

## TREATMENT OF MODEL UNCERTAINTY IN MODEL CALIBRATION

Laura P. Swiler<sup>a</sup> and Timothy G. Trucano<sup>a</sup>

<sup>a</sup>*Optimization and Uncertainty Estimation Dept.*

*Sandia National Laboratories\*, Albuquerque, NM 87185*

[lpswile@sandia.gov](mailto:lpswile@sandia.gov); [tgruca@sandia.gov](mailto:tgruca@sandia.gov);

### Abstract

The problem of model calibration is often formulated as finding the parameters that minimize the squared difference between the model-computed data (the predicted data) and the actual experimental data. This approach does not allow for explicit treatment of uncertainty or error in the model itself: the model is considered the “true” deterministic representation of reality. While this approach does have utility, it is far from an accurate mathematical treatment of the true model calibration problem in which both the computed data and experimental data have error bars. Our proposed research focuses on methods to perform calibration accounting for the error in both the computer model and the data, as well as improving our understanding of its meaning for model predictability. We call this approach Calibration under Uncertainty (CUU). This talk presents our current thinking on CUU. We outline some current approaches in the literature, and discuss the Bayesian approach to CUU in detail.

### Introduction

Recent research in the Bayesian statistics community has yielded advances in formal statistical methods that address Calibration under Uncertainty. One approach is that of Kennedy and O’Hagan (2001), hereafter referred to as KOH. They formulate a model for calibration data that includes an experimental error term (similar to standard regression) and a model discrepancy term, with a Gaussian process chosen to model the discrepancy. They then use a Bayesian approach to update the statistical parameters associated with the discrepancy term and with the model parameters. The purpose of updating is generally to reduce uncertainty in the parameters through the application of additional information. Reduced uncertainty increases the predictive content of the calibration, or that is the expectation. We discuss several issues relating to the Bayesian approach: First, is the Gaussian prior the correct one, or the most effective choice for complex computational science and engineering (CSE) models? Second, the mathematics behind this approach involves covariance matrices of joint input variable distributions. Estimating the full joint posterior distribution therefore requires complicated integration and so techniques like Markov Chain Monte Carlo (MCMC) sampling are used to approximate the posterior distributions on the hyperparameters which govern the prior distribution. We discuss MCMC methods and their suitability to the CUU problem. Finally, the approach of Kennedy and O’Hagan assumes that the computer simulation model is deterministic: re-running the model with the same set of inputs produces the same output. We propose extending their framework to allow for stochastic simulation models.

---

\* Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

### Gaussian Process Model Formulation

For those unfamiliar with Gaussian processes (GP), this section provides a very brief introduction. Gaussian Process models are used in response surface modeling, especially response surfaces which “emulate” complex computer codes. Gaussian processes have also been used in conjunction with Bayesian analysis for regression problems and for calibration. Gaussian processes have also been widely used for estimation and prediction in geostatistics and similar spatial statistics applications [Cressie].

A Gaussian process is defined as follows [Williams]: A stochastic process is a collection of random variables  $\{Y(\mathbf{x}) \mid \mathbf{x} \in X\}$  indexed by a set  $X$  (in most cases,  $X$  is  $\mathfrak{R}^d$ , where  $d$  is the number of inputs). The stochastic process is defined by giving the joint probability distribution for every finite subset of variables  $Y(\mathbf{x}_1), \dots, Y(\mathbf{x}_k)$ . A Gaussian process is a stochastic process for which any finite set of  $Y$ -variables has a joint multivariate Gaussian distribution. A GP is fully specified by its mean function  $\mu(\mathbf{x}) = E[Y(\mathbf{x})]$  and its covariance function  $C(\mathbf{x}, \mathbf{x}')$ . The basic steps in defining/using a GP are:

1. Define the mean function. The mean function can be any type of function. Often the mean is taken to be zero, but this is not necessary. A common representation, for example in a regression model, is that  $y(x) = \sum_j w_j \phi_j(x) = \mathbf{w}^T \boldsymbol{\phi}(x)$ , where  $\{\phi_j\}$  is a set of fixed basis functions and  $\mathbf{w}$  is a vector of weights. Combining Gaussian process and a Bayesian approach, one places a prior probability distribution over possible functions and lets the observed data transform the prior into a posterior.
2. Define the covariance. There are many different types of covariance functions that can be used. At this stage, we shall focus on stationary covariance functions where  $C(\mathbf{x}, \mathbf{x}')$  is a function of  $\mathbf{x} - \mathbf{x}'$  and is invariant to shifts of the origin in the input space. A commonly-used covariance function (KOH) is:

$$C(\mathbf{x}, \mathbf{x}') = v_o \exp\left\{-\sum_{u=1}^d \rho_u^2 (\mathbf{x}_u - \mathbf{x}'_u)^2\right\} \quad (1)$$

This covariance function involves the product of  $d$  squared-exponential covariance functions with different lengthscales on each dimension. The form of this covariance function captures the idea that nearby inputs have highly correlated outputs.

3. Perform the “prediction” calculations. Given a set of  $n$  input data points  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  and a set of associated observed responses or “targets”  $\{z_1, z_2, \dots, z_n\}$ , we use the GP to predict the target  $z_{n+1}$  at a new set of inputs  $\mathbf{x}_{n+1}$ . The target is usually represented as the sum of the “true” response,  $y$ , plus an error term:  $z_i = y_i + \varepsilon_i$ , where  $\varepsilon_i$  is a zero mean Gaussian random variable with constant variance  $\sigma_\varepsilon^2$ . We assume that the prior distribution on the  $y_i$ 's is given by a GP defined as  $Y \sim N(\mathbf{0}, \mathbf{K})$ , where  $\mathbf{K}$  is the  $n \times n$  covariance matrix with entries  $K_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$ . Then the prior distribution on the targets  $z_i$  is  $N(\mathbf{0}, \mathbf{K} + \sigma_\varepsilon^2 \mathbf{I}_n)$ . The distribution of the predicted term  $z_{n+1}$  is conditional on the data  $\{z_1, z_2, \dots, z_n\}$ . It is Gaussian with the following mean and variance:

$$E[z_{n+1} \mid z_1, z_2, \dots, z_n] = \mathbf{k}^T \mathbf{C}^{-1} \mathbf{z} \quad (2a)$$

$$\text{Var}[z_{n+1} \mid z_1, \dots, z_n] = C(\mathbf{x}_{n+1}, \mathbf{x}_{n+1}) - \mathbf{k}^T \mathbf{C}^{-1} \mathbf{k} \quad (2b)$$

where  $\mathbf{k}$  is the vector of covariance between the  $n$  known targets and the new  $n+1$  data point:  $\mathbf{k} = (C(\mathbf{x}_1, \mathbf{x}_{n+1}), \dots, C(\mathbf{x}_n, \mathbf{x}_{n+1}))^T$ ,  $\mathbf{C}$  is the  $n * n$  covariance matrix of the original data, and  $\mathbf{z}$  is the  $n * 1$  vector of target values.

The equations (2) for the mean and variance of the predictive distribution for  $z_{n+1}$  both require the inversion of  $\mathbf{C}$ , an  $n \times n$  matrix. In general, this is a  $O(n^3)$  operation. Neal (1997) and Williams (2002) claim that this is feasible on modern computers when  $n$  is the order of a few hundred, but that it becomes computationally expensive when  $n$  is larger than 1000.

4. Use Monte Carlo Markov Chain (MCMC) sampling to generate posterior distributions on the hyperparameters which govern the covariance function (and the mean function). A common approach in GP is to assume all GPs are zero mean, so the Bayesian updating only involves hyperparameters governing the covariance function. Since these may be quite complex, one usually still needs a MCMC sampling method to generate the posterior. For example, Neal assumes the  $\rho^2$  terms in the covariance function are distributed as gamma distributions (which themselves are governed by three parameters), so one needs to calculate/update these three parameters for every  $\rho^2$  term.

The reader should note: We do not address the issue of Bayesian analysis/Bayesian updating in this paper due to space limitations. Gelman, Carlin, Stern and Rubin (1995), Press (2003) et al. provides a good primer on Bayesian topics; specific references for MCMC include Gilks, Richardson and Spiegel (1996) and Gamerman (1997). The basic concept in Bayesian analysis is to combine “prior” information (in terms of a distribution on a parameter, where the distribution itself is characterized by “hyperparameters”) and actual data (through a likelihood function) to obtain a “posterior” estimate of various parameters.

### **Kennedy and O’Hagan formulation**

With this background in Gaussian process models and Bayesian analysis, we proceed to the KOH model for calibration. KOH assume that the calibration inputs are supposed to take fixed but unknown values  $\boldsymbol{\theta} = (\theta_1, \dots, \theta_{q2})$ . The output of the computer model when the variable inputs are given values  $\mathbf{x} = (x_1, x_2, \dots, x_{q1})$  and when the calibration inputs are given values  $\mathbf{t} = (t_1, t_2, \dots, t_{q2})$  is denoted by  $\eta(\mathbf{x}, \mathbf{t})$ . KOH differentiate between the unknown value  $\boldsymbol{\theta}$  of the calibration inputs which we wish to determine (calibrate) and a known particular set of their values,  $\mathbf{t}$ , which we set as inputs when running the model. The “true” value of the real process when the variable inputs take value  $\mathbf{x}$  by  $\zeta(\mathbf{x})$ . The code outputs from  $N$  runs of the computer code are represented as  $y_j = \eta(\mathbf{x}_j, \mathbf{t}_j)$ . The observed data (consisting of  $n$  points, where  $n < N$  usually) is denoted as  $\mathbf{z} = (z_1, z_2, \dots, z_n)^T$ . In KOH’s formulation, they represent the relationship between the observations, the true process, and the computer model output by the equation:

$$z_i = \zeta(\mathbf{x}_i) + e_i = \rho \eta(\mathbf{x}_i, \mathbf{t}_i) + \delta(\mathbf{x}_i) + e_i \quad (3)$$

where  $e_i$  is the observation error for the  $i^{\text{th}}$  observation,  $\rho$  is an unknown regression parameter, and  $\delta(\mathbf{x})$  is a model discrepancy or model inadequacy function that is independent of the code output  $\eta(\mathbf{x}, \mathbf{t})$ .

A few comments: this is a highly parameterized model, with both the code output  $\eta(\mathbf{x}, \mathbf{t})$  and  $\delta(\mathbf{x})$  represented as a Gaussian process. The error term  $e_i$  should include both residual variability as well as observation error, but KOH do not use replicated points and their model is deterministic, so they do not strictly address residual variability. They assume that  $e_i$  is normally distributed as  $N(0, \lambda)$ . Assumption of a constant value of  $\rho$  implies that the underlying process  $\zeta(\mathbf{x})$  is stationary.

In the approach recommended by KOH, the prior information about  $\eta(\mathbf{x}, \mathbf{t})$  and  $\delta(\mathbf{x})$  is given by Gaussian processes:  $\eta(\mathbf{x}, \mathbf{t}) \sim N(m_1(\mathbf{x}, \mathbf{t}), c_1((\mathbf{x}, \mathbf{t}), (\mathbf{x}', \mathbf{t}')))$  and  $\delta(\mathbf{x}) \sim N(m_2(\mathbf{x}), c_2(\mathbf{x}, \mathbf{x}'))$ . KOH assume that the mean functions are:  $m_1(\mathbf{x}, \mathbf{t}) = \mathbf{h}_1(\mathbf{x}, \mathbf{t})^T \boldsymbol{\beta}_1$  and  $m_2(\mathbf{x}) = \mathbf{h}_2(\mathbf{x})^T \boldsymbol{\beta}_2$ . If a noninformative prior is assumed,  $p(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2) \propto 1$ . KOH then formulate all of the hyperparameters relating to this problem. They denote  $(\rho, \lambda, \psi)$  by  $\phi$ , where they state that  $\psi$  represents some “further hyperparameters” relating to the covariance functions. Finally, KOH assume that the prior distribution takes the form:

$$p(\boldsymbol{\theta}, \boldsymbol{\beta}, \phi) = p(\boldsymbol{\theta})p(\phi) \quad (4)$$

because of the weak prior distribution on  $\boldsymbol{\beta}$  and assumptions of independence.

The details of calculating the full joint posterior distribution  $p(\boldsymbol{\theta}, \boldsymbol{\beta}, \phi | \mathbf{d})$  are given in KOH; space does not permit reproducing them here. The important thing to note is that this joint posterior density is a Gaussian process, with a complex mean and variance structure. The covariance matrix of the posterior involves four “submatrices” which depend on the correlation structure of the individual Gaussian processes  $\eta(\mathbf{x}, \mathbf{t})$  and  $\delta(\mathbf{x})$ . The posterior distribution is not tractable to calculate analytically. Even with simplification, it would require a high-dimensional quadrature to integrate  $p(\boldsymbol{\theta}, \boldsymbol{\beta}, \phi | \mathbf{d})$  over  $\boldsymbol{\beta}$  and  $\phi$  to obtain the posterior estimate for the calibration parameters  $p(\boldsymbol{\theta} | \mathbf{d})$ . KOH address this by fixing many of these parameters and use a two stage process, where they estimate the hyperparameters relating to the covariance matrix for the model term,  $c_1$ , separately and before estimating the hyperparameters relating to the covariance matrix of the discrepancy term,  $c_2$ .

### Implementation Issues

We are investigating the feasibility of using a GP formulation such as provided by KOH as a practical calibration method for engineering design problems. Katherine Campbell has also looked at KOH’s work with an emphasis on implementation [Campbell, 2002]. She concluded that information about model quality gained through the formulation of a model discrepancy term could be useful, but that a user should be careful in situations of limited observational data and “avoid exaggerating the contribution of the Bayesian updating process.”

We started by developing a GP emulator for the Rosenbrock function (see Figure 1), a standard function used in testing optimization algorithms. This function is given by:

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

This exercise was interesting because we took an initial set of 11 samples (generated with a Latin Hypercube sampling routine), then we took a larger set of 110 LHS samples. For the 11 sample set, we were able to generate a Gaussian process and perform the calculation of the expected value with confidence bounds at a new input point with no difficulty.

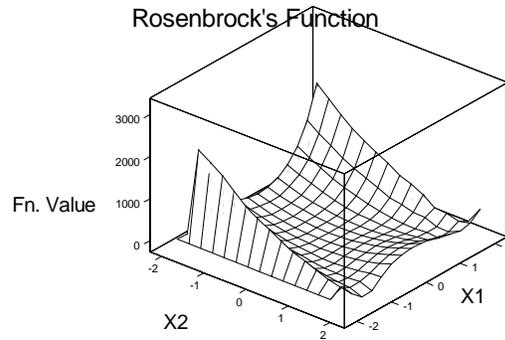


Figure 1. Rosenbrock's Function

However, with the 110-point data set, the covariance matrix  $\mathbf{C}$  became extremely ill-conditioned, and the inverse covariance matrix  $\mathbf{C}^{-1}$  needed for the prediction calculations could not be generated (the ratio of largest and smallest eigenvalues in the covariance matrix was  $10^{16}$ !) Normalization of the output is also critical: one has to subtract the mean and divide by the standard deviation of the original output data to obtain normalized values for use in the GP model. Figures 2 and 3 show the two GPs against one of the input dimensions. Note that there are no error bounds in Figure 3, because of the inability to invert the covariance matrix. Note that this behavior intuitively seems wrong: more data should always be better in terms of creating a response model or performing prediction. But in GP models, if points are close together in the input space, the resulting covariance matrix can have rows that are nearly linearly dependent, and the inversion falls apart. There are methods to address this (e.g., singular value decomposition), but our experience stands as a caution.

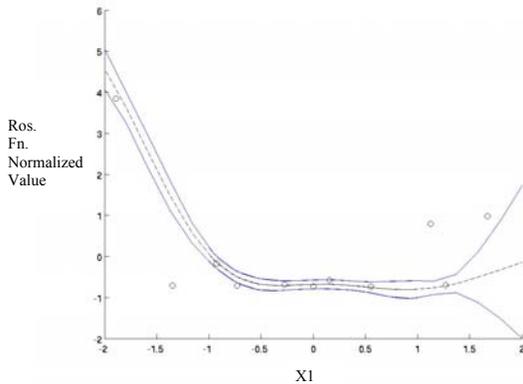


Figure 2. GP Model based on 11 data pts.

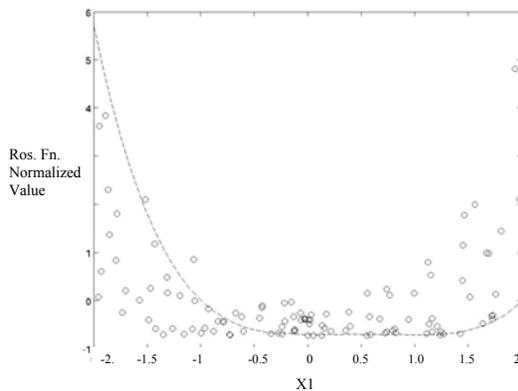


Figure 3. GP Model based on 110 data pts.

The next step is to add “noise” to the Rosenbrock function and perform the same type of GP emulation as shown above, but with the full KOH equation (3) including a model emulator GP and a model discrepancy GP. Thus far, we have found that it is difficult to separate the model discrepancy and the observational error term,  $e_i$ , unless one has very good information about measurement error in their system. Also, the framework posed by KOH requires that the user have reasonably good prior estimates for both the model emulator and the model discrepancy GP. In practice, this is not always the case. Finally,

there is a software implementation issue. Most public domain software (e.g., Netlab, FBM), allow one to create a GP and update the hyperparameters governing the GP, usually with a MCMC method. However, the ability to perform simultaneous updating on two GP models coupled by equation (3) is not generally available. From a software perspective, any automated approach would try and take the two GPs, sum their mean and variance terms to consolidate the updating down to one GP model which defeats our goal of having separate parameter estimates for each.

The calculation of the joint pdf (4) requires careful consideration in terms of what assumptions to make to simplify the calculation. The MCMC methods which are commonly used for these types of problems require the user to have a fair amount of statistical knowledge about the form of the posterior (in terms of a “proposal distribution” used to generate the Markov chain), certainly in order to make evaluation more efficient. MCMC methods require a lot of tuning parameters, such as step sizes and leaping parameters, and it is not trivial to tune the MCMC to obtain a recommended acceptance rate of 25%, for example. Finally, testing convergence of MCMC methods is difficult. There are some convergence diagnostics available [Gilks, Richardson and Spiegelhalter, 1996] but they test if the chain has “settled” out and do not really test if the Markov chain has converged to the “true” underlying posterior distribution. We have tested cases where two different chains produced substantially different posterior distributions. It appears to us that tuning MCMC performance and improving convergence diagnostics is a research question of interest.

## Conclusions

Overall, our conclusions to date from this work in progress are the following: Gaussian process models are powerful emulators. Implementing them requires some knowledge about the data set used and what data will have to be discarded to make the covariance matrix well-conditioned; or additional formulations are needed that are robust to poor covariance conditioning. KOH’s formulation of “observation = model + discrepancy + error” is very important because it explicitly separates the model discrepancy term from the model itself. The Gaussian process assumption of the model discrepancy needs further examination, but in general, GP models are extremely flexible at representing a wide variety of functional relationships. The additional assumption that these GP models are governed by parameters that can be updated using Bayesian methods adds a great deal of computational complexity to the picture. The formulation of the joint posterior is difficult. Even if one does not try for analytic solutions but uses MCMC methods, there are many issues to resolve around numerical performance, such as convergence of the MCMC to the correct underlying posterior and determination of tuning parameters, etc.

At this point, we see some interesting paths for further investigation. One is using KOH’s formulation expressed in (3) but calculating the parameters by Maximum Likelihood Estimation (MLE) methods instead of Bayesian updating. Dennis Cox has pursued this approach [Cox et al.] and it removes the difficulties associated with posterior generation (such as via MCMC). Another is to look at the first term in equation (3), the model emulation term, and replace it with another type of surrogate, perhaps a lower

fidelity or reduced order model. This has the advantage of “simplifying” the estimation in that one is solely focused on calculating the parameters for the GP model, but it may introduce limitations in terms of the capability to predict.

We wish to emphasize the difference between calibration and validation. Calibration of a computational model is adjusting a set of model parameters associated so that we maximize the model agreement with a set of experimental data (or, in certain cases, a set of numerical benchmarks). Validation of a computational model is quantifying our belief in the predictive capability of a computational model through comparison with a set of experimental data. Uncertainty in both the data and the model is critical and must be mathematically understood to do both calibration and validation correctly.

CUU is therefore a progression of thought that leads to an overlap of the concepts of calibration and validation. For example, the formalism discussed above of incorporating model uncertainty in Bayesian calibration procedures through the model discrepancy term  $\delta(\mathbf{x})$  in equation (3) is directly relevant to validation. In validation, we seek to quantify the discrepancy term by comparisons with experiments. From the validation perspective, it is natural to expect that  $\delta(\mathbf{x})$  is a random process of some type [Trucano et al., 2001]. The Gaussian process characterization of the model discrepancy discussed above seems to us to be useful in this context as well as in the CUU task. The task of validation should provide information that helps define specific parameterizations of the model discrepancy and should facilitate the process of calibrating this term.

We believe that predictability of a computational model centers on a specification of intrinsic limitations of the model as well as on our ability to predict model accuracy for specific applications (for example, as formalized in equation (2) above). Directly attacking the problem of characterization of the model discrepancy formalized above in  $\delta(\mathbf{x})$  really strikes at the need to quantify model uncertainty in a foundational way that is to some extent independent of the calibration problem of attempting to reduce this uncertainty. A rigorous validation process should achieve the goal of characterizing  $\delta(\mathbf{x})$ . CUU provides the proper formalism, at least in principle, for using this characterization to improve model accuracy. Thus, we see CUU as an important formalism for linking calibration and validation for quantitative improvement of the predictive content of computational models. Our future work on CUU will elaborate this view more systematically.

## References

- Campbell, K. (2002), “Exploring Bayesian Model Calibration: A Guide to Intuition.” Los Alamos Technical Report LA-UR-02-7175.
- Cox, D.D., J-S. Park, and C. E. Singer (2001), “A statistical method for tuning a computer code to a data base.” *Computational Statistics and Data Analysis* 37, pp. 77-92.
- Cressie, N. A. C. (1993), *Statistics for Spatial Data*, Wiley, New York.

Probabilistic Mechanics Conference, Albuquerque, New Mexico, July, 2004  
SAND2004-2317C

Gamerman, D. (1997), *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, Chapman and Hall/CRC, Boca Raton.

Gelman, A. J. B. Carlin, H. S. Stern and D. B. Rubin (1995), *Bayesian Data Analysis*, Chapman and Hall/CRC, Boca Raton.

Gilks, W.R., S. Richardson, and D.J. Spiegelhalter (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC, Boca Raton.

Kennedy, M. C. and A. O'Hagan (2001), "Bayesian Calibration of Computer Models." *Journal of the Royal Statistical Society*, 63, pp. 425-464.

Press, S. J. (2003), *Subjective and Objective Bayesian Statistics: Principles, Methods and Applications*, Wiley, New York.

Neal, Radford (1997). *Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification*. University of Toronto Technical Report No. 9702, Dept. of Statistics.

Neal, Radford. Flexible Bayesian Software documentation:  
<http://www.cs.toronto.edu/~radford/fbm.software.html>

Nabney, Ian. Netlab software. Documentation and software at:  
[www.ncrg.aston.ac.uk/netlab/](http://www.ncrg.aston.ac.uk/netlab/)

Trucano, T. G., R. G. Easterling, K. J. Dowding, T. L. Paez, A. Urbina, V. J. Romero, B. M. Rutherford, and R. G. Hills (2001), "Description of the Sandia Validation Metrics Project," Sandia National Laboratories, SAND2001-1339, Albuquerque, NM.

Williams, Chris (2002). "Gaussian Processes" chapter in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, ed. Cambridge, MA: MIT Press.