BAYESIAN EMULATION OF GEOTHERMAL NUMERICAL MODELS: A SYNTHETIC RESERVOIR CASE-STUDY

Ariel Vidal[1]and Rosalind Archer[1]

[1]Department of Engineering Science, The University of Auckland, Private Bag 92019, Auckland 1142, New Zealand

avid587@aucklanduni.ac.nz

Keywords: *Bayesian emulators, numerical modelling, uncertainty.*

ABSTRACT

Complex numerical models are built in almost all fields of science as a means to simulate the behaviour of natural, real-world systems. These models are normally implemented as computer codes, which can take from fractions of seconds to several hours for a single run. The outputs of these models are generally used as a prediction of the real-world phenomena that are simulated by the model, but they will inevitably be faulty in some way. There will be uncertainty about how close the true quantities will be to the outputs of the numerical model and in the input parameters used for setting up the model. A formal sensitivity/uncertainty analysis for a particular numerical model may require thousands of model runs which could be impractical for complex models. In the last years there has been an increasing attention on the use of Bayesian methods to quantifying, analysing and managing uncertainty in the application of complex numerical models. Statistical emulators are being used to understand complex numerical codes and their parameter space in a wide variety of applications. Emulators based on Gaussian processes can inexpensively produce a reasonable representation of outcomes for a numerical model for a large set of potential input parameter settings without running the simulator itself, which may be valuable when the expense to run the simulator is high. In this paper a brief review of how a Gaussian process emulator works is presented together with some preliminary results on its implementation on a synthetic geothermal reservoir.

1. INTRODUCTION

Computer codes are normally used to make predictions about real-world systems in many fields of science and technology. In this paper we will refer to such a computer code as a simulator. For our purposes it is convenient to represent the simulator in the form of a function $y = f(x)$, and a run of the simulator is defined to be the process of producing one set of outputs y for one particular input configuration x . Additionally, as a prerequisite it is necessary that the simulator under study is deterministic, so, running the simulator for the same input configuration x several times it will produce the same output y. In this paper attention is focused on single scalar values for the code output, but the approximation can also be extended to multidimensional code outputs (Conti and O'Hagan, 2010; Hankin, 2012). Under a Bayesian perspective the true value of the code output is a random variable, drawn from a distribution that is conditioned by our prior knowledge, and by the previous code runs. In this sense the computer code is viewed as a random function (Hankin, 2005).

The output y of a simulator is a prediction of the real-world phenomena that is simulated by the model, but as such will inevitably be imperfect. There will be uncertainty about how close the true real-world quantities will be to the output y. This uncertainty arises from many sources, particularly uncertainty in the correct values of the input parameters and uncertainty about the correctness of the model $f(.)$ in representing the real-world system. A more detailed description of uncertainties involved in using simulators can be found in Kennedy and O'Hagan (2001).

The complexity of a simulator can become a problem when it is necessary to make many runs for different input configurations. For example, the simulator user may wish to study the sensitivity of y to variations in x, which implies a large number of simulator runs. For example, standard Monte Carlo-based methods of uncertainty/sensitivity analysis require thousands of simulator runs which become impractical when the code is costly to run (Saltelli et al., 2000).

In general terms an emulator is a statistical representation of $f(.)$ that is constructed using a training sample of simulator runs (Conti et al., 2009). The next sections present a brief review of the theory behind emulation and Gaussian processes. This is followed by an application to a simple synthetic geothermal reservoir numerical model.

2. EMULATION OF COMPLEX SIMULATORS

As was mentioned an emulator is a representation of a simulator. However, an emulator is not just a representation, but a statistical representation. An approximation to $f(.)$ for any input configuration x is a single value $\hat{f}(x)$. An emulator provides an entire probability distribution for $f(x)$. We can regard the mean of that distribution as the approximation $\hat{f}(x)$, but the emulator also provides a distribution around that mean which describes how close it is likely to be to the true $f(x)$. In fact, an emulator is a probability distribution for the entire function $f(.)$ (O'Hagan, 2006).

An important requirement is that $f(.)$ be a smooth function, so that, if we know the value of $f(x)$, we should have some idea about the value of $f(x')$ for x close to x'. It is this property of $f(.)$ that will give us the opportunity to improve on Monte Carlo sampling, since the extra information that is available after each code run is ignored in the Monte Carlo approach (Oakley and O'Hagan, 2002).

2.1 Gaussian process emulation

Next section is based mainly on the ideas of Kennedy and O'Hagan (2001) and Oakley and O'Hagan (2002). Formally, $f(.)$ is a Gaussian process if, for every $n \in \mathbb{N}$, the joint distribution of $f(x_1),..., f(x_n)$ is multivariate normal provided $x_i \in X$. In particular, $f(x)$ is normally distributed for all $x \in X$. The Gaussian process is a distribution for a function, where each point $f(x)$ has a normal distribution. This distribution is characterized by its mean function $m(.)$, where $m(x)=E\{f(x)\}$, and its covariance function $c(.,.)$, where $c(x,x')=\text{cov}\{f(x),f(x')\}$.

The mean and covariance functions should then be specified to reflect prior knowledge about *f*(*x*). In general, *m*(.) may be any function on *X* but *c*(.,.) must have the property that for every *n*=1, 2,… the variance-covariance matrix of $f(x_1), f(x_2),..., f(x_n)$ is non-negative definite for all $x_1, x_2,..., x_n \in X$.

A normal option is to model *m*(.) and *c*(.,.) hierarchically. In the case of *m*(.) a common choice is the linear model structure

$$m(x) = h(x)^{\mathrm{T}} \beta \tag{1}$$

where $h(.) = (h_1(.), h_2(.),..., h_p(.))^{\mathrm{T}}$ is a vector of *p* known regressor functions over *X* and $\beta = (\beta_1, \beta_2,..., \beta_p)^{\mathrm{T}}$ is a vector of *p* unknown coefficients. A common choice, also used in this paper, is $h(x) = (1, x)^{\mathrm{T}}$, but one is free to choose any function of *x* that could include any beliefs we might have about the form of *f*(.). As a prior distribution for β the multivariate normal distribution is a convenient choice (Kennedy and O'Hagan, 2001). This prior distribution should be specified to reflect genuine belief, but in practice prior information about parameters such as β will usually be weak.

It is possible to separate the mean and covariance by writing

$$f(x) = m(x) + e(x) = h(x)^{\mathrm{T}} \beta + e(x) \tag{2}$$

using equation (1), where $e(x)$ is a zero-mean Gaussian process with covariance function

$$\mathrm{cov}\{f(x), f(x') \,|\, \sigma^2\} = \sigma^2 c(x, x') \tag{3}$$

conditional on σ^2, where $c(x, x')$ is a correlation function that measures the correlation between *x* and *x*'. In this paper the form

$$c(x, x') = \exp\{-(x - x')^{\mathrm{T}} B (x - x')\} \tag{4}$$

will be used, where *B* is a diagonal matrix of positive roughness parameters or scales (Oakley and O'Hagan, 2002). This choice of function implies that the output is a smooth function of the inputs, which is one of the main assumptions that we make when building an emulator. Of course, equation (4) is just one possible formulation. The scales are estimated by maximizing the posterior likelihood for a given set of scale parameters. In this work optimization techniques (either Nelder-Mead or simulated annealing) are used to find the optimal scales (Hankin, 2005). If the scale value for the i-th dimension is big, even small displacements in parameter space in that dimension will result in small correlations. On the contrary, if the scale value for the i-th dimension is small, even large displacements in that dimension will have large correlations. According to examples presented in Kennedy and O'Hagan (2001), Hankin (2005) and experiments made by the authors during the validation of the model, not shown in this paper, the values used for scales (roughness parameters) and the effect of using alternative covariance structures does not seem to make much difference in practice.

For convenience, Oakley and O'Hagan (2002) assumed the normal inverse gamma distribution as the conjugate prior for β and σ^2:

$$p(\beta, \sigma^2) \propto \sigma^{-\frac{1}{2}(r+q+2)} \exp\left[\frac{-\{(\beta - z)^{\mathrm{T}} V^{-1} (\beta - z) + a\}}{(2\sigma^2)}\right] \tag{5}$$

and, for the examples given in this paper, a weak form of this prior $p(\beta, \sigma^2) \propto \sigma^{-2}$ will be used which implies an infinite prior variance of *f*(.) suggesting that there is little knowledge about the output of the computer code. Oakley (2002) gives a deep treatment to the problem of eliciting right values for *a*, *r*, *z* and *V*, using the properties of a normal inverse gamma distribution $(E(\beta) = z, \mathrm{var}(\beta) = E(\sigma^2)V, E(\sigma^2) = a/(r-2))$. As this author mentioned it is generally considered preferable to elicit beliefs about observable quantities, rather than parameters in some statistical model. This is because the expert may have difficulty in interpreting the meaning of some model parameters. Making probability statements about β and σ^2 (given the function *h*(.)) will automatically imply probability statements about *f*(.) through the Gaussian process model. The underlying technique in the elicitation process is to ask the expert for probability statements about the observable quantity *f*(*x*) in the form of percentiles of *f*(*x*), and then finding *a*, *r*, *z* and *V* such that the implied percentiles through the Gaussian process model are similar. It is important to mention that, as was stated by Oakley (2002), since we ask the expert for statements about observables, and then find an appropriate prior to match these statements, we cannot assert that the Gaussian process prior is a perfect representation of the expert's uncertainty. We can only find hyperparameters such that the Gaussian process prior resembles the expert's judgments as closely as possible, and so we think of the chosen prior distribution resulting from the elicitation as being a 'fitted prior'.

The next step is to condition the random function *f*(.) at specific points in the parameter space. This set of points is known as the experimental design or design matrix and they represent points where our code will be run and used for calibration of the Gaussian process model. In this paper a maximin Latin hypercube design is used. These designs give good coverage of the space and are evenly distributed in each one-dimensional projection (Kennedy and O'Hagan, 2001). With the specified prior, and an experimental design $D = \{x_1,..., x_n\}$ on which *f*(.) has been evaluated giving $d = f(D)$, it can be shown (Oakley and O'Hagan, 2002) that:

$$\left.\frac{f(x) - m^*(x)}{\hat{\sigma}\sqrt{c^*(x,x)}}\right| d, B \sim t_{r+n} \tag{6}$$

Hence, as equation (6) states, the outputs corresponding to any set of inputs will now have a multivariate *t* distribution, where the main components of this equation can be calculated according to the next expressions:

$$m^*(x) = h(x)^{\mathrm{T}} \hat{\beta} + t(x)^{\mathrm{T}} A^{-1} (d - H\hat{\beta}), \tag{7}$$

$$\begin{aligned} c^*(x, x') &= c(x, x') - t(x)^{\mathrm{T}} A^{-1} t(x') + \\ &\{h(x)^{\mathrm{T}} - t(x)^{\mathrm{T}} A^{-1} H\} V^* \{h(x')^{\mathrm{T}} - t(x')^{\mathrm{T}} A^{-1} H\}^{\mathrm{T}}, \end{aligned} \tag{8}$$

Further terms needed to account for equations (7) and (8) may be derived from:

$$t(x)^{\mathrm{T}} = (c(x,x_1),...,c(x,x_n)), \tag{9}$$

which is a vector of data- to-unknown correlation values.

$$H^{\mathrm{T}} = (h^{\mathrm{T}}(x_1),...,h^{\mathrm{T}}(x_n)), \tag{10}$$

$$A = \begin{pmatrix} 1 & c(x_1,x_2) & \cdots & c(x_1,x_n) \\ c(x_2,x_1) & 1 & & \vdots \\ \vdots & & \ddots & \\ c(x_n,x_1) & \cdots & & 1 \end{pmatrix}, \tag{11}$$

Where matrix A corresponds to the correlation matrix of data-to-data samples used for the construction of the emulator.

$$\hat{\beta} = V^*(V^{-1}z + H^{\mathrm{T}}A^{-1}d), \tag{12}$$

$$\hat{\sigma}^2 = \frac{a + z^{\mathrm{T}}V^{-1}z + d^{\mathrm{T}}A^{-1}d - \hat{\beta}^{\mathrm{T}}(V^*)^{-1}\hat{\beta}}{n + r - 2}, \tag{13}$$

$$V^* = (V^{-1} + H^{\mathrm{T}}A^{-1}H)^{-1}, \tag{14}$$

$$d^{\mathrm{T}} = (f(x_1),..., f(x_n)). \tag{15}$$

So, the covariance between any two outputs may be calculated by equation (8) and an estimation of uncertainty given by $\hat{\sigma}\sqrt{c^*(x,x)}$. The former equations are consistent in that the estimated value for points actually in the design matrix is in fact the observations (with zero error). It may similarly be shown that $c^*(x,x) = 0$ for $x \in D$, as expected: the emulator should return zero error when evaluated at a point where the code output is known. For full details of the prior to posterior analysis please refer to O'Hagan (1994).

3. EMULATION OF A SYNTHETIC GEOTHERMAL RESERVOIR

In order to illustrate the construction and use of an emulator, in the next section a synthetic geothermal reservoir will be modelled with reference input parameters and then, through emulation, we show how it is possible to get a set of cheap output model approximations that correlate well with the actual outputs of the simulator.

3.1 Model setup

With the purpose of testing the Gaussian process emulation technique described in the previous chapter a simple synthetic model of a rectangular geothermal reservoir was used. This reservoir consists of three different horizontal layers with permeability values of -12,-15 and -13 from top to bottom in $\log_{10}(\mathrm{m}^2)$ units. The modelled reservoir size is 25x20x4 km and the grid used for simulation is a regular one composed of 25x20x20 blocks in the x-y-z directions with a block size of 400x400x200 m, comprising a total of 10,000 blocks. Boundary conditions at the bottom of the model correspond to a centred rectangle of 19x14 blocks with a heat flux of 300 mW/m^2 and no heat flux generators for the blocks outside this zone were defined. In the same way no mass upflows were used. Closed lateral boundary conditions were defined for the model. Equal values of permeability in the x and y directions were assumed. The z direction permeability was a fraction of the x direction permeability (K_x=K_y=10K_z). With this configuration a reference TOUGH2 (Pruess, 1990) run was carried out to establish the natural-state of the system. Ten vertical wells are spread across the reservoir which later were used for extracting temperature values and hence comparing different runs by means of an objective function. This objective function corresponds to our function *f*(.) which we will try to reproduce through emulation.

3.2 Design Matrix

The first step to build an emulator is to choose input configurations $x_1, x_2, ..., x_n$ at which to run the simulator to get training data. As could be expected the accuracy of the emulator will depend on the set of sample points selected. There is considerable literature on the design of computer experiments for interpolating the simulator or for uncertainty analysis of the code, for a comprehensive review please refer to Santner et al. (2003). As was mentioned we will use the statistical approach of Latin hypercube sampling based on maximin criteria which maximize the minimum distance between points, but place the points in a randomized location within its interval. Three different set of input points were used, each one with 30, 50 and 100 samples, respectively, to test the influence of the number of points in the accuracy of the results. Adaptive sampling algorithms have been also used (Pau et al., 2013) but, as O'Hagan (2006) has noticed it is not clear whether more sophisticated sampling algorithms would yield much improvement in the emulator's accuracy.

The value of permeability for each one of the three different layers of the synthetic reservoir is considered uncertain in the analysis. As a prior belief we specified that the ranges for the logarithm of the permeability in each layer are [-13, -11], [-16, -14] and [-14, -12] for the upper, intermediate and lower layer, respectively. The output of interest, *f*(*x*), is the scalar value of the objective function built as the squared sum of residuals between the values of temperature in the ten wells in the reference model, described in the previous section, and the temperature values in the same ten wells generated by new runs of the simulator built with the different set of points of each design matrix. Figure 1 presents the distribution in space of sample points for the three training sets generated. As can be seen in this simple three-dimensional case with just 30 samples it is clear that there are wide zones in the input parameter space that are not being sampled and hence represented in the training phase of the emulator. This under-representation could produce uncertain results of the emulator approximation when it is evaluated inside these areas of the input parameter space. It seems that, at least for this 3-dimensional scenario, 100 samples represent a conservative number of runs to include a considerable portion of the input parameter space. Also, as can be expected, the values for the objective function shows lower values closer to the values of permeability used for the reference model and higher values if we begin to move away from the reference values. Additionally, it is important to mention that it is not straightforward to compare between the three different sets of training points as each design matrix samples from specific locations which could not be replicated in a new set of points, even when the new set includes a bigger number of samples. So, this restriction will influence the behaviour of the emulator as it will be trained with samples in different locations. A sequential sampling strategy may

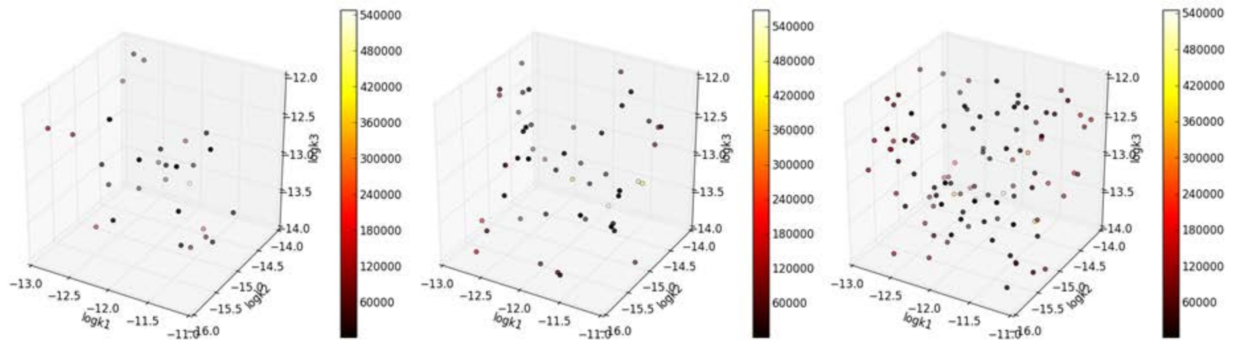

Figure 1: Distribution of sampling points from three different sets of 30, 50 and 100 samples from a Latin hypercube sampling. Units in three axes represents logarithm of permeability values, with logk1, logk2 and logk3 the values of permeability for the upper, intermediate and lower layer respectively in the synthetic model. Also values for the objective function are showed for each point used for the training.

overcome this obstacle and make the effect of the number of sampling points clearer as a set with a bigger number of samples will include all the points from previous sets.

Histograms with values for the objective function used for training in each scenario can be seen in Figure 2. The first feature that appears evident is that in the three cases the values of the objective function tend to be clustered into low values with a weak tail towards high values. More than 80 percent of the samples for each training set are below the value of 100,000 which leaves underrepresented the output space for high values. Special care must be taken into account when trying to infer conclusions with high values in the output space.

3.3 Results and discussion

The next sets of results were obtained using the R (R core team, 2014) package Emulator (Hankin, 2005) which implements the ideas of Oakley and O'Hagan (2002). A stationary covariance function and for the prior mean a regression function of the form $h(x) = (1, x)^{\mathrm{T}}$ were used which is a simple approach but it has been used successfully in previous works for its simplicity and reliability (Oakley and O'Hagan, 2002; Kennedy et al., 2006). Other forms of the prior mean could be used if they would provide a more accurate estimation of the function we are emulating.

Figure 3 shows a comparison between observed vs predicted values of the objective function for the three different emulators tested against a data set of 200 random samples for which we already have run the simulator. As could be expected the best performance was produced by the emulator built with a 100 training samples which, as a whole, seems to produce a good representation of the simulator behaviour. A second important point, as was described previously, is the clustering of objective function values into the lower range of values where it is possible to see better results than for high values. In the case with 30 training samples we see that the emulator produces values that are both, lower and higher than the actual values of the simulator, but in the case with 50 training samples it seems that the emulator tends to produce values that are lower than those obtained by running the simulator. Moreover, these underestimated samples possess the bigger errors. If we were tempted to discard these values the performance of the emulator appears to correctly reproduce the test data set. One simple explanation for these errors could be due to the dissimilar distribution of point samples in the input parameter space between the training and test set of samples, where points with the poorest fit must be those located more away from training samples. Another feature, without an easy explanation, is the presence of negative emulated values for the objective function which are not allowed in the output space of the simulator as it is built as a sum of strictly positive values. These kinds of situations could be avoided including restrictions on the permitted output space for the emulator.

The increase in the number of training samples also has effects on the distribution and values of the standard deviation of each emulated value. Figure 4 presents histograms with the values of the standard deviation vs frequency for each emulator. In general they tend to cluster into low values, below 10,000, situation which is stronger as we increase the number of training samples where the errors are mainly concentrated below 5,000. The highest values over 40,000 are quite scarce and just represent outliers inside the population. In most cases these errors represent a small fraction of the value being emulated (<5%).

Although the main objective of this work is not to present a solution to the calibration/inverse problem of a numerical model, an easy task could be to compare the values of permeability of the lowest objective function values obtained through emulation and simulator runs with the actual values used in the reference model presented in Section 3.1. As can be seen in Table 1 eight of ten points with the lowest values for the objective function are present in both models (simulator vs emulator) which is a good indicator of consistency between both models. A result which is not so evident from Table 1 is that values in this table for both models are not necessarily the closest points to the actual values declared for permeability in the reference model, which points to the non-uniqueness nature of the inverse/calibration problem. A deeper understanding of the physics of the process under study and how the simulator works should elucidate some clues for helping to understand this source of uncertainty. For a thorough Bayesian treatment of the calibration problem please refer to Kennedy and O'Hagan (2001).

The gain in time by using the emulator in this example is quite impressive as each run of the simulator takes an average of 300 seconds, depending on the input parameter configuration used, and the whole evaluation of 200 points in the test data set takes less than 1 second. The main computational bottleneck in the process is the inversion of

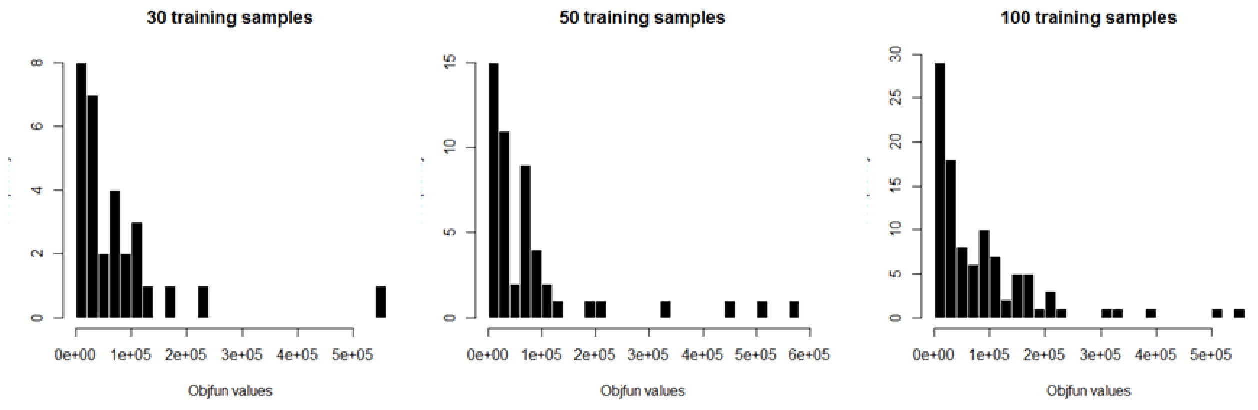

Figure 2: Histograms with values of the objective function vs frequency used for training for each one of the three different set of points used for the construction of an emulator.

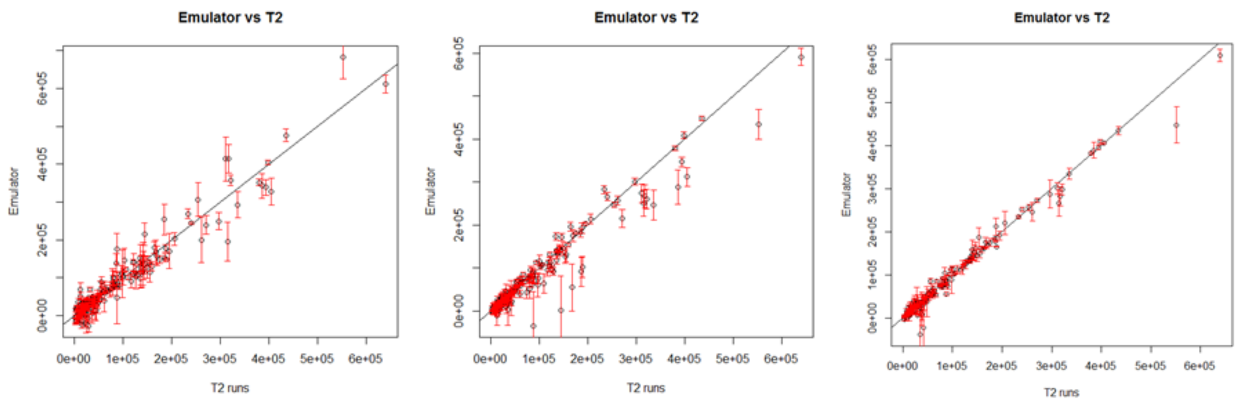

Figure 3: Values for the objective function obtained by the emulator vs actual values obtained by simulator runs (T2 axes) for a set of 200 random samples in the input parameter space. From left to right there is an increase in the number of training samples from 30 to 50 to 100. Errors bar are 95% confidence intervals.

the matrix A in equation (11), but this only has to be done once per emulator (Hankin, 2005). This saving in time is even more pronounced for bigger and more complex models which can take several hours or days for only a single run.

4 FUTURE WORK

This work represents the first attempts made by the authors in the direction of including Bayesian techniques in the analysis of geothermal numerical models and sources of uncertainty. As such, there are still some points that require more attention and analysis.

More sampling techniques must be tested and incorporated in the analysis; the sequential sampling algorithms used by Pau et al. (2013) are of particular interest for a better understanding of the influence of the number of training points in the emulator's performance.

The choice of a linear regression function $h(x) = (1, x)^{\mathrm{T}}$ as a prior belief of the mean could be refined and new forms for this function could be explored. The selection of this or another type of function will depend on the nature and complexity of the problem under study, as a simple expression that represents previous knowledge about the function appears unlikely when using complex simulators such as TOUGH2. In this same line, the assumption of stationarity is a modeller decision and more complex expressions for the covariance or even the use of multiple covariance functions for different zones in the input parameter space will depend of each problem and smoothness of the function being emulated.

The presented methodology treats the computer code as a black-box. There is no use of information about the mathematical model implemented by the code. New methods that open this black box will be more powerful, but, certainly the resulting approach will be more complex to apply.

The main future objective is being able to emulate a complex model in a high-dimensional input parameter space, as should be expected when modelling a natural geothermal reservoir. It is important to mention that emulation is rarely an end in itself; the purpose of building an emulator is almost always to facilitate other calculations that would not be practical to do using the simulator itself mainly by time or computational restrictions (O'Hagan, 2006). One key point is that to build the emulator only one set of runs of the simulator is used. After the emulator is built, we do not need any more simulator runs, no matter how many analyses are required of the simulator's behaviour. Uncertainty and sensitivity analysis based on Monte-Carlo techniques may be conducted inexpensively by using the emulator. A more ambitious task is to include observations of the real-world process for a Bayesian calibration of the model under study.

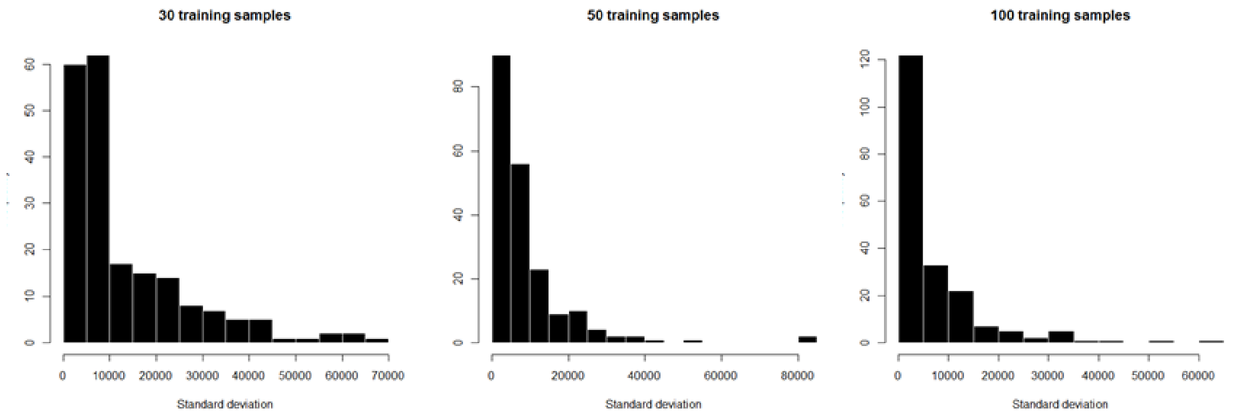

Figure 4: Histograms with values of standard deviation vs frequency for the three different set of points used for the construction of an emulator.

Simulator runs				Emulator results			
k1	k2	k3	Obj fun	k1	k2	k3	Obj fun
-11.905	-15.685	-13.135	2403.189	-12.085	-14.985	-12.585	2045.376
-11.955	-14.795	-12.795	2472.678	-12.055	-15.825	-13.115	2500.268
-11.965	-15.695	-13.015	2487.069	-11.905	-15.685	-13.135	3166.314
-12.105	-15.225	-13.065	2524.799	-12.045	-15.715	-13.155	3493.057
-11.715	-15.805	-13.095	2764.758	-11.715	-15.805	-13.095	3586.244
-12.085	-14.985	-12.585	4739.452	-11.955	-14.795	-12.795	3646.845
-12.015	-14.755	-12.645	4805.511	-11.135	-15.505	-13.025	4420.304
-12.045	-15.715	-13.155	4985.419	-12.115	-15.815	-13.175	4551.697
-12.055	-15.825	-13.115	5014.328	-11.965	-15.695	-13.015	4818.394
-11.505	-15.185	-13.145	6555.417	-12.015	-14.755	-12.645	5747.820

Table 1: Ten lowest values for the objective function in the test data set for simulator runs and emulated values using the 100 training samples emulator. Please remember that the reference model was built with values of -12, -15 and -13 for k_1, k_2 and k_3, respectively.

5 CONCLUSIONS

This paper has shown how through emulation it is possible to get a statistically rigorous and cheap approximation to a simulator output in a geothermal setting. The posterior approximations obtained by using the emulator are orders of magnitude faster than running the simulator with the same points and a strong confidence in the results can be achieved after a successful training phase.

The current global tendency of bigger and more complex numerical models in environmental sciences places a growing importance on the use of fast and accurate approximations that can be applied with tasks that require multiple runs of the simulator.

There are still some aspects that required more attention and analysis but the current results encourage the usage of Bayesian techniques for a deeper analysis of the behaviour of simulators currently used in the geothermal industry, their sources of uncertainty and how to extract the maximum value from the outputs that we are getting.

REFERENCES

Conti, S., Gosling, J. P., Oakley, J. E., O'Hagan, A.: Gaussian process emulation of dynamic computer codes. *Biometrika, 96*(3), 663-676. (2009).

Conti, S. and O'Hagan, A.: Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference, 140*(3), 640-651. (2010).

Hankin, R.: Introducing BACCO, an R bundle for Bayesian analysis of computer code output. *Journal of Statistical Software, 14*. (2005).

Hankin, R.: Introducing multivator: A multivariate emulator. *Journal of Statistical Software, 46*(8). (2012).

Kennedy, M. C. and O'Hagan, A.: Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63*(3), 425-464. (2001).

Kennedy, M. C., Anderson, C. W., Conti, S., O'Hagan, A.: Case studies in Gaussian process modelling of computer codes. *Reliability Engineering and System Safety, 91*(10-11), 1301-1309. (2006).

Oakley, J.: Eliciting Gaussian process priors for complex computer codes. *Journal of the Royal Statistical Society: Series D (The Statistician), 51*(1), 81-97. (2002).

Oakley, J. and O'Hagan, A.: Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika, 89*(4), 769-784. (2002).

O'Hagan, A.: Kendall's Advanced Theory of Statistics, 2B, Bayesian Inference. London: Edward Arnold. (1994).

O'Hagan, A.: Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering and System Safety, 91*(10-11), 1290-1300. (2006).

Pau, G., Zhang, Y. Q. and Finsterle, S.: Reduced order models for many- query subsurface flow applications. *Computers & Geosciences., 17*(4), 705-721. (2013).

Pruess, K.: TOUGH2– A General Purpose Numerical Simulator for Multiphase Fluid and Heat flow. Report: LBL-29400, Lawrence Berkeley Laboratory, Berkeley, California. (1990).

R Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/. (2014).

Saltelli, A., Chan, K., Scott, E.M.: Sensitivity analysis. New York: Wiley. (2000).

Santner, T. J., Williams, B., Notz, W.: The design and analysis of computer experiments. New York: Springer. (2003).