distribution function of x is given by

$$F(x) = \int_{-\infty}^{x} p(t)\,dt \tag{141}$$

where

$$F(\infty) = 1 \tag{142}$$

One may also define the inverse cumulative distribution function

$$F^{-1}(y) = min\{x; F(x) \geq y\} \tag{143}$$

This inverse function may be used to generate a random variable with cumulative distribution function $F(x)$.

Where we are prediction some function $f(x)$ the probability of $f(x)$ falling between an interval arbitrary interval (a', b') is given by:

$$P\{a' \leq f(x) \leq b'\} = \int_{f(x) \geq a'}^{f(x) \leq b'} f(x)\, p(x)\, dx \tag{144}$$

The cumulative distribution function of $f(x)$ is given by

$$F(x) = \int_{-\infty}^{f(t) \leq x} f(t)\, p(t)\, dt \tag{145}$$

These equations can be generalised to the multivariate case where x is a vector x

Simple Monte Carlo

In simple Monte Carlo, each dimension of x is independent with its own probability density function. The process to calculate the cumulative distribution function is then:

- Generate random numbers in the interval $(0,1)$, and use the inverse cumulative probability distribution F^{-1} to generate samples for each dimension of x
- For each sample x, calculate $f(x)$
- Sort the samples $f(x)$ and generate the cumulative distribution function, or S curve.

Note that the samples generated in the first step are equiprobable samples.

This approach can be extended, through the use of copulas, to handle correlations between parameters.

This method is suitable for an appraisal case where we have no history data.

Markov Chain Monte Carlo

When the probability density function is complex and cannot be partitioned into independent distributions, Markov Chain Monte Carlo (MCMC) methods need to be used. This is the case where we have model calibration or history matching. MCMC methods generate a sequence of samples, which conform to certain strict rules such as being a valid Markov chain, geometric ergodicity, and detailed balance (reversibility).

Metropolis Hastings

There are two distinct parts of an MCMC algorithm. The first is to generate new samples. The second is whether to accept or reject the new sample. The accept/reject rule is normally based on Metropolis Hastings.

The general algorithm is:

Given x_i with probability $p(x_i)$ propose a new sample $x_i^{'}$ with probability $p(x_i^{'})$

Choose a random number a in the interval (0,1)

$$If \;\; a \; < min\left\{ \frac{p(x_i^{'})}{p(x_i)}, \; 1 \right\} \; then \;\; x_{i+1} = \; x_i^{'}, \; else \;\; x_{i+1} = \; x_i \tag{146}$$

In this way, new samples are always accepted if they have a higher probability, and are accepted according to a random number if they have a lower probability.

General remarks

In general, when running MCMC methods, there are some common principles whatever chain generation method is used.

It is often found useful to run several independent chains with different starting points. Sometimes this is regarded as a way to improve exploration, but that is a misconception as the overall set of chains has only converged if every single chain has converged. However, multiple chains have their uses in testing for convergence, and several chain generation methods rely on the existence of multiple chains.

Burn-in is normally used, whereby initial samples are rejected until the chain has settled down. For example, if starting far from the minimum, samples are rejected until the area around the minimum starts to be explored.

Tuning is a critical factor. MCMC methods have various parameters, and performance depends critically on those parameters. Often in the past this tuning has been done by hand, which makes it impractical for our application. However, recent work has made significant progress in this area, and self-tuning algorithms are emerging. In particular, self tuning Hamiltonian methods are finding wide and successful application.

MCMC methods struggle when dealing with multi modal or discrete systems. Simulated annealing or tempering schemes have been proposed, whereby different chains operate at different temperatures, but these can be inefficient as only the zero temperature chain is valid. There is also the problem of choosing suitable temperatures, although recent work on heat baths, learning from statistical physics, may be a way forward.

Convergence criteria is a major concern. There are several methods used in practice, including visualisation of the chains, and the general recommendation seems to be ‘use them all’. It is extremely easy to find chains which seem to have converged spuriously. Repeatability is a very unreliable convergence indicator – it is easy to simply be repeating the wrong answer. Repeatability is particularly dangerous when starting from the same or similar points.

A common convergence criterion is the Gelman and Rubin method. This takes multiple chains and compares the variance and mean within each chain against the mean and variance across all chains. The overall chains have only converged when every individual chain has converged. Recent work indicates that this criterion should be applied to every dimension.

Random walk

The random walk method generates new samples by taking a random step with zero mean. A normal distribution is often used. The step size is a parameter which needs to be tuned. This method is very simple to implement, but is not at all suitable for high dimensional complex systems.

DREAM

The DREAM approach is inspired by differential evolution from evolutionary algorithms. It is a family of algorithms, and we describe the simplest case. It has been applied to problems in simulation of subsurface aquifers, which is close to the oil and gas domain.

The algorithm to generate new proposals may be described as:

- Given a step , a set of chains j and a set of samples x_{ij}**,** generate proposals
- Set $x_{ij}^{'} = x_{ij}$ for all chains
- For each chain j
    - Choose at random two other chains $m, n \neq j$
    - Set $x_{ij}^{'} = x_{ij}^{'} + \alpha\left(x_{\mathrm{im}}^{'} - x_{\mathrm{in}}^{'}\right)$
    - Perform Metropolis Hastings test to accept or reject $x_{ij}^{'}$
- End

There are several important comments to be made.

If the dimension is higher than the number of chains, the chains will be stuck in a hyperplane defined by the chains. DREAM is modified extensively to overcome this – the difference operator is applied to random subsets of the dimensions, and there is also a random walk element introduced. These each have their own tuning criteria.

The approach seems to be suitable for discrete and multi modal situations. The DREAM authors have published in this area, and it makes it attractive for our application.

The DREAM approach as described above does not allow multithreading at the chain level, as the chains are not independent. The chain loop includes an accept/reject step, so each loop depends on results of previous loops. A variant of DREAM has been modified to handle this, by generating a data store of prior samples.

There are theoretical results based on a multivariate normal distribution to determine the optimal value of $\alpha = 2.38/\sqrt{2d}$**,** where d is the number of dimensions (or, where a subset is used, the effective number of dimensions)

Hamiltonian methods

Hamiltonian methods have been around for some time, and came from the physics community. They are a way to greatly improve the exploration capabilities, by introducing an adjoint momentum variable. They are based on a Hamiltonian, and use the well known Hamiltonian equations to define a dynamic system. They may be regarded like rolling a ball around a likelihood surface.

The Hamiltonian is defined as

$$H(x, p) = -L(x) + \frac{1}{2} p^T M^{-1} p \tag{147}$$

where $L(x)$ is the likelihood function**,** p is a new momentum variable, and M is a mass matrix, which ideally corresponds to the curvature of the likelihood function.

The equations of motion are then

$$\frac{\partial x}{\partial t} = M^{-1} p \tag{148}$$

$$\frac{\partial p}{\partial t} = - \nabla_x L(x) \tag{149}$$

Note that this involves the derivative of the likelihood function.

To apply these equations of motion, we can use an integrator. The integrator must obey the rules of detailed balance (reversibility), and is commonly chosen as:

$$p\left(\tau + \frac{\epsilon}{2}\right) = p(\tau) - \frac{\epsilon}{2} \nabla_x L(x(\tau)) \tag{150}$$

$$x(\tau + \epsilon) = x(\tau) + \epsilon M^{-1} p\left(\tau + \frac{\epsilon}{2}\right) \tag{151}$$

$$p(\tau + \epsilon) = p\left(\tau + \frac{\epsilon}{2}\right) - \frac{\epsilon}{2} \nabla_x L(x(\tau + \epsilon)) \tag{152}$$

In practice, we allow the integrator to take multiple steps, and then apply the Metropolis Hastings rule.

There are three problems in practice.

- How many steps do we take?
- What value do we use for the step size ϵ ?
- What mass matrix do we use?

For detailed balance, the mass matrix must be fixed throughout. Recent work has examined higher order integrators which allow the mass matrix to change, and the equations include the determinant of the mass matrix and higher order derivatives. Another approach has been to apply BFGS to the mass matrix, based on the positions of all the chains at a given step. This seems promising, but in practice cannot handle curvature well. For a fixed, but unknown, curvature, it works remarkably well.

NUTS is a very recent algorithm which helps considerably for the first two problems. It applies a dual averaging technique to self tune the step size, and it used a concept (NUTS = No U Turn Sampler) to stop steps when they start to return to the original position. The algorithm is complex, as a tree of positions is required in order to comply with the strict MCMC rules.

In our approach, we use a diagonal mass matrix which is based on ideas common in application of the BFGS algorithm.

Synergistic method

In our work, we use a novel synergistic approach which uses all three methods (random walk, DREAM and NUTS) in a self tuning manner.

NUTS is much more computationally expensive than random walk or DREAM. It involves derivative calculations, and also a multitude of integrator steps to generate a new sample.

At each step, and for each chain, we select from the different methods at random using a weighting scheme. Each method applies its own self tuning, using dual averaging methods.

In addition, we tune the proportion of time spent using the DREAM v. NUTS method. We note that the stepsize rule in DREAM can be used to judge how well DREAM is behaving. Where the dual averaging finds a value close to the theoretical optimal value, we weight DREAM highly. If the step size falls below the theoretical optimum, we place more weight on NUTS. In this way, we can achieve an optimal performance, taking advantage of the merits of each individual method. We also allow a small proportion of random walk samples, which is always helpful to avoid sticking.

One final point is that the limitations described above concerning DREAM (e.g. the problem of hyperplanes and the need for introducing randomness) are eliminated when it is part of a synergistic solution.

Importance sampling and effective sample size

The Markov Chain Monte Carlo method is applied repeatedly, and we may use this fact to take advantage of sequential Markov Chain algorithms, sometimes called particle filtering or importance sampling. The concept is to use a previous set of samples, together with previous and revised sets of likelihoods, and generate a revised set of samples. These ideas can also be applied to the problem of data assimilation and the ensemble Kalman filter.

For iteration k, we may generate a new set of samples for iteration $k+1$. By iteration, we mean an S curve generation iteration.

For each sample x_i we have a weight

$$w_{ik} = \frac{\pi_{k+1}\left(x_i\right)}{\pi_k\left(x_i\right)} \tag{153}$$

where π_k and π_{k+1} are the previous and revised likelihoods. Note that we only know π up to a normalizing constant, but this does not affect the calculation, as we normalize the weights w_{ik}.

The simplest approach is to generate a new set of samples using weighted sampling of the previous samples with replacement with weight w_{ik}. However, this can easily result in a collapse of the set of samples, as the number of samples which propagate to the next iteration can fall drastically. This is a challenge, likewise, for EnKF methods.

The effective ensemble size is given by

$$\widehat{N}_k = \frac{1}{\sum_i \left(w_{ik}\right)^2} \tag{154}$$

where the set of weights have bee normalized. With all samples having equal weight, the effective sample size is the same as the sample size. The effective ensemble size may also be used as a diagnostic for how much the likelihood function is changing.

To avoid problems of ensemble collapse, we use a resample and move algorithm. Following the initial resampling above, we follow with a move step. This move may be one or more steps of a Markov

Chain Monte Carlo sampler. The sampler does not require a burn-in, and MCMC parameters can be reused from previous iterations.

Normal MCMC uses a small number of chains and a large number of steps. The move algorithm above can be regarded as a large number of chains (up to the number of samples) and a small number of steps (down to 1). Our implementation is adaptive. Depending on the effective ensemble size, we vary the number of MCMC steps. In addition, when the effective ensemble size falls below a threshold, we use a tempering scheme to maintain the diversity, and perform an increased number of MCMC steps.

It can be argued that, within the overall iterative process, we can maintain diversity by having a much smaller number of samples than is generated from the initial MCMC process, which may be of the order of 100,000. At present, however, we are cautious and maintain the complete set of samples.

The sampler used in the move process can be less strict that a normal sampler. It need neither be irreducible nor reversible. This may give an opportunity for investigating other samplers.

Sensitivity

Where we have no prediction points, we may express the sensitivy of the likelihood function with respect to changes in the tolerance of individual history match points. The overall likelihood is a product of likelihoods of each individual match point, so we only need to consider the sensitivy of the tolerance of a match point against the likelihood of that same match point. Importantly, we do not examine the sensitivity of an individual simlutaion run or sample, but average over the ensemble of samples.

Using, for match point j

$$L_j(x_s) = \mathit{Likelihood}_j(x_s) \tag{155}$$

and the overall likelihood

$$L(x_s) = \prod_{j=1}^{n_h} L_j(x_s) \tag{156}$$

where n_h is the number of history match points and summing over samples S, where n_S is the number of samples, we have

$$\mathit{history\ \ match\ \ sensitivity}_j = \mathit{tol}_j \times \sum_s \frac{\partial L(x_s)}{\partial \mathit{tol}_j} \tag{157}$$

$$\mathit{history\ \ match\ \ sensitivity}_j = \mathit{tol}_j \times \sum_s \left\{ \frac{\partial L_j(x_s)}{\partial \mathit{tol}_j} \times \prod_{i=1,\ i \neq j}^{n_h} L_i(x_s) \right\} \tag{158}$$

$$\mathit{history\ \ match\ \ sensitivity}_j = \mathit{tol}_j \times \sum_s \left\{ \frac{\partial L_j(x_s)}{\partial \mathit{tol}_j} \times \frac{1}{L_j(x_s)} \times L(x_s) \right\} \tag{159}$$

$$\mathit{history\ \ match\ \ sensitivity}_j = \mathit{tol}_j \times \sum_s \left\{ \frac{\partial \log(L_j(x_s))}{\partial \mathit{tol}_j} \times L(x_s) \right\} \tag{160}$$

$$history\ \ match\ \ sensitivity_j = \frac{tol_j}{n_S} \times \sum_s \frac{\partial \log\left(L_j\left(x_s\right)\right)}{\partial tol_j} \tag{161}$$

as all samples are equiprobable.

For prediction points, we may examine the derivative of the area under the S curve wrt. tolerance .

The height on the Y axis for any given sample is given by

$$Height_s = \frac{\sum_{i=1}^{s} L_i}{\sum_{i=1}^{n} L_i} \tag{162}$$

and the derivative of the height is given by

$$\frac{\partial Height_s}{\partial tol_j} = \frac{\sum_{i=1}^{s} \frac{\partial L_i}{\partial tol_j}}{\sum_{i=1}^{n} L_i} - \frac{\sum_{i=1}^{s} L_i \sum_{i=1}^{n} \frac{\partial L_i}{\partial tol_j}}{\left(\sum_{i=1}^{n} L_i\right)^2} \tag{163}$$

Assuming all sample are equiprobable

$$\sum_{i=1}^{n_s} L_i = 1 \tag{164}$$

$$\sum_{i=1}^{s} L_i = \frac{s}{n_S} \tag{165}$$

then

$$\frac{\partial Height_s}{\partial tol_j} = \sum_{i=1}^{s} \frac{\partial L_i}{\partial tol_j} - \frac{s}{n_S} \sum_{i=1}^{n_s} \frac{\partial L_i}{\partial tol_j} \tag{166}$$

and using Y_s as the response value for sample S we have an approximation, using the trapezoidal rule for integration

$$\frac{\partial Area}{\partial tol_j} = \sum_{s=1}^{n_s-1} \left(Y_{s+1} - Y_s\right) \times 0.5 \times \left(\left| \frac{\partial Height_s}{\partial tol_j} \right| + \left| \frac{\partial Height_{s+1}}{\partial tol_j} \right| \right) \tag{167}$$

The calculation of this sensitivy is fast as it does not involve any extra estimator calculations, opening the path for an interactive graphical modification of the S curve with changing tolerances, based on the derivatives alone. However, any large changes in tolerances which go beyond the applicable range for the derivatives will require additional calculations.

Bayesian Optimisation

So far we have been discussing history matching and probabilistic prediction uncertainty. Most of the techniques, however, can be used for the field optimization case. The main difference is that instead of generating an S curve and sampling from the S curve, we find the modifier values for performing the next set of simulation runs by doing an optimization using the set of proxy models.

Our overall objective may be a weighted sum of individual data points, each of which is represented by a proxy model. Expressed in terms of simulation runs we have (where *i* indicates the proxy model index):

$$Obj = \sum_i w_i Y_i \tag{168}$$

and in terms of proxy models we have

$$obj(x) = \sum_i w_i y_i(x) \tag{169}$$

We also have the variance of the objective

$$\sigma^2(x) = var(obj(x)) = \sum_i w_i var(y_i(x)) \tag{170}$$

To generate the next set of runs, we use the set of proxy models to maximize the expected improvement.

If the current best value (assuming a minimization optimization) of the objective from simulation runs is Obj_{min} then we will achieve an improvement I if $Y = Obj_{min} - I$.

The likelihood of achieving this improvement is given by the normal density function

$$\varphi = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(Obj_{min} - I - obj(x))^2}{2\sigma^2(x)}} \tag{171}$$

The expected improvement is then

$$E(I) = \int_{-\infty}^{+\infty} I \ \varphi \, dI \tag{172}$$

Using a change ofvariable, and using

$$u = \frac{Obj_{min} - obj(x)}{\sigma(x)} \tag{173}$$

we may derive

$$E(I) = \sigma(x)\left[u\Phi(u) + \varphi(u)\right] \tag{174}$$

where $\Phi(u)$ and $\varphi(u)$ are the normal cumulative distribution function and desnity function respectively.