Uncertainty Quantification and Data Assimilation (UQ/DA) Study on a VERA Core Simulator Component for CRUD Analysis

Hany S. Abdel-Khalik
North Carolina State University

Reviewed by J. Michael Doster (NCSU) Rod Schmidt (SNL)

August 30, 2013
Executive Summary

This milestone focused on the deployment of recent advances in uncertainty and inference algorithms into the VERA environment to allow access and utilization by CASL domain experts. Over the past 3 years, VUQ focus area researchers have addressed a number of challenges that currently face existing uncertainty and inference techniques when applied to complex engineering problems such as nuclear reactors. Some of these challenges include the “curse of dimensionality” in the uncertain parameter space, the nonlinear and multi-physics nature of the associated physics models, and the high computational cost of constructing surrogate models. These challenges have been addressed using a hybrid approach that combines both stochastic and deterministic methods for uncertainty and inference analyses in an effective manner that seeks to overcome their limitations and combine their advantages.

The DAKOTA suite is selected for deployment of these developments and for demonstrating their applications to the CTF code package (formerly COBRA-TF), which is currently operating as part of the VERA environment. The QUESO capability was selected to complete the inference analysis. The CTF model is called by DAKOTA/QUESO to provide data calibration capability on the input parameters for CTF. Verification tests are employed to ensure that the noted developments are correctly implemented.

This report describes the theoretical and numerical work contributing to this milestone; in particular we overview the use of a Bayesian model calibration methodology followed by a description of the CTF model framework. We then provide a general framework for verifying model calibration results for CASL codes. We subsequently illustrate the construction of a surrogate model and implementation of the calibration framework for the CTF model. Finally, user and programmer’s instructions are given. The appendix contains a sample DAKOTA input file, and Git repository locations for the supporting material.
Contents

1 Purpose and Objectives 3

2 Bayesian Model Calibration 5
   2.1 Direct Implementation of Bayes’ Relation 6
   2.2 Metropolis Methods 6
   2.3 Burn-in or Chain Convergence 7
   2.4 Generation of Synthetic Calibration Data 8
   2.5 Deterministic Methods and DAKOTA Implementation 8

3 CTF Model Characteristics 8
   3.1 Surrogate Model 10
   3.2 Practical Implementation 10

4 Verification Methodology and Tests 12
   4.1 Verification Methodology 12
   4.2 Algebraic Test 13
   4.3 Physical Verification Test 15

5 CTF: Surrogate Model and Interface to QUESO 19
   5.1 Surrogate CTF Model Construction 19
      5.1.1 One Parameter Case 19
      5.1.2 Two Parameter Case 20
   5.2 Calibration of parameters with CTF 22
      5.2.1 One Parameter Case 22
      5.2.2 Two Parameter Case 23

6 Summary 24

7 Future Work 25
   7.1 Software Delivery 27
   7.2 Documentation 27
   7.3 Methodology Enhancement 27

Acknowledgements 28

Appendix A Sample DAKOTA Input File 28
1 Purpose and Objectives

In support of CASL’s mission to provide credible measures of uncertainties and devise ways for their reduction, this milestone has focused on deploying recent advances in data assimilation methods and calibration techniques (collectively referred to as inference analysis) produced by VUQ focus area researchers over the past few years. This deployment will demonstrate the use of VUQ tools to the wider CASL community in order to realize the benefits of VUQ in addressing CASL’s challenge problems. As part of this demonstration, the CTF thermal-hydraulics code will be used to illustrate the use of inference analysis, which will be completed using the QUESO library, currently interfaced with the DAKOTA software package [1], and all available under the VERA environment.

The high level purpose of this document is to provide a documented example of what is considered “best practices” by the VUQ team for the use of the newly developed tools in DAKOTA. A significant investment was made to provide explicit details and examples of the correct use of these tools in DAKOTA. Although the application of CASL simulation codes were limited to a small number of parameters with CTF, great care was taken to demonstrate the correct processes for using these advanced VUQ tools and also to verify that these processes produce the correct results. Part of what we are emphasizing in this document (and the follow on document “CASL Based DAKOTA User Guide and Best Practices”) is that it is easy to get bad results with high end VUQ tools and care needs to be taken to use them properly.

A broad objective of the CASL VUQ mission is to use models, simulation codes, and experiments to predict system responses and Figures of Merit for reactor designs with quantified and reduced uncertainties. Two fundamental steps are required to achieve this objective.

- Model Calibration: This involves the assimilation of data to quantify and update uncertainties associated with parameters, initial or boundary conditions, or forcing functions – which are collectively referred to as input uncertainties. We employ Bayesian model calibration techniques, which are based on the assumption that inputs are represented by random variables having associated probability density functions (PDFs). The goal in Bayesian inference is to construct and update these densities using measured data for the process of interest. This process is detailed in Section 2.

- Model Prediction: In this step, one computes the response, or quantity of interest, along with statistics, error bounds, prediction intervals, or a probability density function for the quantity of interest. This requires efficient propagation of input uncertainties determined either experimentally or via model calibration.

Model calibration and uncertainty propagation typically requires numerous forward or adjoint model evaluations which is computationally infeasible for CASL neutron transport, thermal-hydraulic, and chemistry codes. This necessitates the construction of surrogate models which incorporate the fundamental physics embodied in the high-fidelity simulation codes but are sufficiently efficient to permit numerous – e.g., up to millions – of evaluations for varying input and independent variable values.

Within the last two years, there has been significant emphasis on the incorporation of highly efficient algorithms in DAKOTA to facilitate the propagation of uncertainties.
These include spectral Galerkin and collocation methods which are commonly referred to as polynomial chaos expansion (PCE) techniques as well as Gaussian process (GP) surrogates.

This milestone focuses on the implementation and integration of advanced model calibration techniques into DAKOTA for use in the VERA environment. A key component of this is the integration and verification of general techniques to construct surrogate models for CASL codes. The following objectives are targeted.

The state-of-the-art in inference analysis techniques employs the engineering model in the forward (and less commonly the adjoint) mode to construct a surrogate model. A surrogate model is an approximation to the original model that can be executed more efficiently. The surrogate model is evaluated in place of the full model many times with different parameter perturbations. In an uncertainty analysis, the objective is to propagate parameter uncertainties to estimate response uncertainties. The results of the surrogate model execution are employed to construct a probability density function for the responses of interest. In the case of inference analysis, one has prior uncertainties on the parameters, which are then updated based on available measurements for some responses. The results of surrogate model execution can be processed using Bayes’ formula to generate a posterior probability distribution for the parameters. When parameter settings can be found so that the experimental data and computational model are consistent, and such settings are consistent with the prior distribution, the posterior distribution will be more informative on the model parameters than the prior distribution.

To complete uncertainty and inference analyses for realistic reactor models, it is necessary to construct a surrogate in an efficient manner. In practice, this tends to be difficult since the execution of the original model is typically computationally intensive. Moreover, the parameter uncertainty space is often of a high dimensionality, rendering the construction of the surrogate even more complex. Finally, for multi-physics nonlinear models, a brute force approach to constructing a surrogate model by treating the whole model as a black box is often computationally intractable and methods must be devised to transfer information effectively between the different physics models.

The research conducted over the past few years has focused on developing methods to address these challenges. In particular, sophisticated computational techniques, including reduced order modeling, hybrid sensitivity analysis techniques, and subspace techniques were shown to limit the search of the uncertainty space to a small subspace in order to render a computationally practical exploration of the uncertainty space. The subspace is identified using rigorous randomized techniques which allow one to identify patterns in large complex data sets. These techniques are used to characterize the dominant sources of uncertainties using quantifiable error metrics to ensure none of the important uncertainties are ignored. Rigorous mathematical analysis has been provided in support of these developments. Performing this reduction is essential given the typically large size of the uncertainty space, which must account for all the physical phenomena affecting system behavior, such as thermal-hydraulics feedback, isotopic depletion and their associated uncertainties. These methods were developed first for a single physics model, and have been later extended to multi-physics models. More details may be found in [2].
2 Bayesian Model Calibration

In Bayesian analysis, model parameters or inputs are represented as random variables having associated probability density functions (PDF). The goal in Bayesian inference is to construct these PDF using measurements or observations of the process of interest.

We let
\[ y = f(\chi, q) \]  
(1)
denote the output of a high-fidelity CASL code such as DENOVO or HYDRA where \( q \in \mathbb{R}^p \) denotes the inputs or parameters and \( \chi \) denotes values of independent variables such as discrete space or time values. The associated statistical model is
\[ Y_i = f(\chi_i, Q) + \varepsilon_i, \quad i = 1, \cdots, n \]  
(2)
where \( Y_i \) and \( \varepsilon_i \) are random variables representing measurements and measurement errors and \( Q \) is the random vector associated with inputs. Realized or measured values of the process are given by
\[ y_i = f(\chi_i, q) + \epsilon_i. \]  
(3)
Here errors are assumed to be independent and identically distributed (iid) and normally distributed with mean 0 and variance \( \sigma^2 \); that is \( \varepsilon_i \sim N(0, \sigma^2) \).

The densities associated with the inputs \( q \), for an observed data vector \( y = [y_1, \cdots, y_n] \), are specified by Bayes’ formula
\[ \pi(q|y) = \frac{\pi(y|q)\pi_0(q)}{\int_{\mathbb{R}^p} \pi(y|q)\pi_0(q) dq}. \]  
(4)
Here the random vector \( Q \) of input parameters is assumed to have a (possibly non-informative) probability density function, \( \pi_0(q) \), typically referred to as the prior or a priori density\(^1\), which encodes present state of knowledge about the parameters. The observations \( y \) are either experimental measurements of the process or values generated by the higher-fidelity model. The use of the latter to construct synthetic data is detailed in Section 2.4. Based on the assumption of independent and identically distributed errors from a normal distribution, the likelihood of observing \( y \) given a parameter value \( q \), for a model with a single response, is
\[ \pi(y|q) = \frac{1}{\sqrt{(2\pi\sigma^2)^n}} e^{-SS_q/2\sigma^2} \]  
(5)
where the sum of squares error is
\[ SS_q = \sum_{i=1}^{n} [y_i - f(\chi_i, q)]^2. \]  
(6)
Hence the likelihood provides information that updates the prior information in \( \pi_0(q) \) to provide a posterior density \( \pi(q|y) \) that more accurately quantifies the uncertainty associated with \( q \).

\(^1\)For brevity, we often use the convention of referring to probability density functions simply as densities since there is no confusion with physical densities.
2.1 Direct Implementation of Bayes’ Relation

In principle, the posterior densities associated with inputs can be directly computed by solving (4). In practice, this can be achieved only for small to moderate input dimensions \( p \) due to the difficulty associated with approximating the normalizing integral. Recently, sparse grid quadrature techniques have extended the range of \( p \) for which the integral can be reasonably approximated. However, for \( p \) greater than approximately 30, it is still necessary to employ alternative techniques such as those detailed in Section 2.2. In the verification examples, we illustrate the use of this direct solution to verify sampling-based techniques for \( p = 1 \) and 2.

2.2 Metropolis Methods

To avoid the difficulty associated with approximating the denominator for large input dimensions \( p \), one typically employs extensions of Metropolis algorithms initially developed in the 1950’s. The objective is to sample from a related distribution in a manner that converges to the unknown posterior density.

One shortcoming of typical Markov chain Monte Carlo (MCMC) sampling schemes is that they often rely on rejection based approaches to generate new samples from the posterior density. When searching a complex probability space the chain often fails to adequately explore the posterior, with a large proportion of the proposed samples being rejected. Each proposed sample corresponds to an evaluation of the model\(^2\) to determine the corresponding likelihood, thus rejecting a large number of the sample points is wasteful as rejected values are simply discarded. Further, if the rejection ratio becomes too high, the chain can become stuck, unable to proceed because it cannot find any valid samples to draw.

These difficulties have been addressed by recently developed Delayed-Rejection Adaptive Metropolis (DRAM) and DiffeRential Evolution Adaptive Metropolis (DREAM) algorithms.

Delayed Rejection Adaptive Metropolis (DRAM) Algorithms

The DRAM method addresses the issue of sample rejection by combining two essentially distinct but related methods:

1. Delayed Rejection (DR)

   When a sample is rejected by the Metropolis sampler, instead of being immediately discarded, a second stage proposal sample is generated with an acceptance probability that is specially calculated to guarantee convergence to the posterior density. This second stage proposal is allowed to be dependent on the previous rejected sample, yielding partial local adaptation of the proposal distribution at each step of the sampling chain and potentially allowing the second stage to more reliably generate a valid sample point. This refinement process can be continued for an arbitrary number of iterations, though one or two steps is typical. These refinements are local in nature and are discarded after each step. This method is detailed in [4].

\(^2\)In practice, a sample would correspond to an evaluation of the surrogate model and not necessarily the full model.
2. Adaptive Metropolis (AM)

This method relies on global adaptation of the proposal covariance based on previously accepted samples in the chain. At specified intervals the proposal covariance is updated to reflect information gleaned from the previous samples drawn by the chain. This process of adaptation will often improve the mixing of the chain so that it covers the target distribution more efficiently for any given number of iterations. This adaptation procedure implies that the resulting chain is neither reversible nor does it obey the Markov property; however in [5] the authors demonstrate that under mild assumptions the proper stationary distribution is preserved. Unlike DR, the adaptations are carried forward through subsequent steps of the chain.

Both of these methods can potentially produce large improvements in performance for the MCMC sampling procedure. For more details on both of these methods, see [4]. Parallel implementation of DRAM algorithms is detailed in [9].

DiffeRential Evolution Adaptive Metropolis (DREAM) Algorithms

Whereas DRAM algorithms have proven highly robust and successful for a number of large-scale applications, including climate models, there are regimes for which these algorithms are inefficient. This includes problems in which posterior densities are multi-modal, highly complex, or have heavy tails. For such regimes, DiffeRential Evolution Adaptive Metropolis algorithms can prove advantageous. These algorithms employ multiple chains with a differential evolution algorithm used to specify switching. Hence DREAM algorithms are inherently parallel which facilitates their implementation for computationally intense problems. Details regarding these algorithms are provided in [11–13]. The implementation of a DREAM algorithm into DAKOTA constituted one component of the L3 milestone VUQ.VVDA.P7.03 Model Reduction and Data Assimilation: Transport Codes. The initial integration has been completed and the development of a verification test suite constitutes part of a future milestone; see Section 7.

2.3 Burn-in or Chain Convergence

When discussing an MCMC-based method, it is important to address the issue of whether or not samples generated by the chain are being drawn from the true stationary distribution. This issue is colloquially referred to as burn-in, and is generally addressed by discarding some proportion of samples generated from the beginning of the chain. This helps to avoid the case where the chain has not yet properly stabilized to the stationary distribution. For our purposes, we choose burn-in periods of 10 to 50%.

Judging whether or not the chains have converged to the stationary distribution is difficult to assess in a theoretically rigorous manner. We attempt to avoid this issue by using a very large burn-in period, which adds little to the overall computational time due to the use of a cheap surrogate model. Some attempts at assessing MCMC chains using quantitative measures are described in [3].
2.4 Generation of Synthetic Calibration Data

When one is calibrating a model \( f(\chi, q) \) using experimental data, the variables \( y_i \) and \( \epsilon_i \) in (3) respectively denote measurements and measurement errors. This represents the final goal for CASL codes. As an intermediate step, however, one often calibrates the code using synthetic or model generated data. This is advantageous when developing or implementing algorithms or for regimes for which experimental data is unavailable. For example, we employ synthetic data for the COBRA example detailed in Section 5.

To construct synthetic data, one fixes a nominal parameter value \( \tilde{q} \) and constructs a solution \( f(\chi_i, \tilde{q}) \) at one or more spatial or temporal values \( \chi_i \). One then generates \( n \) values for the simulated measurement errors \( \epsilon_i \) by sampling from a normal distribution with mean 0 and standard deviation \( \sigma = 0.1 \times f(\chi_i, q)/3 \) (for example). This yields \( n \) simulated data values \( y_i \) given by (3) where the distribution of \( \epsilon_i \) is known.

2.5 Deterministic Methods and DAKOTA Implementation

MCMC is a stochastic method since it relies on the use of random sampling. Other deterministic methods have been devised for inference analysis and examples include nonlinear least-squares and linear adjoint methods. These methods are usually less computationally demanding than MCMC. They broadly treat the task of data calibration as an optimization problem, seeking to find a set of parameters that minimizes the residual between the predicted and observed values. Deterministic methods also typically provide rigorous guarantees of numerical convergence, whereas with MCMC methods the issue of assessing convergence can often prove difficult. On the other hand, these methods are typically less flexible than MCMC methods and give less detailed information about the posterior distributions. Generally, deterministic methods only provide an estimate for the mean and perhaps a covariance matrix, whereas MCMC approaches provide a detailed picture of the posterior density over the range of the problem. For a general discussion of non-Bayesian inference methods, we refer the reader to [10].

For implementation of these techniques, we employ DAKOTA (Design Analysis Kit for Optimization and Terascale Applications), which is selected because it is currently integrated into the VERA code package. DAKOTA provides a general framework for analysis of computer simulations. To complete the inference analysis capability, the QUESO library will be employed. QUESO provides an implementation of DRAM as a library that can be embedded in other software or used on its own. Recent work has provided an early implementation of an interface between DAKOTA and QUESO, which will serve as the vehicle for the tests in this milestone.

3 CTF Model Characteristics

COBRA-TF (Coolant Boiling in Rod Arrays - Two Fluids) is a reactor thermohydraulics code designed for Light Water Reactor vessel analysis [6, 7]. It is based on the two-fluid, three-field modeling methodology and contains both subchannel and 3D Cartesian forms of nine conservation equations for reactor modeling. It has been selected as part of the VERA code package and has been selected as the physics model for this study.
As a basis for our tests we choose the 17x17 PWR example for CTF obtained from the casl-dev Git repository. This test problem is the basis of the CTF contribution to the VERA-CS progression problem 6. The initial run time for the steady state problem was too large. We then chose a shorter running problem for this study. Thus the case was modified to a full transient case, with the response taken after a time interval of 0.3 seconds. Note that this problem will be studied further next year as part of our milestone on a complete VUQ study of VERA-CS applied to progression problem 6. Table 1 summarizes the parameters for the model.

The CTF parameters selected for our study were the inlet mass flow rate (MFR) and inlet enthalpy. These parameters are both available in the CTF preprocessor input control.inp, with perturbations generated by DAKOTA. The model response is taken to be the outlet power of the coolant at the end of the transient time interval; it is well-behaved and accessible in the file heat_balance.out.

An energy balance of the following quantities may be described as

\[ P_{outlet} = P_{inlet} + P_{core} + P_{transient} \]

where \( P_{outlet} \) and \( P_{inlet} = mh_{in} \) are the outlet and inlet coolant power. Here \( m \) denotes the inlet mass flow rate and \( h_{in} \) is the inlet enthalpy. Furthermore, \( P_{transient} \) quantifies the stored energy during transient; e.g., energy transferred to or from the surrounding structure causing a temperature rise or drop.

In the calibration experiment, the computed outlet coolant power is assumed discrepant from the measured value. The discrepancy will be used by the QUESO calibration modules to update the knowledge about the inlet conditions which are assumed to be uncertain.

Units for the input and output parameters are given in Table 2. Note that all parameter and response variations will be described in relative terms, with a relative value of 1.0 describing the nominal (reference) absolute value, and all other values as relative perturbations thereof.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of rods</td>
<td>264</td>
<td>N/A</td>
</tr>
<tr>
<td>Size of array</td>
<td>17</td>
<td>N/A</td>
</tr>
<tr>
<td>Number of guide tubes / water rods</td>
<td>25</td>
<td>N/A</td>
</tr>
<tr>
<td>Active length</td>
<td>3548.5</td>
<td>mm</td>
</tr>
<tr>
<td>Bundle pitch</td>
<td>215.0</td>
<td>mm</td>
</tr>
<tr>
<td>Fuel pellet diameter</td>
<td>8.1915</td>
<td>mm</td>
</tr>
<tr>
<td>Pin pitch</td>
<td>12.5984</td>
<td>mm</td>
</tr>
<tr>
<td>Initial MFR</td>
<td>85.979</td>
<td>kg/s</td>
</tr>
<tr>
<td>Initial rod temperature</td>
<td>292.7</td>
<td>C</td>
</tr>
<tr>
<td>Reference pressure</td>
<td>155.13</td>
<td>bar</td>
</tr>
<tr>
<td>Reference enthalpy</td>
<td>1281.97</td>
<td>kJ/kg</td>
</tr>
<tr>
<td>Total power</td>
<td>18.47507</td>
<td>MW(th)</td>
</tr>
</tbody>
</table>

Table 1: Parameter values for CTF model.
### 3.1 Surrogate Model

The calibration process can require thousands to millions of model evaluations to reasonably characterize the posterior PDF. Due to the computational expense of running CTF, it is more practical to use a surrogate model. Currently, the Bayesian calibration method in DAKOTA employs automated construction of Gaussian Process (GP) and Polynomial Chaos Expansion (PCE) surrogates, with the possibility to use other surrogate types in a non-automated fashion. In addition to the GP surrogate, for several cases we will also rely on a linear regression surrogate, which is also provided by DAKOTA.

Table 3 shows the number of samples used to construct the surrogate for each case. We choose to use fewer samples to construct the linear regression surrogate simply because we only use linear surrogates in cases where the model is highly linear.

A major advantage of relying on a surrogate model is the parallelism inherent in the construction process. For the GP surrogate construction, the model is sampled uniformly over the input space and the response computed at each sample point. These samples are collected into a vector of training points and then the GP surrogate is constructed based off of the training data. The sample evaluations all occur independently, implying that the sampling can take place in parallel, with parallelism up to the number of samples requested. This is advantageous since the implementation of DRAM in DAKOTA has only limited parallelism, while the version of CTF selected for this milestone is primarily serial in nature. A parallelized variant of CTF has been recently developed by CASL, but it was not used in this milestone. We achieve parallelism by taking advantage of the independence in model evaluations inherent to the problem.

Once constructed, the GP surrogate can be evaluated many times for negligible cost. Using a GP surrogate constructed in parallel with the DRAM calibration process produces an enormous speedup; while running the calibration process directly on CTF could potentially take months on a single workstation, instead the same computation only takes on the order of a few days. Also, once the sampling to construct the GP surrogate has been performed, the resulting surrogate can be reused across multiple experiments in DAKOTA. This allows us to change the parameters for the various cases without having to go through the expensive process of resampling CTF, which requires several orders of magnitude more computational time than the subsequent calculations. Only when changing the scale of the input parameters (e.g. 10% perturbation versus 1%) is it necessary to resample the model to ensure a good sampling density over the region of interest.

### 3.2 Practical Implementation

Here we briefly discuss the practical details of coupling DAKOTA with CTF. DAKOTA requires an interface with the target code that handles passing inputs to the code, invocation

---

3A parallelized variant of CTF has been recently developed by CASL, but it was not used in this milestone. We achieve parallelism by taking advantage of the independence in model evaluations inherent to the problem.
and processing of outputs. For this case, we use a mixture of Python code, shell scripts, and the DAKOTA preprocessor to manage execution of CTF during the sampling process. A directory contains template inputs for the CTF preprocessor that have a special control sequence entered in place of the values that DAKOTA is perturbing. DAKOTA copies this template directory to a numbered working directory, which allows multiple invocations of CTF to run concurrently without overwriting the output files. DAKOTA then generates a parameter vector and invokes a shell script, passing to this script the name of a file that contains the DAKOTA preprocessor input corresponding to the parameter values. The script then calls the DAKOTA preprocessor to generate the actual CTF preprocessor inputs and then calls the CTF preprocessor to generate the actual CTF input deck. Then, the CTF executable is called and runs to completion. After CTF returns control to the shell script, the script then calls a simple Python script that extracts the outputs via regular expression, which is then returned back to DAKOTA. The Python script also handles the possibility of a failed CTF run, in which case it returns IEEE NaN.

This process seems somewhat complex; however, in practice it is reliable and reasonably flexible. It is capable of handling the introduction of perturbations across multiple input files (though we do not use this ability in the cases shown in this report) and can easily be adapted to perturb the CTF input deck directly instead of using the preprocessor inputs. Use of the DAKOTA preprocessor also allows algebraic transformation of variables in the input file, permitting even greater flexibility in building the input decks. As an example, we use simple algebraic transformations to automatically convert between relative perturbations from DAKOTA and their absolute values in the CTF input files. This allows DAKOTA to deal with simpler relative perturbations, but to easily reconstruct the actual values that are needed by CTF.

Three distinct steps are used for the test cases described. The first is to generate the training set for the surrogate model. For this, DAKOTA is used to generate samples uniformly in the input parameter space and the output samples are saved for use during GP surrogate construction. DAKOTA uses the interface described above to evaluate CTF directly and determine the true model value at each point and outputs the input-output data to a text file in a tabular format.

The second step is to generate the synthetic calibration data for DRAM. An error distribution is specified in the DAKOTA input deck (typically an uncorrelated, multivariate normal distribution), which then generates samples of the error process. These sampled errors are then added to the nominal model calculation(s) to obtain the simulated data value(s). Since the surrogate model appears to be highly accurate for the three cases considered, instead of calling CTF, we choose to use the GP surrogate constructed from the training data built during the previous step in calculating the simulated data values. It would be simple to instead invoke CTF directly, but tests showed that the surrogate was
accurate enough that there was no detectable difference using the surrogate versus direct evaluation of CTF. This process results in the synthetic calibration data used to illustrate Bayesian model calibration in the following sections.

The final step is to actually run the calibration process. In this step, the DAKOTA input is set to construct a GP surrogate model from the training points generated in the first step and no new evaluations of CTF are used. It is also possible to have DAKOTA construct the GP surrogate on-the-fly before the MCMC chain is initialized, but since we already had the training data available this would waste a large amount of time. After construction of the surrogate, DAKOTA then runs DRAM until the requested number of samples are drawn from the posterior distribution, with extra samples beyond the requested number generated at the beginning to ensure that the chain has stabilized to the stationary distribution (usually termed *burn-in*; see Section 2.3).

After termination, the accepted samples from the posterior densities are available in the directory `OutputData`, under the file `file_cal_ip_raw.m`. This is a MATLAB script that, when run, populates variables in the MATLAB workspace corresponding to the posterior samples. We typically choose to further post-process this file with a simple script that converts it to an HDF5 formatted data file for use with software other than MATLAB. Note that if the DAKOTA output level is set to `VERBOSE` in the input file the calibration process will also produce a file called `QuesoOutput.txt` in the top-level directory. This file contains a listing of all sample points, including rejected points, as well as the computed likelihood for each sample. This file is organized into columns: first inputs, then responses, and finally the likelihood, and can be useful for troubleshooting the calibration process.

As part of this milestone, all of the files relevant to the cases described in this report will be committed to the CASL Git repository. This includes documentation on each case, sample results, and helper utilities. Also, a representative input file for the DAKOTA-QUESO interface is included in the appendix of this report.

## 4 Verification Methodology and Tests

### 4.1 Verification Methodology

We summarize here a general framework for verifying model calibration results for CASL codes implemented in VERA via DAKOTA. This framework is generally applicable to codes with nonlinear parameter dependencies and experimental or synthetic data. The verification methodology for the DAKOTA-QUESO DRAM package has the following components.

(i) Test algorithms using a linearly-parameterized model where analytic uncertainty relations can be computed. Whereas this is not generally possible for CASL codes with nonlinear parameter dependencies, it provides a first step for verifying the capabilities of the model calibration framework and it may be used in certain nearly linear operating regimes.

(ii) Compare to direct implementation of Bayes’ formula (4) for small to moderate input or parameter dimensions \( p \); e.g., \( p \leq 20 \) to 30. For the likelihood relation (5) and a
noninformative prior \( \pi_0(q) \), this involves the numerical approximation of

\[
\pi(q|y) = \frac{e^{-SS_q/2\sigma^2}}{\int_{\mathbb{R}^p} e^{-SS_\zeta/2\sigma^2} d\zeta} = \frac{1}{\int_{\mathbb{R}^p} e^{-(SS_\zeta-SS_q)/2\sigma^2} d\zeta}
\]  

(7)

where the second formulation avoids numerical evaluation.

(iii) Compare to other packages that implement DRAM; e.g., MATLAB.

(iv) Compare the densities with those provided by the alternative Metropolis algorithm DREAM discussed in Section 2.2. The initial implementation of DREAM into DAKOTA constituted one component of the L3 milestone *VU.VVDA.P7.03 Model Reduction and Data Assimilation: Transport Codes.*

(v) Compare to sampling distributions provided by frequentist analysis. Whereas this approach can guide verification, it must be used with care since the underlying assumptions for frequentist and Bayesian inference differ significantly; see Chapter 4 of [8]. For example, asymptotic analysis yields Gaussian sampling distributions which will obviously be inaccurate if the true distribution is highly non-Gaussian.

(vi) Check the convergence of the algorithms by increasing the number of quadrature points used in (7) or number of iterations in DRAM or DREAM chains.

The verification criterion (i) is illustrated in Section 4.2 for a linear algebraic model where one can analytically specify the parameter distribution. This provides an initial test to verify the basic functionality of the QUESO inference library. Verification criteria (ii)–(vi) are illustrated in Section 4.3 for a steady-state heat equation with measured data. This provides a prototype for heat transfer processes analogous to those embodied in the thermal-hydraulic code CTF.

4.2 Algebraic Test

As a first verification test of the DAKOTA-QUESO model calibration capability, we employ the linear algebraic model

\[
Y = b \cdot Q + \varepsilon
\]

where \( Q \) is an uncertain parameter and \( \varepsilon \) is an independent and identically distributed error term that is normally distributed with mean \( \mu = 0 \) and standard deviation \( \sigma \). The goal is

---

4All uncertainty and inference techniques employed by DAKOTA/QUESO require the ability to sample a given probability space. In particular, generation of training data for the surrogate model and generation of the synthetic calibration data (see Section 2.4) require the sampling of a probability space with known density. For the sampling, instead of pure random sampling we have elected to use Latin Hypercube Sampling (LHS) which is available through DAKOTA. This sampling method is well-established in the statistical literature as a more reliable approach for spreading samples over the entire probability space. This is done by subdividing the space into equally probable regions and sampling these regions in a way that enforces marginal stratification of each input, promoting space-filling coverage of the input space. LHS is known to be no worse than pure random sampling and typically performs significantly better for a negligible computational cost.
to use measurements $y$ of $Y$ to inform on the distribution of $Q$. To achieve this, we generate synthetic data $y$ in such a way as to allow the exact posterior distribution of $Q$ to be known analytically. In our implementation, the QUESO suite was used to sample $n$ data values $y_i$ from a normal distribution having mean $b \cdot \bar{q}$ for $b = 0.045213$ and $\bar{q} = 1$, and standard deviation $\sigma = b/2$. This results in a normal posterior density of $Q$ having mean $\mu_\pi = \bar{y}_n/b$ and standard deviation $\sigma_\pi = 0.5/\sqrt{n}$, where $\bar{y}_n$ is the sample mean of the $n$ data values $y_i$. The sampled $y$ values are employed as input to the inference capability (QUESO), with the prior distribution for $Q$ assumed non-informative. We will compare the posterior density obtained this way to the analytically calculated PDF determined using Bayes' formula, which is considered to be the exact result.

For this test, four calibration data sets of sizes $n = 1, 10, 100, 1000$ were generated with QUESO using the process described in the previous paragraph. For this example, 150,000 QUESO posterior samples were generated. After the burn-in discussed in Section 2.3, there were 100050 remaining and each tenth value was subsampled and used for inference.

Figure 1 shows the posterior density from QUESO using 1, 10, 100, and 1000 observation(s) of the response. As expected, the estimated posterior distribution converges towards an atom at the posterior mean $\mu_\pi$.

Table 4 shows analytical and estimated values of the posterior standard deviation $\sigma_\pi$,.

![Figure 1: Posterior density using (a) 1, (b) 10, (c) 100 and (d) 1000 data values.](a) (b) (c) (d)
where the estimated value is the sample standard deviation of the 10050 DAKOTA-QUESO posterior samples. Here we see the effect of diminishing returns — while 10 data values provides an improvement in $\sigma_\pi$ of more than $3\times$ over a single measurement, increasing from 500 to 1000 data values provides a much smaller reduction in $\sigma_\pi$. We note the estimates of $\sigma_\pi$ are expected to approximately follow the Monte Carlo convergence rate of $\mathcal{O}(1/\sqrt{n})$, which is confirmed by the results of Table 4.

4.3 Physical Verification Test

The second verification example is a steady-state heat model for an uninsulated aluminum rod in open air with a heat source at one end and dissipation due to conduction and air-cooling along the length of the rod. This provides a prototype illustrating certain heat transfer properties that are similar to the phenomena modeled by CDT that is simple enough to have an analytic state solution.

As detailed in [8], the steady-state model is

$$
\frac{d^2 T_s}{dx^2} = \frac{2(a+b)h}{abk} \left[T_s(x) - T_{amb}\right]
$$

(8)

Here $\Phi$ is the source flux at $x = 0$, $h$ is a convective heat transfer coefficient, $k = 2.37 \frac{W}{cm \cdot ^\circ C}$ is the thermal conductivity for aluminum, and $T_s$ is the steady-state temperature at positions $x$ along a rod with cross-sectional dimensions $a = b = 0.95$ cm and length $L = 70$ cm. Hence $ab$ is the cross-sectional area and $a + b$ results from the surface area of an infinitesimal unit. The parameters are taken to be $q = [h, \Phi]$. The forcing term on the right-hand side of (8) and boundary condition at $x = L$ result from Newton’s law of cooling with ambient air temperature $T_{amb}$. The boundary condition at $x = 0$ quantifies heat input by the source.

Data consists of temperature measurements at 15 equally spaced spatial locations $x_i = x_0 + (i-1)\Delta x$, $i = 1, \cdots, 15$, where $x_0 = 10$ cm and $\Delta x = 4$ cm. Steady state temperature data for a rectangular, uninsulated aluminum rod is compiled in Table 5.

<table>
<thead>
<tr>
<th>$x$ (cm)</th>
<th>10</th>
<th>14</th>
<th>18</th>
<th>22</th>
<th>26</th>
<th>30</th>
<th>34</th>
<th>38</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp ($^\circ C$)</td>
<td>96.14</td>
<td>80.12</td>
<td>67.66</td>
<td>57.96</td>
<td>50.90</td>
<td>44.84</td>
<td>39.75</td>
<td>36.16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x$ (cm)</th>
<th>42</th>
<th>46</th>
<th>50</th>
<th>54</th>
<th>58</th>
<th>62</th>
<th>66</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp ($^\circ C$)</td>
<td>33.31</td>
<td>31.15</td>
<td>29.28</td>
<td>27.88</td>
<td>27.18</td>
<td>26.40</td>
<td>25.86</td>
</tr>
</tbody>
</table>

Table 5: Steady state temperatures measured at locations $x$ for an aluminum rod.
It is illustrated in Chapter 7 of [8] that ordinary least squares techniques yield the optimal parameter estimates $\Phi = -18.41$ and $h = 0.00191$. The error variance estimate is $\sigma^2 = 0.0627$ and the covariance matrix, computed using analytic sensitivity relations, is

$$V = \begin{bmatrix} 2.1034 \times 10^{-2} & -2.0286 \times 10^{-6} \\ -2.0286 \times 10^{-6} & 2.0972 \times 10^{-10} \end{bmatrix}.$$ 

(9)

The standard deviations for the errors and sampling distribution are $\sigma = 0.2504$, $\sigma_\Phi = 0.1450$, $\sigma_h = 1.4482 \times 10^{-5}$.

(10)

The model fit to this data using the optimal parameter estimates is illustrated in Figure 2(a) and the residuals are shown in Figure 2(b). The latter is checked to verify the validity of the hypothesis that errors $\varepsilon_i$ in the statistical model (2) are independent and identically distributed (iid). The residuals in Figure 2(b) do not exhibit a discernable pattern thus indicating the validity of this assumption.

Due to the highly nonlinear dependence of the solution $T(x,q)$ on the parameters $q$, there is no analytic relation for specifying input uncertainties in terms of measurement uncertainties. Furthermore, the latter must be inferred from the data since they are not directly measured. Hence this example provides a prototype for illustrating the verification criteria (ii)–(vi) in Section 4.1. We compare the QUESO implementation in DAKOTA with MATLAB DRAM and DREAM packages. As detailed in Section 7, the verification of the DREAM algorithm in DAKOTA using this, and other problems in the proposed verification suite, constitutes a future proposed milestone.

**Verification Results**

For the DAKOTA-QUESO implementation, we employed a direct Python interface.\(^5\) To ensure burn-in, or convergence to the posterior, as detailed in Section 2.3, we employ

\(^5\)The direct Python interface allows DAKOTA to pass parameter values directly to the model in memory, bypassing the need to write the values to disk. The implementation of the model runs nearly instantaneously, so writing to disk represents a significant overhead that should be avoided if possible. Using the direct interface provides a speedup of approximately 30x for the overall MCMC sampling process. The Python interface is not enabled by default in DAKOTA and must be explicitly enabled when compiling.
chains of length 30,000 which we verify are converged and fully mixing. For the QUESO implementation, we employed the value $\sigma = 0.2604$ for the measurement error which was obtained using the ordinary least squares analysis. It is shown in Chapter 8 of [8] that this is consistent with the value $\sigma = 0.2504$ obtained using MATLAB DRAM to sample the error distribution. The extension of the DAKOTA package to sample a density for $\sigma$ will be addressed in a future milestone; see Section 7.

We summarize here four verification tests for the algorithms and codes. These are based on the verification criteria (ii)-(vi) in Section 4.1.

1. We illustrate in Figure 3 the marginal densities for $h$ and $\Phi$ obtained through direct solution of Bayes’ relation (4), DAKOTA-QUESO, and MATLAB DRAM and DREAM. It is observed that all four densities agree thus providing a first verification test.

2. The contour plot in Figure 4 illustrates the correlation between $h$ and $\Phi$ and compares the ordinary least squares optimum, marked with a black dot, with the mean of the joint distribution, marked with a blue circle. Comparison with the contour plots for the direct Bayes’ solution and sample plots for MATLAB DRAM and DREAM, plotted in Figure 5, provides a second verification test.

3. A third verification test is provided by Table 6 where the standard deviations obtained with the four techniques are compared with the ordinarily least squares (OLS) estimate for the sampling distribution. The close agreement of first four with the OLS estimate is due to the fact that the estimated densities shown in Figure 3 are nearly Gaussian.

4. To illustrate burn-in properties, we plot in Figure 6 the MATLAB DRAM densities for $h$ and $\Phi$ obtained with differing chain lengths. When combined with statistical tests and visual observation of the chains, this provides a final verification test.

Whereas not all of these tests will be performed with CASL codes, they illustrate typical tests that should be conducted to verify the accuracy of calibrations obtained using Bayesian inference.

Figure 3: Marginal densities for (a) $\Phi$ and (b) $h$ generated through direct solution of Bayes’ relation (4), DAKOTA-QUESO, MATLAB DRAM and MATLAB DREAM.
Figure 4: Joint density for $h$ and $\Phi$ constructed using DAKOTA-QUESO.

Figure 5: Contours for the direct method and joint samples from MATLAB (a) DRAM and (b) DREAM.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\mu_\Phi$</th>
<th>$\sigma_\Phi$</th>
<th>$\mu_h$</th>
<th>$\sigma_h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>-18.417</td>
<td>0.146</td>
<td>1.915 $\times 10^{-5}$</td>
<td>1.461 $\times 10^{-5}$</td>
</tr>
<tr>
<td>QUESO</td>
<td>-18.417</td>
<td>0.139</td>
<td>1.915 $\times 10^{-5}$</td>
<td>1.400 $\times 10^{-5}$</td>
</tr>
<tr>
<td>MATLAB DRAM</td>
<td>-18.417</td>
<td>0.152</td>
<td>1.915 $\times 10^{-5}$</td>
<td>1.524 $\times 10^{-5}$</td>
</tr>
<tr>
<td>MATLAB DREAM</td>
<td>-18.417</td>
<td>0.143</td>
<td>1.915 $\times 10^{-5}$</td>
<td>1.438 $\times 10^{-5}$</td>
</tr>
<tr>
<td>OLS</td>
<td>-18.417</td>
<td>0.145</td>
<td>1.9101 $\times 10^{-5}$</td>
<td>1.448 $\times 10^{-5}$</td>
</tr>
</tbody>
</table>

Table 6: Mean and standard deviation for the parameters $\Phi$ and $h$. 
5 CTF: Surrogate Model and Interface to QUESO

Here we discuss the construction of a surrogate model for CTF and the use of the Bayesian inference techniques of Section 2 to quantify parameter uncertainty using simulated data.

5.1 Surrogate CTF Model Construction

Two types of surrogate models are employed in the later cases: a linear regression model and a GP model. We show some simple tests that verify the accuracy of these surrogate models.

5.1.1 One Parameter Case

The first parameter that we consider is the inlet mass flow rate (MFR). Figure 7 depicts the process of surrogate model construction using DAKOTA and validates the surrogate model

Figure 6: Densities obtained using MATLAB DRAM with differing chain lengths.

Figure 7: Surrogate versus true model evaluations for one parameter case
predictions against the true model predictions determined by executing CTF. The red points in figure 7 show 100 samples drawn from a uniform grid in the parameter space and evaluated with the CTF model. 50 more samples are then selected at random to be used as training data for the construction of the GP surrogate. Finally, we draw 500 more samples from a uniform density and propagate them through the GP surrogate, with the black line showing an interpolation of these values\(^6\). It can be seen that the surrogate interpolates the reference data with a high degree of accuracy. Further, it is apparent that the one parameter case is highly linear, with a correlation coefficient of \(\rho = 0.9999\). Thus, for subsequent variations of the one parameter case we will assume that a linear surrogate can also be used.

5.1.2 Two Parameter Case

Figure 8 shows a simple linearity study for the two parameter case comprised of the Inlet MFR and Inlet Enthalpy. For visualization, the blue surface interpolates the model response at 200 points sampled at random in the input space, with the actual sample points shown

\(^6\)Note here that we do not reuse any samples at each step. The reference data (red points) are constructed on a uniform grid, while the other two datasets are drawn independently and at random from a uniform distribution.
as the smaller points on the surface. Each line of colored points shows the model response sampled along a random vector in the input space. For this case, it is apparent that the response surface is highly linear. Table 7 gives the linear correlation coefficients for this case, which give a quantitative assessment of the degree of linearity in the surface.

Figure 9 shows a direct comparison between the response computed by CTF and the response computed by the GP surrogate. The blue surface in 9 is constructed based on 600 points sampled from a uniform mesh, with response computed directly via CTF. The black points show the response of a GP surrogate model constructed from 200 training points, with the surrogate model then evaluated at 1000 points randomly sampled from the same input distribution as the direct model evaluations. It is clear from Figure 9 that the surrogate is in good agreement with the true response values as computed by CTF.

<table>
<thead>
<tr>
<th></th>
<th>Inlet MFR</th>
<th>Inlet Enthalpy</th>
<th>Outlet Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inlet MFR</td>
<td>1.0</td>
<td>0.0</td>
<td>0.88</td>
</tr>
<tr>
<td>Inlet Enthalpy</td>
<td>0.0</td>
<td>1.0</td>
<td>0.48</td>
</tr>
<tr>
<td>Outlet Power</td>
<td>0.88</td>
<td>0.48</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 7: Correlation coefficients for two parameter case.
5.2 Calibration of parameters with CTF

In this section, we show the results from calibrating the CTF model by assuming the inlet conditions to be uncertain with a range of variability of ±10%, which is representative of inlet reactor conditions uncertainties. For the single parameter case, the inlet mass flow rate is assumed uncertain, and the outlet coolant power is assumed measured. The basic assumption here is that the discrepancy between calculated and measured coolant power originates from inlet conditions uncertainties. Two different surrogates were employed, the GP surrogate model constructed by DAKOTA, and a linear regression model based on the linearity illustrated in Figure 7.

5.2.1 One Parameter Case

We illustrate first the use of QUESO to estimate a density for the Inlet MFR using synthetic data generated in the manner described in Section 2.4. To construct a surrogate model, we employed 51 samples drawn uniformly from within 10% of the nominal parameter value reported in Table 2. We then ran CTF to \( T = 0.4 \) seconds using the nominal parameter value to obtain the outlet coolant value \( y_{ss} = 126.831.7 \). We constructed synthetic data by sampling 400 values \( \epsilon_i \) from a normal distribution with mean 0 and standard deviation \( \sigma = \frac{0.1}{3} y_{ss} = 4229.5 \) and adding these to \( y_{ss} \).

To construct a posterior density using QUESO, we ran a chain of length 150,000 and discarded 49,950 during burn-in. Each tenth entry in the burn-in chain was used for inference. Due to the low parameter dimensionality, we also constructed the ‘true’ posterior density by approximating Bayes’ formula in the manner illustrated in (7). Finally, we can exploit the observation that the outlet response exhibits a nearly linear dependence on the Inlet MFR at the computed time, as illustrated in Figure 7, to compute an analytic normal relation for the parameter density in the manner detailed in Section 4.2.

The three sets of normalized results are compared in Figure 10 where it is observed that the QUESO posterior very closely matches the direct Bayes and analytic solutions. This

![Figure 10: Densities for the Inlet MFR obtained using QUESO DRAM, direct Bayes solution, and the analytic solution.](image-url)
verifies the accuracy of the QUESO implementation. To provide a second verification test, we note that the standard deviation $\sigma = 1.9093 \times 10^{-3}$ obtained using QUESO is within 1.24% of the analytic value $1.8859 \times 10^{-3}$.

5.2.2 Two Parameter Case

Secondly, we consider the performance of the QUESO DRAM package for estimating densities associated with the two inputs Inlet MFR and Inlet Enthalpy. In the notation of Section 2, these inputs are denoted by $q = [q_1, q_2]$.

We first used CTF to compute training data over the time interval $[0.2, 0.35]$ which we used to construct a surrogate model that was quadratic in time and approximately linear with respect to the parameters; e.g., see Figure 9. Using nominal parameter values, this surrogate was then used to compute nominal outlet coolant responses $y_i$ at the four times $t_i = 0.2, 0.25, 0.3, 0.35$. Fifteen noise values $\epsilon_i$, drawn from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = \frac{0.0001}{3}y_i$, were added to the responses to generate 60 synthetic data values.

The prior distribution for $q$ was taken to be a zero mean normal distribution with covariance matrix

$$ V = \begin{bmatrix} (0.1/3)^2 & 0 \\ 0 & (0.1/3)^2 \end{bmatrix}. $$

Because the response at the specified temporal values is approximately linear with respect to the parameters, we can analytically specify the posterior distribution for $q$ based on this assumed prior distribution.

We employed QUESO DRAM in the manner detailed for the one-parameter case to construct marginal posterior densities for the two parameters as well as a joint density for $q$. We compare the marginal QUESO posteriors with the analytic densities in Figure 11. We first note that QUESO is accurately estimating both posterior densities. Secondly, the posterior standard deviation $\sigma_{post} = 0.01$ for the Inlet MFR density has been reduced from the prior

![Figure 11: QUESO DRAM and analytic marginal posterior densities for the (a) Inlet MFR with prior density and (b) Inlet Enthalpy.](image)
value $\sigma_{prior} = 0.1/3 \approx 0.033$ as illustrated in Figure 11(a). Finally, we note that value 0.1696 produced by QUESO for the correlation between the two parameters closely matches the analytic correlation value 0.1730. This is corroborated by the comparison between the contours for the analytic joint density and the sampled chain elements as illustrated in Figure 12.

In combination, this verifies the accuracy of the QUESO DRAM package for this application and illustrates the capability of Bayesian inference to reduce input uncertainties as measurements are acquired.

6 Summary

In this document we have presented the basic theory behind the DREAM and DRAM methods that we employ for Bayesian model calibration. We describe the approach employed in statistical simulation studies to generate synthetic data for use in studies verifying the performance of these methods. This is the beginning of the procedure that we will develop for using high fidelity codes like HYDRA to calibrate lower fidelity codes like CTF. This high fidelity calibration is intended to result in the CASL version of COBRA (CTF) producing a lower uncertainty in predictions of interest than other versions of COBRA, enhancing its value to industry.

We then discussed the process for the construction of a surrogate that accurately predicts code output but runs much faster than the code itself. This example uses CTF on a 17x17 grid (note this CTF input deck is the basis for CTF’s contribution to VERA-CS progression problem 6) to construct a surrogate. Because this surrogate runs much faster than CTF, a more detailed study of the uncertainty in CTF output is possible without the cost of thousands of CTF runs. This surrogate construction will play a key role in VUQ of other high CPU cost VERA components on CASL progression and challenge problems.

We then proceed to present verification results for all of our examples. It is important to note that Bayesian methods are very powerful tools, but if applied improperly they are
likely to generate misleading or incorrect results. This explains the large emphasis we have placed on verification in this document, with the hope that practitioners will be empowered to deploy the procedures outlined herein to conduct verification studies centered on their own applications of Bayesian model calibration.

We first show that our calibration methods correctly solve a simple algebraic inverse problem with analytical solutions. Here we demonstrate how adding calibration data reduces the uncertainty in a single model parameter of interest. This demonstrates an approach we propose to explore that would utilize HYDRA numerical experiments to reduce the uncertainty in a larger set of CTF model parameters than that considered in this document.

We then show the calibration methods on a nonlinear heat conduction problem using DREAM and DRAM and verify that the MCMC-based calibrations are correct. Here we first compare the MCMC-based results to a direct solution of Bayes’ theorem. We follow this by comparing DRAM results obtained by DAKOTA to DREAM and DRAM results obtained by readily available MATLAB software and to the traditional least squares method and show agreement.

Finally we demonstrate surrogate model construction with a 17x17 pin CTF model. We employ this surrogate to quantify parameter uncertainty. This will be the main approach deployed for uncertainty quantification of HPC applications in CASL that possess formidable run times. The surrogate allows for fast uncertainty quantification that would not otherwise be possible with current computer resources available to CASL if relying solely on direct code calculations. We then demonstrate with synthetic data (similar to numerical experiments with HYDRA) how inverse problems involving CTF model parameters would be solved through Bayesian calibration, with the ultimate goal of reducing uncertainty in CTF predictions of interest by leveraging the model parameter uncertainty reduction attained as a result of this calibration.

7 Future Work

The VUQ focus area must address the need for DAKOTA to have direct access to all relevant model parameters in CTF and also the need for automated extraction of relevant figures of merit from CTF. The model parameters for which this access is required will be determined through a hierarchical sensitivity analysis. First, assume in the short term that the droplet field will not play an important role. One then can focus on six equations (three for liquid and three for vapor) and the mass, momentum, and energy exchange between phases and with the fuel rod clad. These assumptions (which can be relaxed in the future if necessary) lead to six meta-parameters:

1. liquid friction with the wall (momentum)
2. liquid heat transfer with the wall (energy)
3. liquid friction with vapor (momentum)
4. liquid mass transfer with vapor, boiling at the wall, boiling in the bulk, and condensation (mass and energy)
5. vapor friction with the wall (momentum)

6. vapor heat transfer with the wall (energy)

These six meta-parameters will be exposed first. To do this, a multiplier must be applied to the physics of the meta-parameter in the code. When the multiplier is one, the code answer should only change by round off (based on a possible order of operation change caused by an optimization in the compiler). After these six multipliers are added, the CTF regression tests must be rerun to verify that the answers obtained are the same (to round off) as those calculated with the unmodified version of CTF. To complete phase one, a separate DAKOTA readable and writeable VUQ input file for these six meta-parameters must be created. At this stage we will have enabled DAKOTA to perform basic VUQ studies with CTF under VERA for all progression and challenge problems.

In follow-on work, based on time and funding, we will then construct a tree from each of these meta-parameters. We select the tree to expand by identifying the meta-parameter having the largest sensitivity of these six meta-parameters with respect to induced figure of merit variation in progression problems (5,6,7) or challenge problems (CIPS, CILC, RIA, PCI, etc.). Each of these meta-parameters is comprised of multiple correlations with multiple parameters in each and a set of decision points which selects the correlation to use (for example, use laminar or turbulent friction factors based on the Reynolds number being less than or greater than 2300). Depending on the depth of the tree, how many branches and decision points, we can either add new meta-parameters to our VUQ input file or directly expose the parameters from the correlations (add the parameters to the VUQ input file), whichever is easier and makes more sense from a technical standpoint.

This approach is intended to provide VUQ with early access to the most essential model parameters so that comprehensive and meaningful uncertainty quantification studies may be conducted. It is important to note that for many of our early progression and challenge problems, there will not be many parameters of interest since we will be operating at or near an operational point. The large number of parameters in CTF will naturally get “pruned” by the progression and challenge problems. For example, a large amount of CASL work by CTF will be turbulent single phase flow. This involves a relatively small subset of parameters.

The output data from CTF is in a format that is primarily intended to be human readable. This presents a challenge when running CTF in an automated fashion, where DAKOTA must read a figure of merit. To this end, CTF must be modified to output a predefined set of figures of merit. With this capability, DAKOTA can read the figure of merit, adjust the parameters, and rerun CTF in an automated fashion. The figures of merit will be selected from a PIRT analysis of the important progression and challenge problems.

In the remainder of this section, we discuss improvements that will be made to DAKOTA in the future based on knowledge gained from studies done as part of this work. We believe that these changes will improve the usability of DAKOTA for CASL applications. It should be noted that many of the capabilities employed in this document are emerging experimental capabilities in DAKOTA. Similar to other CASL software underdevelopment, there are gaps in the documentation that should be improved. The VUQ team will put a major effort next year into improving the documentation and providing best practices, user guidelines, and detailed worked examples that clearly demonstrate the proper use of these new DAKOTA tools for VUQ.
7.1 Software Delivery

Work needs to be done to absorb the new uncertainty quantification capabilities into the DAKOTA build process. We need to decide which platforms and compiles will be supported by CASL for the use of DAKOTA and provide executables for these systems. When possible, directions will be provided for advanced users to install the software.

7.2 Documentation

Documentation will be improved in the short term, but by the end of next fiscal year we will have a Best Practices and User Guideline Document. This document will include worked examples of how rigorous VUQ should be done for CASL applications.

7.3 Methodology Enhancement

The following capabilities will be added to DAKOTA during the next fiscal year:

1. The covariance matrix employed in the DAKOTA-QUESO DRAM proposal (jumping) distribution is presently constructed automatically based on bounds provided by the user, with no mechanism to enter a predetermined covariance matrix. A use case will be provided that allows user specification of the proposal covariance matrix.

2. DAKOTA use cases will be provided that allow user specification of a measurement (observational error) covariance matrix. Specifically, we will address the setting in which \( n \) observations are available, and the error process is assumed mean-zero Gaussian with measurement covariance matrix \( \Sigma \). The following cases will be covered:

   - \( \Sigma = \text{diag}\{\sigma_i^2, i = 1, \ldots, n\} \), where \( \sigma_i^2 \) are fixed and input by the user for \( i = 1, \ldots, n \) (DAKOTA checks if \( \sigma_i^2 > 0 \) for \( i = 1, \ldots, n \)).
   - \( \Sigma \) fixed and input by the user (DAKOTA checks if \( \Sigma \) as specified is a legitimate covariance matrix).
   - \( \Sigma = \sigma^2 R_n \), where \( \sigma^2 \) is input by the user as random or fixed, and \( R_n \) is fixed and input by the user (DAKOTA checks if \( \sigma^2 R_n \) is a legitimate covariance matrix).
   - \( \Sigma = \text{diag}\{\sigma_i^2 R_{n_i}, i = 1, \ldots, m\} \), where the \( \sigma_i^2 \) are input by the user as random or fixed, and the \( R_{n_i} \) are fixed and input by the user for \( i = 1, \ldots, m \) and \( n = \sum_{i=1}^{m} n_i \) (DAKOTA checks if the \( \sigma_i^2 R_{n_i} \) are legitimate covariance matrices).

The last use case above represents the situation in which multiple replicate data sets on a single response are available from independent laboratories. The default use case will be \( \Sigma = \sigma^2 I_n \), which is the case of replicate data on a single response from a single laboratory.

3. DAKOTA will be extended to provide a use case allowing specification of noninformative prior distributions.
4. DAKOTA will be extended to allow for user provided log-likelihood and log-prior sub-
routines with a mechanism to identify random parameters vs. fixed quantities. Al-
though most users will be satisfied with the previously identified, more specific options,
a more extensive capability to deal with multivariate or functional output data, and
hierarchical models, will be needed. This work may extend beyond the next fiscal year.

Acknowledgements

We would like to specially thank Robert Salko, Brian Adams, Laura Swiler, Cameron
Thames, Paul Turinsky, Rod Schmidt and Michael Doster, each of whom contributed valu-
able advice and insight to this milestone. Their expert knowledge has contributed substan-
tially to both the content and exposition of this report. This research was supported by
the Consortium for Advanced Simulation of Light Water Reactors (http://www.casl.gov),
an Energy Innovation Hub (http://www.energy.gov/hubs) for Modeling and Simulation of
Nuclear Reactors under U.S. Department of Energy Contract No. DE-AC05-00OR22725.
North Carolina State University and Los Alamos National Laboratory (LANL) are core
CASL partners.

Appendix A  Sample DAKOTA input file

Figure 13 shows a prototypical input file for running calibration with CTF and DAKOTA-
QUESO. The first block, strategy, defines the type of problem to be solved by DAKOTA;
it will generally be set to single_method.

The method section lays out the configuration for the algorithm to employ, in this case
we select the one corresponding to Bayesian calibration with QUESO. We also specify that
we wish to construct a GP surrogate model from 200 real model evaluations. The
Lines 9 and
and 10 enable the DRAM capability. Lines 13 and 14 give scaling factors; here they are set to
unit values.

The variables block declares the input parameters. These are required to be of type
continuous_design, which implies that they will have a non-informative prior distribution
with lower and upper bounds given on lines 18 and 19. The descriptors definition is an
optional specification that allows the user to give names to associate with the variables in
the output. Default names will be assigned if it is left empty.

In the responses block, the response of interest for the calibration process is given. This
also points to the data file containing the calibration data and describes how it is laid out.
The no_gradients and no_hessians options tell DAKOTA not to calculate gradient or
hessian values. Other options exist (e.g., numerical_gradients), but they are not used by
the QUESO interface.

In the final interface block, we specify how DAKOTA should interface with CTF. This
interface will only be called to initially construct the GP surrogate. fork indicates that
DAKOTA should spawn invocations as forked processes, with the asynchronous setting

More complex surrogate model specifications can be given in a separate block, but this input illustrates
the simplest case.
strategy
  single_method
method
  bayes_calibration queso
    emulator
      gaussian_process
      surfpack
      emulator_samples = 200
    rejection delayed
    metropolis adaptive
    samples = 200000
    output verbose
    likelihood_scale = 1
    proposal_covariance_scale = 1 1
variables
  continuous_design = 2
  descriptors = 'inlet_mf' 'inlet_enth'
  lower_bounds = -.1 -.1
  upper_bounds = .1 .1
responses
  calibration_terms = 1
  num_experiments = 100
    calibration_data_file = 'caldata.dat'
    freeform
      descriptors = 'outlet_power'
    no_gradients
    no_hessians
interface
  fork
    asynchronous evaluation_concurrency = 6
    analysis_driver = 'run_cobra.sh'
template_directory = 'template'
work_directory named 'workdir'
  directory_tag directory_save
  file_save
aprepro
parameters_file 'params.in'
results_file = 'results.txt'

Figure 13: Sample input file for DAKOTA-QUESO

indicating that up to 6 invocations can be run simultaneously. The remaining settings point
DAKOTA to the various scripts and settings necessary to use the interface described in
section 3.2. Note especially the `aprepro` setting, which tells DAKOTA it should format the parameter data as input files for the DAKOTA preprocessor.

More detailed inputs and supporting files corresponding to the cases in this report will be available on the `casl-dev` Git repository.

References


