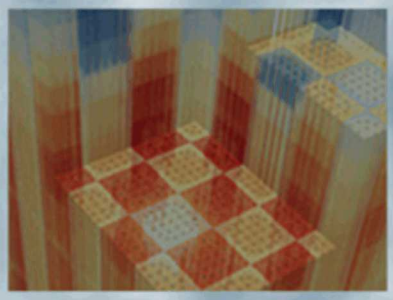
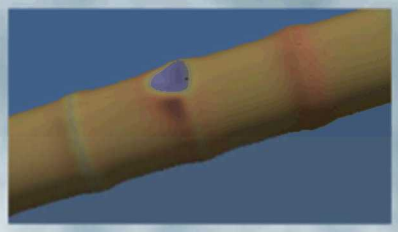
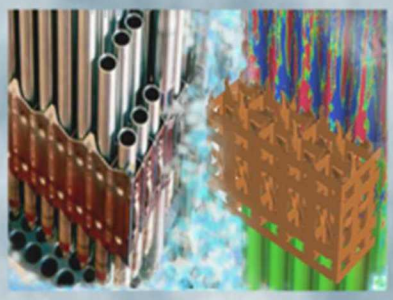
CASL A DOE Energy Innovation Hub

SAND2018-12962R

CASL-X-2018-XXXX







Calibration of MAMBA and CRUD Sources Using Full-Core CTF+MAMBA

L3:PHI.CRUD.P16.02

Lindsay Gilkey and Adam Hetzler, Sandia National Laboratories

Benjamin Collins and Robert Salko, Oak Ridge National Laboratory

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2018-XXXXXXXX

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

REVISION LOG

Revision	Date	Affected Pages	Revision Description
0		All	Initial Release

Document pages that are:

Export Controlled _____

IP/Proprietary/NDA Controlled_____

Sensitive Controlled_____

Requested Distribution:

To:

Copy:

EXECUTIVE SUMMARY

This report outlines a process for the deterministic calibration of MAMBA using the computational toolkit Dakota. The tools and processes for deterministic calibration have been built and are laid out in this report. While completing this milestone, issues emerged with MAMBA that resulted in delays. The consequences for these difficulties to the calibration process are briefly discussed. The report concludes with an outline of a path forward for Bayesian calibration. The Bayesian calibration will be performed next year. This process was laid out by Benjamin Collins, Robert Salko, and Adam Hetzler.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. MILESTONE DESCRIPTION

The purpose of this milestone is to implement deterministic calibration of MAMBA. The deterministic process can be used for calibration of MAMBA. The outlined process and tools are used with Dakota. Dakota is a computational toolkit developed at Sandia National Laboratories for calibration, optimization, and uncertainty quantification. It can be coupled to codes and post-processing to intelligently automate a calibration process. During the milestone process, Dakota was coupled to MAMBA and post processing scripts for data collection and calculations. Much of this framework can be reused for Bayesian calibration.

This milestone report outlines the tools used for the Dakota calibration, as well as the thought process that went into model selections and other important decisions for the deterministic process. The milestone report also outlines a tentative plan for Bayesian calibration of MAMBA. The Bayesian calibration plan is a work in progress, and it will likely change as unanticipated issues or delays are sure to arise during the process.

1.1 Working Group and Acknowledgements

This work was performed in collaboration with Benjamin Collins (ORNL) and Robert Salko (ORNL). Support and advice for the milestone was given by Adam Hetzler (SNL), Ralph Smith (NCSU), and Dusty Brooks (SNL). Support for Dakota was provided by Brian Adams (SNL) and Adam Stephens (SNL).

2. DETERMINISTIC CALIBRATION EXERCISE

The deterministic calibration exercise was performed by perturbating parameters in the surface kinetics and heat transport areas of MAMBA. The physics in MAMBA are coupled to MPACT and CTF. A representation of this coupling is shown in Figure 1. MAMBA models the clad surface chemistry, CTF models the thermal hydraulics, and MPACT models the neutron transport.

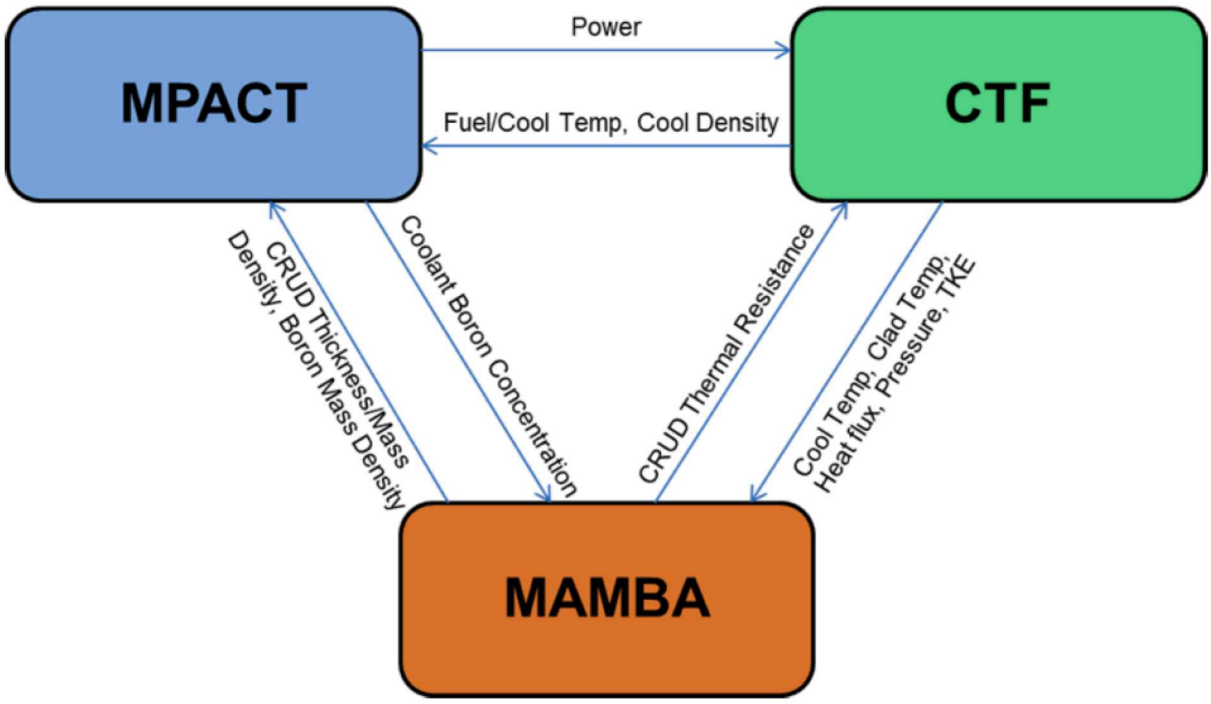


Figure 1: VERA and MAMBA coupling interfaces. Image is taken from [2].

All three codes and associated physics are needed to accurately predict the formation of crud. The way the code is implemented, a local higher power will cause a higher surface temperature, which will cause additional boiling, and more crud will be deposited on the surface. More crud will reduce the moderation around the pin and increase the precipitation of boron as lithium tetraborate. This will lead to the power being suppressed [2]. These highly coupled physics are needed to capture the formation of crud. The coupled nature of this problem makes running MAMBA non-trivial and can be a complicating factor for calibration.

As the codes and physics are coupled, calibration in MAMBA will lead to different outputs being pushed from MAMBA to IMPACT and CTF. Only one of the codes (MAMBA) will be calibrated due to time and resource constraints, however feedback and physics from all three codes will contribute to the calibrated solution. This can make calibration difficult for a few reasons. Noise can become a dominating factor during the calibration process as code output and physics are passed between codes. Additionally, it is possible that small changes in code output can be smeared numerically by the code coupling. Since some schemes in Dakota rely on small gradients (where noise can potentially dominate the response), the effects of this can be significant during calibration. This is also possible when calibrating a de-coupled code, however the coupled nature of the problem adds a layer of complexity that can make issues much more difficult to diagnose as problems can arise from any component in the coupled system.

2.1 MAMBA Background Information

This is a high-level overview of the MAMBA parameters that will be the focus of the calibration effort. The two components of MAMBA that will be calibrated are the surface kinetics and the heat transfer equations. For more complete documentation of MAMBA, refer to [2].

2.1.1 Surface Kinetics

In MAMBA, the growth of the crud layer on the surface of the cladding is caused by the deposition of nickel ferrite particulate [2]. The following equation defines the kinetic equation that drives the surface deposition:

$$\frac{dC_{NiFe_2O_4}}{dt} = \left(k^p_{s,nonboil} + \kappa^p_{s,boil} q''_{s,boil}\right) C^p_{NiFe_2O_4,cool} - \gamma_{s,e} k_{TKE}$$

The terms of the equation are as follows:

- $C_{NiFe_2O_4}$ (mol/cm^3): Concentration of nickel ferrite in the crud layer
- dt (s): Time step of equation
- $k^p_{s,nonboil}$ (mol/J cm): Nonboiling deposition of nickel ferrite
- $\kappa^p_{s,boil}$ (cm^2/J): Deposition coefficient of deposition caused by boiling
- $q''_{s,boil}$ (W/cm^2): Boiling heat flux on the surface of the crud layer
- $C^p_{NiFe_2O_4,cool}$ (mol/cm^3): Concentration of nickel ferrite in the coolant
- $\gamma_{s,e}$: Scaling factor applied to model erosion of crud layer
- k_{TKE} (J/kg): Predicted turbulent kinetic energy

The crud structure in MAMBA is assumed to grow at a user defined porosity. Nickel ferrite is assumed to be the only compound that can contribute to the growth of the crud layer in MAMBA. The crud structure will grow to a set porosity, por, and after the concentration of crud becomes sufficient to fill the node at the set porosity, the node is switched from being an external node to an internal node. The centration of the crud in the node has the following equation:

$$C_{NiFe_2O_4} \geq \frac{por\ \rho_{NiFe_4O_4}}{M_{NiFe_4O_4}}$$

$\rho_{NiFe_4O_4}$ is the theoretical density of the nickel ferrite, $M_{NiFe_2O_4}$ is the atomic mass of the nickel ferrite, and por is the specified porosity of the crud skeleton. The default value in MAMBA for por is 70%. After a boundary node is switched to an internal node, any additional concentration of nickel ferrite is moved to the newly activated boundary node. The initial values of the soluble nickel, iron, boron, lithium, and hydrogen gas are used as initial concentration for the internal kinetics in MAMBA.

2.1.2 Heat Transport

Heat transfer is important to the growth of the crud layer as the temperature distribution is an important quantity in determining the crud growth in MAMBA. The steady-state heat conduction equation is:

$$\vec{\nabla} \cdot k\vec{\nabla}T = q'''_{sink}$$

where k is the thermal conductivity as a function of local temperature and porosity. q'''_{sink} is the local heat sink caused by boiling. The heat sink is described by the following equation:

$$q'''_{sink} = \begin{cases} 2\pi r_{chim}\mu(\eta)h_{chim}\rho_{chim}(T - T_{sat}), & T > T_{sat} \\ 0, & T \leq T_{sat} \end{cases}$$

The terms of the equation are as follows:

- r_{chim} : Characteristic radius of a chimney
- h_{chim} : Boiling heat transfer coefficient of the chimney
- ρ_{chim} : Surface density of chimneys
- $\mu(\eta)$: Permeability of the crud as function of porosity

A more complete derivation of the above equation is in [2]. MAMBA uses a 1D radial heat conduction model. This is different from MAMBA3D, which solved a 3D heat conduction equation. The change was made to MAMBA to speed up run time during developmental and testing stages for the code. In the future, 3D heat conduction model may be added back into MAMBA.

2.2 Calibration Parameters

Four parameters were chosen by Ben Collins to be manipulated by the Dakota calibration exercise. These parameters are:

- knsb : $\kappa^{p}_{s,boil}$, deposition coefficient of deposition caused by boiling (Section 2.1.1)
- por : Initial porosity that the NiFe2O4 skeleton grows in at (Section 2.1.1)
- htc : h_{chim}, boiling heat transfer coefficient of the chimney (see Section 2.1.2)
- pmult : $C^{p}_{NiFe_2O_4,cool}$ (mol/cm^3), scaling term for coolant precipitation (see Section 2.1.2)

The ranges and the initial points for each of the calibration parameters is in Table 1.

Table 1: Calibration Parameters

Parameter	Initial Point	Lower Bound	Upper Bound
knsb	7.20E-4	1.00E-4	7.50E-4
por	0.700	0.600	0.800
htc	40.0	20.0	60.0
pmult	1.00	0.700	1.30

After revisiting the calibration parameters several months later, the porosity parameter may be eliminated from the calibration parameters. It is also possible that additional calibration parameters will be added in the future as the calibration process for MAMBA is still in development. Ideally, a sensitivity analysis would be performed to determine the most appropriate calibration parameters.

2.3 Quantity of Interest

The quantity of interest during the calibration process is the deposited boron mass.

In order to couple MAMBA with Dakota, it was necessary to reduce the amount of calibration data. In its raw form from MAMBA, there are hundreds of thousands of data points as the boron mass data has both temporal and spatial coordinates. It was necessary to pre-process the data before it was read by Dakota, as otherwise this would cause out-of-memory errors. It is possible to run Dakota in parallel, where this is not a limiting issue, however for this first calibration attempt, Dakota was run in serial which made a data pre-processing step necessary.

The reduced quantity of interest was decided to be the total assembly boron between each grid span in the upper regions of the core at the beginning, middle, and end of life. This was an initial calculation made based on best judgement for the application, and will likely change as time goes on and the MAMBA calibration process matures. The same states and summed spans were used for both cycle 7 and cycle 8. It is possible to use different spans and states for the two cycles, however the same number of calibration points was used for both to give both cycles equal weight in the calibration (see Section 3.3 for additional explanation).

The spans used for the boron calculations were spans [0,26,32,40,47,52]. The beginning state was defined as state 4, which corresponds to Watts Bar Unit 1 cycle 7, day 33.2 and cycle 8, day 20.18. The middle state was state 10, which is Watts Bar cycle 7, day 199.40 and cycle 8, day 152.08. The end state was state 16 which is Watts Bar cycle 7, day 361.30 and cycle 8, day 319.38. This results in a total of 30 calibration terms for the reduced data set. The 30 terms include two cycles and have spatial and temporal terms. All the pre-processed data is weighted equally in its current form. This initial preprocessing of the calibration data is summarized in Table 2.

Table 2: Pre-Processing Calibration Data[1]

Data Point	Cycle	State / Day	Summed Spans
1	7	4 / 33.2	[0,26)
2	7	4 / 33.2	[26,32)
3	7	4 / 33.2	[32,40)
4	7	4 / 33.2	[40,47)
5	7	4 / 33.2	[47,52)
6	7	10 / 199.40	[0,26)
7	7	10 / 199.40	[26,32)
8	7	10 / 199.40	[32,40)
9	7	10 / 199.40	[40,47)
10	7	10 / 199.40	[47,52)
11	7	16 / 361.30	[0,26)
12	7	16 / 361.30	[26,32)
13	7	16 / 361.30	[32,40)
14	7	16 / 361.30	[40,47)
15	7	16 / 361.30	[47,52)
16	8	4 / 20.18	[0,26)
17	8	4 / 20.18	[26,32)
18	8	4 / 20.18	[32,40)
19	8	4 / 20.18	[40,47)
20	8	4 / 20.18	[47,52)
21	8	10 / 152.08	[0,26)
22	8	10 / 152.08	[26,32)
23	8	10 / 152.08	[32,40)
24	8	10 / 152.08	[40,47)
25	8	10 / 152.08	[47,52)
26	8	16 / 319.38	[0,26)
27	8	16 / 319.38	[26,32)
28	8	16 / 319.38	[32,40)
29	8	16 / 319.38	[40,47)
30	8	16 / 319.38	[47,52)

[1] Note: Pre-processing of calibration data is subject to change. These are the initial quantities used for the calibration data.

3. DAKOTA SETUP FOR CALIBRATION

Dakota is a computational toolkit developed at Sandia National Laboratories for iterative calibration and uncertainty quantification studies. It uses algorithms to intelligently probe specified parameter spaces and decide on the next evaluations points. This is a major advantage over traditional expert-based calibration studies, even though it requires more total model evaluations. Expert-based calibration typically involves an experienced analyst manually perturbing parameters based on their prior knowledge of model behavior or the behavior of the actual system. The disadvantage is this type of calibration study is labor intensive and expert opinion based calibration or uncertainty quantification will vary depending on the opinion of the expert performing the study.

In a national laboratory setting (i.e. SNL or ORNL), labor is typically the major cost in a project, which can make these types of calibrations studies a very attractive option. Additionally, certain calibration methods, such as Bayesian calibration, would be in practice impossible to perform by hand due to the number of evaluations required to converge the results. Having Dakota drive the calibration process allows us to intelligently select model parameters and yield confidence intervals for the parameter ranges. Bayesian calibration also gives distributions on the parameters and the probability density functions that the optimal solution will lie in that range.

The setup for the Dakota calibration is described in this section. The Dakota calibration was set up for a deterministic calibration using the NL2SOL or NCSU_DIRECT methods. The coupled Dakota-MAMBA interface will select the calibration parameters from a user-defined range, run the required models, perform post-processing to extract model results and calculate residuals with the calibration data, and then return the residuals to Dakota. Dakota greatly simplifies the process by intelligently selecting the parameters for calibration. Some methods in Dakota can also return confidence intervals. It should be noted that the performance of Dakota is dependent on the user selecting an appropriate calibration method, calibration parameters and ranges, and calibration data.

3.1 Dakota Input

The Dakota input file is included in A.1 Dakota Input File. This section describes some of the details of the Dakota input file. Full details of all the options in the input file can be found in the Dakota reference manual ([3]).

3.1.1 Calibration Method

The calibration methods chosen for Dakota were both deterministic methods. These methods will be described in the following sections.

Deterministic calibration was chosen for a first approach since it is much simpler to implement than an alternative approach, Bayesian calibration. Deterministic calibration can be performed without needing to use a surrogate as the number of total evaluation iterations is much less than would be needed for Bayesian calibration and surrogate construction. Being able to perform deterministic calibration is an essential step along the way to being able to perform Bayesian calibration. Bayesian calibration is desirable as it yields distributions for the calibrated parameters, whereas deterministic calibration will only be able to yield the confidence interval of the calibrated parameters. Bayesian calibration is the eventual goal with this calibration exercise, however that first requires being able to complete the deterministic calibration step.

3.1.1.1 NL2SOL Method

NL2SOL was the first deterministic method attempted for calibration of MAMBA. NL2SOL is the non-linear least-squares solver method in Dakota. NL2SOL is a gradient-based, local calibration

method. NL2SOL returns N objective functions for N calibration terms. The objective functions returned from Dakota are the residual term per calibration term.

$$F_{i,objective} = T_i^{Model} - T_i^{Data}$$

In the case where no calibration data is provided to Dakota, Dakota assumes any data returned to the Dakota output files is in terms of residuals, as shown above.

NL2SOL can perform quick calibrations and optimizations provided that the method is able to identify large enough gradients in the code output returned to Dakota. NL2SOL is also able to return confidence intervals for the calibrated parameters. As the method is gradient-based, if no gradients are identified, the method will not be able to find a calibrated solution. This can pose problems if the gradient calculation is too fine (parameter selections have too small of a delta) or if the model output is insensitive to small changes in model inputs, which is why a sensitivity study should be performed prior to attempting calibration.

3.1.1.2 NCSU_DIRECT Method

If the NL2SOL method fails, it is possible to instead utilize the NCSU_DIRECT method. NCSU_DIRECT is the dividing rectangles method and is a global derivative-free optimization method. NCSU_DIRECT can be an effective calibration method if NL2SOL is not able to find calibrated parameters. NCSU_DIRECT utilizes a dividing rectangle algorithm that can explore the entire parameters space as well as local parameters spaces that have lower residual terms. Figure 2 shows a simplified diagram that illustrates how a dividing rectangles algorithm searches local parameters spaces and the global parameter space as it attempts to minimize the residuals.

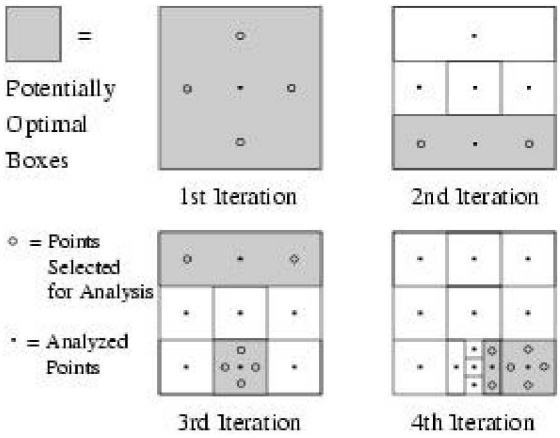


Figure 2: Dividing rectangles algorithm subdivides the parameter space to balance local and global search [3].

NCSU_DIRECT is also different from NL2SOL as the residuals returned to Dakota are treated differently. Dakota performs a least squares calculation on the residuals, which results in one objective function term per set of calibration data. This objective function is calculated for N calibration terms as:

$$F_{objective} = \sum_{i=1}^{i=N} \left(\frac{T_i^{Model} - T_i^{Data}}{\sqrt{var_i}} \right)^2$$

where “var” is the variance associated with individual calibration terms.

As NCSU_DIRECT is globally based and derivative-free, it is a much more expensive method than NL2SOL. It does not return confidence intervals for the calibrated parameters. The benefit of using NCSU_DIRECT is it is often able to find a calibrated solution in parameters spaces or for data sets that NL2SOL is not able to. NL2SOL will not be able to find a calibrated set of input parameters in cases where there is a lack in gradients in model outputs, whereas NCSU_DIRECT will almost always be able to find a solution in the same parameter space. It is up to the user to determine if the solution determined from NCSU_DIRECT is valid or if the calibration needs to be performed with a different method.

3.1.2 Model Failure

In cases where the chosen parameter space causes a model failure, this needs to be indicated to Dakota to prevent the algorithm from choosing that parameter space for future evaluations and to prevent Dakota from failing when it receives unexpected output. This can be done in many ways, however the method chosen for this calibration exercise was to return a very large residual (10.0e+12) to Dakota. This prevents Dakota from failing, as a numerical residual is returned, and it prevents Dakota from attempting to use parameters in the neighborhood of the failed parameter set.

There are alternative methods for returning a model failure to Dakota. One of the other methods is to capture failures by writing the word “FAIL” to the Dakota output. If this failure flag is used, it is possible to use a “continuation” flag in Dakota. The continuation flag will indicate to Dakota to attempt to approach the failed solution by halving the interval of the parameter step between the failed solution and the last successful model evaluation. Dakota will attempt to halve the interval ten evaluations. If the model does not successfully evaluate during this time, Dakota will abort and stop all evaluations. For this reason, we did not choose to use the “FAIL” flag to indicate model failure to Dakota. The alternative method of returning a large residual is effective and is a recommended method for failure capture in the Dakota manual.

It should be noted that the user should proceed with extreme caution when using model failure flags in Dakota. Model failures caused by inappropriate ranges for the calibration parameters should be addressed by adjusting the parameter ranges, therefore eliminating the model failure. Model failures caused by actual code errors need to be fixed and the model needs to be appropriately revalidated before any calibration can be attempted.

3.2 Dakota Driver

The interface from Dakota to MAMBA is performed by running a driver script. The driver script performs the following operations:

1. Performs keyword replacement with the Dakota Dprepro utility to replace placeholder values with selected parameter values in scripts for MAMBA
2. Runs all cases
3. After all cases have run, runs the post-processing script that performs calculations and returns residuals to Dakota.
4. If model failure is detected, large residuals (10.0e+12) are returned to Dakota.

The Dakota driver script is contained in Appendix A.2.

3.2.1 Dprepro

Dprepro is the **D**akota **pre-pro**cessor that is distributed with Dakota. It is a useful utility that allows keyword replacement of template files to create simulation files for Dakota coupled simulations. The typical usage of Dprepro within the Dakota driver is the following:

```
dprepro params.in input.template input
```

When Dprepro is used on a file, it looks for any input between the left and right delimiters in the input.template file. Once the delimiters are identified, Dprepro replaces keywords specified in the Dakota input file with the Dakota input parameters contained in params.in for the specific evaluation. Dprepro has several useful abilities, such as the ability to perform simple mathematical operations (such as would be needed for unit conversions), and the ability to write the Dakota inputs in different formats, such as writing the numerical values as integers or in scientific notation. It is also possible to list default values for unused Dakota parameters in Dprepro. This can be done by writing the parameter name and value between the Dprepro delimiters:

$$\{param = default\ value\}$$

The default Dprepro delimiters are '{' and '}'. If '{' or '}' are used in the modified input file (e.g. Python dictionary format), a different delimiter set can be specified with a slightly modified dprepro command:

```
dprepro --left-delimiter='{{{' --right-delimiter='}}}' params.in
input.template input
```

3.3 Dakota Output

Output is returned to Dakota in the form of residuals (instead of model responses). This was preferable for this exercise as it allowed us total control of the post-processing performed on the model outputs. Residuals are anticipated by Dakota if no calibration data set is specified to be read into Dakota. Returning residuals instead of model responses technically makes this an optimization problem, where Dakota attempts to minimize the residuals, however this is in practice the same as a calibration problem. The only difference is a post-processing script is calculating the residuals instead of an internal calculation in Dakota calculating the residuals.

Output from Dakota was specified as a field calibration term. This was specified in the Dakota input file (Appendix A.1). Field calibration terms are sets of related calibration terms with a certain length. The advantage of using field calibration terms in Dakota is that they can be assigned to certain coordinates (such as a temporal or spatial coordinates) and Dakota can perform linear interpolation between terms if the calibration data and the model data occur at different coordinates. In a case where there are multiple experiments used for calibration data, different coordinates can be specified for each set of experimental data. However, there is a limitation that Dakota expects that all simulation data to occur at the same coordinates for every evaluation.

In the case of the MAMBA calibration, that means that if cycle 7 and cycle 8 are specified as two different experiments, instead of a single group of calibration data, Dakota will anticipate that the same states and spans are used for the model calculations and residuals. This can be a complicating factor when setting up a calibration in Dakota.

Dakota anticipates a fixed number of residuals or model responses for each evaluation. This makes some sort of post-processing necessary if multiple cycles or "experiments" are being considered. In other words, if the Dakota input file specifies that thirty residuals will be returned every evaluation, all outputs to Dakota must contain thirty residuals. If all cycles / "experiments" are grouped together as a single set of calibration data (instead of treated separately by Dakota), including more calibration data from a single cycle will weigh the calibration so the cycle with more calibration data has higher precedence than the other cycle. To remedy this, a weighting function can be specified in the Dakota input file. This will weigh residuals in the following way:

$$f = \sum_{i=1}^{i=N} w_i \left(y_i^{Model} - y_i^{Data}\right)^2$$

If no weight function is used, it is assumed that all residuals have the same relative emphasis. No weight function was used for this calibration exercise. To account for this, the same states and spans were used for the cycle 7 and cycle 8 data, which resulted in the same amount of data and the same relative weight for each calibration data point. In the future, it may be desirable to use all data from the cycle 7 and cycle 8 data and instead place weights on the cycle data.

The post-processing script to calculate the residuals is contained in Appendix A.4. The post-processing script also returns the residuals to Dakota with the file "results.out." This is easier to implement than handling the results with the special results shell variable in the Dakota driver for the current Dakota setup, however this is case specific and may not be true for different Dakota methods or applications. The quantity of interest for the Dakota calibration exercise is described in Section 2.3.

3.4 Difficulties

Difficulties emerged early in the calibration process. MAMBA, in its then current state, was not able to predict the formation of crud due to errors that existed in the code. This resulted in delays as the MAMBA code needed to be fixed before proceeding with calibration.

Once the errors in MAMBA were resolved, it was decided to proceed with a hand calibration. The hand calibration was performed instead of carrying out the deterministic calibration to its end because it is easier to implement. Once the hand calibration is complete, it should give a better idea of the appropriate parameters and calibration method to use with MAMBA for when the actual deterministic and Bayesian calibrations are moved forward.

Another complicating factor that contributed to this decision is the long run time for MAMBA coupled to CTF and MPACT. Long runs times and multiple evaluations per calibration make deterministic calibration a lengthy process, and Bayesian calibration (which will involve building a surrogate) an even longer process. By hand selecting points, this avoids inefficiencies that will happen while the deterministic calibration attempts to probe the parameter space. Once we have the ranges for the calibration parameters better defined and a better idea of how the coupled code behaves to the user inputs, we will then be able to perform deterministic calibration, which will be able to yield confidence intervals for the parameters.

Additionally, there were some difficulties in data handling for cycle 7 and cycle 8. The large amount of data required pre-processing, since Dakota was being run in serial. Additional thought will likely need to be put into the data preprocessing. This was discussed in some length in Section 2.3. There are also some limitations in Dakota which deal with the model output. Dakota anticipates that the model data returned to Dakota has identical coordinates for different experiments. This means that the coordinates (spatial or temporal) for the cycle 7 data must be identical to the cycle 8 data. This poses a problem since the two cycles have a different numbers of states in the current implementation of the model. This difficulty was addressed during data postprocessing before output was returned to Dakota.

The previous subsections of Section 3 were written with the assumption that a deterministic calibration was performed with MAMBA, and deterministic calibration will be performed in the future.

3.5 Results

Due to difficulties outlined above, a successful deterministic or "hand-picked" calibration of MAMBA has not been completed at the time of writing of this report. The hand calibration is currently a work in progress, and deterministic calibration will follow. These steps will need to be completed before attempting the Bayesian calibration plan laid out in Section 4.

4. FUTURE WORK

While this milestone report covers the work that was completed to perform deterministic calibration of MAMBA, the ultimate goal is to perform Bayesian calibration, which accounts for uncertainties and yields confidence intervals and probability distributions. This future work is briefly outlined below and will be performed during FY19. Several milestones have been laid out for next year which involve the Bayesian calibration of MAMBA.

4.1 Bayesian Calibration Overview

Bayesian calibration is fundamentally different from other conventional calibration methods (such as deterministic calibration) [4]. Bayesian calibration is an application of Bayes Theorem, which is a mathematical formula for determining a conditional probability. During Bayesian calibration, the probability density functions (PDFs) of the parameters are updated iteratively as the calibration advances. After enough iterations have elapsed, the PDFs determined by the Bayesian calibration are the most likely to contain the optimal calibrated solution. The Bayesian process will yield confidence intervals and their distributions, which makes Bayesian calibration desirable. Bayesian calibration also reduces the tendency to overfit models to data as the calibration process is not attempting to minimize residuals, but instead ensure that model output will be statically consistent with the calibration data [4]. Uncertainties in the calibration data are taken into account during the Bayesian process.

Bayesian calibration also has disadvantages. A major disadvantage is the number of iterations needed to converge the PDFs. Since the PDFs are updated iteratively, this can require a very large number of evaluations. It is especially true if the initial PDFs are far from the solution reached by the Bayesian process, since it will take more iterations to converge. Also, if there is insufficient calibration data (in terms of quantity, quality, or relevance), that can render a Bayesian calibration essentially useless. This is also true for all other calibration methods.

The number of evaluations needed to complete Bayesian calibration can potentially be very large (order of 10^4 and larger), which can make Bayesian calibration a daunting undertaking. For codes with long run times (order of hours), performing this many evaluations is virtually impossible. For this reason, Bayesian calibration is often performed with a surrogate.

4.2 Surrogate Model Overview

The surrogate is a reduced model, built from model data to perform in the parameter space being used for the calibration. Surrogates are typically simple and can have runtimes on the order of seconds. This is a major advantage over the full model, which may have run times on the order of hours. With a properly built and validated surrogate, Bayesian calibration is in the realm of possibility, even for computationally expensive codes. The following steps outline an example process that could be used to build a surrogate [1]:

1. Generate a Latin Hypercube Sampling (LHS) design to use to train the surrogate.
2. Generate a LHS design to test the surrogate.
3. Run the model using the testing design to build the surrogate.
4. Run the testing design to validate the surrogate against the model output.

Once the surrogate is built and validated, it can be used in place of the full model to perform Bayesian calibration, if the surrogate behaves sufficiently similar to the full model.

4.3 Bayesian Calibration of MAMBA

The plan for Bayesian calibration of MAMBA is outlined below. This plan is a work in progress and may change in the future. This section is a summary of a plan presented by Adam Hetzler (SNL) [5].

4.3.1 Calibration Goal and Benefit

The goal of the calibration is to make predictions of CIPS with uncertainty over different loading patterns to determine if there is a significant difference between high, medium, or low risk loading patterns [5]. If the calibration shows that the predictions of CIPS over the different loading patterns is not significant, this can potentially offer a huge benefit to industry as it would provide evidence for utilities to move from low risk loading patterns.

The quantity of interest for the calibration is the axial offset associated with specific loading patterns. The calibration study will need to analyze three different loading patterns (high, medium, and low risk). The results of this study will be used to determine if the differences in the axial offset between the different loading patterns is significant.

MAMBA is the main goal for the Bayesian calibration as it is the greatest source of uncertainty for the prediction of CIPS. It may be possible to perform Bayesian calibration of other coupled models, but it may be impractical to pursue more than one calibration. The MAMBA Bayesian calibration is the primary goal and if it is the only calibration that can be completed given time and computational constraints, the uncertainty can be propagated to the other models.

4.3.2 Calibration Plan and Risks

This calibration process will require collaboration from different focus areas in CASL to be successful. An outline of the process is below:

1. Data collection
2. Surrogate model creation
3. Determine prior distributions on model parameters
4. Bayesian calibration of MAMBA
5. Uncertainty from MAMBA is propagated through the models to estimate uncertainty in the quantity of interest

The first step for the calibration will be data collection from different focus areas in CASL. The data collection also presents a major risk for the calibration. There must be a sufficient quantity of relevant and high-quality data for the different loading patterns to make the calibration useful and meaningful. The data must be collected and then assessed to make sure both the quality and quantity is sufficient before moving forward.

The second step of the calibration is to build surrogates. Due to the coupled nature of MAMBA, it may be necessary to build surrogates for different models, which will require input from other focus areas. Surrogates will need to be built for any model that runs too slowly to be directly used during the Bayesian calibration.

It is possible that a surrogate will be needed for MAMBA and its source term as well, however the current implementation plan assumes that a surrogate will not be used for MAMBA itself. The reason for this is the number of internal evaluations performed is on the order of 84 million per single MAMBA evaluation. Any surrogate model built to capture this quantity of information in a functional form is likely to have a run time on the order of MAMBA (which runs quickly compared to MPACT). Additionally, the time-dependent nature of MAMBA will need to be captured in the surrogate model, which will make the surrogate building process difficult to implement. In other words, it is likely that the end result of the creation of a MAMBA surrogate would not be worth the effort required, however both MPACT and CTF (which are coupled to MAMBA) will need surrogates.

The number of MAMBA evaluations during Bayesian calibration will ultimately be determined by the computational resources and time available to run the calculations. The current thought process is that we will run as many evaluations as possible within the time window available. Obviously, more

evaluations completed is better for the Bayesian calibration, however there are other considerations to be made, such as the number of plants used for the calculations. Additional plants will require us to reduce the total number of evaluations, which may or may not be an acceptable trade-off.

It is not possible to determine beforehand how many model runs will be needed to build an adequate surrogate, however a starting number of 200 model evaluations can be used to both build and then test the surrogate. It may also be possible to leverage data instead of model runs to build surrogates (i.e. MAMBA source term). As with any iterative study, prematurely stopping the evaluations before the surrogate construction or Bayesian calibration is complete can result in an unconverged surrogate or calibration, which is a major risk. Ideally, the number of evaluations for the surrogate construction and calibration would be determined by Dakota (which iterates until it reaches convergence), however due to time and resource limitations, this is not possible.

Before proceeding with Bayesian calibration, it is also necessary to determine the prior distributions on the model or surrogate parameters. These “priors” are the PDFs determined based on beliefs or previous knowledge of the model parameters. Input will be needed from the different focus areas to determine appropriate PDFs.

During the Bayesian calibration, a MCMC (Markov Chain Monte Carlo) sampler will accept the priors, model form, and data for the scenario. Sampling options such as the size and number of chains will also be specified. The sampler will determine the posterior distributions on the parameters, which will be outputted and used to propagate uncertainty through the various models and surrogates. This process will result in an estimate of uncertainty in the quantity of interest (axial offset).

Dakota includes four different methods for Bayesian calibration and will be used for the Bayesian calibration of MAMBA [3]. The most appropriate method for the calibration will be selected with input from Ralph Smith and Brian Adams.

5. CONCLUSIONS

The setup for a deterministic calibration of MAMBA was completed for this milestone. Errors in MAMBA made it not possible at the time of the milestone completion to attempt a deterministic calibration, however the framework is in place and can be used in the future. Additionally, a plan for Bayesian calibration of MAMBA was laid out during the work performed to complete this milestone. The Bayesian calibration of MAMBA will require participation across several CASL focus areas and have valuable impact for industry. This work will be carried out in the future.

LIST OF REFERENCES

[1] N. Gordon, "L3:VVI.H2L.P15.01 Milestone Report," CASL, 2017.

[2] B. Collins, J. Galloway, R. Salko, A. Wysocki, K. Clarno, B. Okhuysen, S. Slattery, N. Adamowicz, D. Andersson, A. Manera, D. Pointer, A. Neukirch, W. Gurecky and V. Petrov, "Development of a Comprehensive Crud-Induced Power Shift (CIPS) Capability with VERA," CASL, August 2017.

[3] B. Adams, L. Bauman, W. Bohnhoff, K. Dalbey, M. Ebeida, J. Eddy, M. Eldred, P. Hough, K. Hu, J. Jakeman, J. Stephens, L. Swiler, D. Vigil and T. Wildey, "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 Theory Manual," Sandia Technical Report SAND2014-4253, July 2014, Updated November 2017 (Version 6.7).

[4] R. Muehleisen and J. Bergerson, "Bayesian Calibration - What, Why And How," in *International High Performance Buildings Conference*, 2016.

[5] A. Hetzler, N. Gordon, D. Brooks and L. Gilkey, Personal communication, 2018.

APPENDIX A

A.1 Dakota Input File

```
environment
  tabular_data
    tabular_data_file 'dakota_results.dat'
    custom_annotated
    header
    eval_id
    interface_id
    graphics

  results_output
    results_output_file 'dakota_final_results'

method,
  nl2sol
        #ncsu_direct
    max_iterations = 100
          #max_function_evaluations = 200
    convergence_tolerance = 1e-4
        #output quiet

variables,
  continuous_design = 4
    descriptor        'ksnb'    'por'   'htc'   'pmult'
    initial_point     7.2E-4    0.7     40.0     1.0
    lower_bounds      1.0E-4    0.6     20.0     0.7
    upper_bounds      7.5E-4    0.8     60.0     1.3


interface,
  fork
        failure_capture continuation
        asynchronous
        evaluation_concurrency  9

    work_directory named 'eval'
  analysis_drivers = 'driver.sh'
  parameters_file = 'params.in'
  results_file = 'results.out'
  file_save
  directory_save


responses,
calibration_terms 1
  calibration_data
  num_experiments 1
  field_calibration_terms 1
  lengths 30
  descriptors 'wb'
  numerical_gradients
        #no_gradients
  no_hessians
```

A.2 Dakota Driver

```
#!/bin/bash
params=$1
results=$2

cases='c7 c8'
nprocs=193

for case in ${cases}; do
  mkdir $case
  cd $case && rm *

  dprepro ../$params ../../templates/mambaPL.xml.template mambaPL.xml
  dprepro ../$params ../../templates/${case}.mstate.inp.template
${case}.mstate.inp

  ln -s ../../data/${case}.h5 .
  for ((i=1;i<=${nprocs};i++)); do
    cp ../../models/deck.${nprocs}.${i}.inp ${case}.${nprocs}.${i}.inp
  done
  cp ../../models/deck.master.inp ${case}.master.inp
  cp ../../templates/runctf.sh.template runctf.sh
  sed -i "s/{case}/$case/g" runctf.sh

  qsub runctf.sh
  cd ..
done

for case in $cases; do
  while [ ! -f ${case}/DONE ]; do
    sleep 30
  done
  echo ${case} " has finished"
done

python ../scripts/post_process_data.py $cases
```

A.3 Dakota Templates

A.3.1 c7.mstate.inp.template

```
[control]
model_crud 1
crud_tool 3
mamba_param 'mambaPL.xml'

[states]

! STATE  1
[state]
day       0.00
boron    1754.25
fesol       2.10
hydrogen       0.00
lithium       3.34
nipar       {2.75*pmult}

! STATE  2
[state]
day       5.60
boron    1112.53
fesol       2.10
hydrogen       0.00
lithium       3.35
nipar       {2.79*pmult}

! STATE  3
[state]
day      19.80
boron    1117.16
fesol       2.12
hydrogen       0.00
lithium       3.37
nipar       {2.90*pmult}

! STATE  4
[state]
day      33.20
boron    1138.05
fesol       2.13
hydrogen       0.00
lithium       3.41
nipar       {3.24*pmult}

! STATE  5
[state]
day      62.00
boron    1174.98
fesol       2.16
hydrogen       0.00
lithium       3.49
nipar       {4.01*pmult}

! STATE  6
[state]
day      90.20
boron    1179.25
```

```
fesol       2.15
hydrogen       0.00
lithium       3.49
nipar       {4.48*pmult}

! STATE  7
[state]
day     118.10
boron    1156.26
fesol       2.11
hydrogen       0.00
lithium       3.42
nipar       {4.75*pmult}

! STATE  8
[state]
day     151.80
boron    1100.52
fesol       2.06
hydrogen       0.00
lithium       3.26
nipar       {4.92*pmult}

! STATE  9
[state]
day     172.60
boron    1055.01
fesol       2.02
hydrogen       0.00
lithium       3.12
nipar       {5.13*pmult}

! STATE 10
[state]
day     199.40
boron     987.25
fesol       1.99
hydrogen       0.00
lithium       2.94
nipar       {5.05*pmult}

! STATE 11
[state]
day     227.30
boron     907.71
fesol       1.96
hydrogen       0.00
lithium       2.73
nipar       {4.94*pmult}

! STATE 12
[state]
day     255.40
boron     820.46
fesol       1.94
hydrogen       0.00
lithium       2.52
nipar       {4.83*pmult}

! STATE 13
```

```
[state]
day     283.30
boron     728.22
fesol       1.91
hydrogen       0.00
lithium       2.27
nipar       {4.72*pmult}

! STATE 14
[state]
day     311.10
boron     631.92
fesol       1.89
hydrogen       0.00
lithium       2.02
nipar       {4.62*pmult}

! STATE 15
[state]
day     343.70
boron     515.16
fesol       1.88
hydrogen       0.00
lithium       1.75
nipar       {4.65*pmult}

! STATE 16
[state]
day     361.30
boron     450.88
fesol       1.87
hydrogen       0.00
lithium       1.61
nipar       {4.59*pmult}

! STATE 17
[state]
day     387.50
boron     354.90
fesol       1.84
hydrogen       0.00
lithium       1.34
nipar       {4.51*pmult}

! STATE 18
[state]
day     415.70
boron     250.31
fesol       1.81
hydrogen       0.00
lithium       1.08
nipar       {4.46*pmult}

! STATE 19
[state]
day     421.50
boron     228.89
fesol       1.78
hydrogen       0.00
lithium       0.74
```

```
nipar       {4.80*pmult}
```

A.3.2 c8.mstate.inp.template

```
[control]
model_crud 1
crud_tool 3
mamba_param 'mambaPL.xml'

[states]

! STATE  1
[state]
day       0.00
boron   1.82232448e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE  2
[state]
day       5.13
boron   1.27582519e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE  3
[state]
day      15.58
boron   1.18483998e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE  4
[state]
day      20.18
boron   1.18283387e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE  5
[state]
day      31.08
boron   1.18308482e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE  6
```

```
[state]
day      41.08
boron   1.18399556e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE  7
[state]
day      70.28
boron   1.17305489e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE  8
[state]
day      96.08
boron   1.13871287e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE  9
[state]
day     124.08
boron   1.08202896e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 10
[state]
day     152.08
boron   1.01052217e+03
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 11
[state]
day     180.08
boron   9.31764543e+02
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 12
```

```
[state]
day     208.08
boron   8.40828127e+02
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 13
[state]
day     236.08
boron   7.43056325e+02
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 14
[state]
day     263.78
boron   6.42374988e+02
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.15000000e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 15
[state]
day     291.58
boron   5.38857039e+02
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.13021314e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 16
[state]
day     319.38
boron   4.32867545e+02
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   2.04752106e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 17
[state]
day     347.38
boron   3.26250574e+02
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   1.82883296e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 18
```

```
[state]
day     375.28
boron   2.19495122e+02
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   1.42904365e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 19
[state]
day     403.38
boron   1.11930175e+02
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   1.02584058e+00
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 20
[state]
day     420.08
boron   4.91410716e+01
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   6.19324678e-01
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}

! STATE 21
[state]
day     436.88
boron   1.00000000e-07
fesol   1.99000000e+00
hydrogen   3.50000000e-05
lithium   1.00000000e-02
nisol   2.07000000e-01
nipar   {4.32000000e+00*pmult}
```

A.3.3 mambaPL.xml.template

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet version="1.0" type="text/xsl" href="PL9.xsl"?>
<ParameterList name="CRUD_Pin">
   <Parameter name="ksnb_Fe2O4" type="double" value="{ksnb=0.72E-3}"/>
   <Parameter name="CRUD_porosity" type="double" value="{por=0.7}"/>
   <Parameter name="chimney_htc" type="double" value="{htc=6.7E2}"/>
</ParameterList>
```

A.3.4 runctf.sh.template

```
#!/bin/bash
#
#PBS -N calibrate
#PBS -l nodes=6:ppn=36
#PBS -l walltime=24:00:00
#PBS -n node-exclusive
#PBS -V
#PBS -o run.stdout
#PBS -j oe
#PBS -m ea
#PBS -M bn7@ornl.gov
#PBS -q batch

case={case}
nproc=193

date >> output.dat
cd $PBS_O_WORKDIR
pwd >> output.dat

#source /home/tools/gcc-4.8.3/load_dev_env_mod.sh
ctf=/home/bn7/ctf_calibration/build_dir/install/bin/multistate_cobra

rm DONE

mpirun -np $nproc $ctf $case   >> output.dat

touch DONE
```

A.4 Post-Processing Script

```
import h5py
import numpy as np
import sys

#fidi='assem19.h5'
cases=sys.argv[1:]

states=[4,10,16]
spans=[0,26,32,40,47,52]

fido='results.out'
f=open(fido,'w')

def reduce_data(boron,istt,istp):
  return np.sum(boron[:,:,istt:istp,:],axis=(0,1,2))

for case in cases:
  print case
  fidc=case+'/'+case+'.ctf.h5'
  prd_dataset='pin_avg_crud_borondensity'  #CTF
  fidm=case+'/'+case+'.h5'
  ref_dataset='pin_crud_boron_dens'  #MPACT

  h5ctf=h5py.File(fidc,'r')
  h5ref=h5py.File(fidm,'r')
  #get number of states.. eventually
  print('reading file '+fidc)
  try:
    for state in states:
      path='/STATE_{0:04d}/'.format(state)
      bctf = h5ctf[path+prd_dataset].value
      try:
        bref = h5ref[path+ref_dataset].value
      except:
        print "no reference found"
        bref = bctf*0.0
      for i in xrange(len(spans)-1):
        bspanctf=reduce_data(bctf,spans[i],spans[i+1])
        bspanref=reduce_data(bref,spans[i],spans[i+1])
        x=np.linalg.norm(bspanctf-bspanref)
        f.write(str(x)+'\n')
  except:
    print('this case failed')
    f.close()
    f=open(fido,'w')
    for x in xrange(len(states)*(len(spans)-1)*56):
      f.write('10.0E12\n')
    exit()

  h5ctf.close()
  h5ref.close()

f.close()
```