UQ Tool (Task 2.3) Documentation

GPIC EEB Hub Confidential

Slaven Peleš, Sunil Ahuja and Satish Narayanan

February 1, 2012

This document for now contains documentation for subtask 2.3 of GPIC initiative.

Contents

List of Figures

1 Introduction

1.1 Objectives

The overall project goal is to prototype and evaluate a systematic methodology and easily accessible tools for performing rapid uncertainty quantification, sensitivity analysis, and parameter investigations within an integrated deep retrofit design process, leveraging accessible and affordable high performance computing and cloud infrastructure. UTRC team objective in support of this is to develop energy simulation use cases for conventional and deep building retrofit designs for process and tool demonstration, and to support the evaluation of alternative uncertainty propagation and parametric sensitivity analysis techniques.

A significant challenge facing the use of modelling and simulation during energy efficient building design is answering the question "what is the certainty of my computed answer?". Whole building simulation and energy performance models contain thousands of uncertain parameters, many of which are dynamic, which induce large variability in energy performance estimates. Energy predictions can be off by 30%, or in some cases even more, relative to design intent. These problems manifest in the design-construction-operation of a building where energy performance predictions are used to prioritize, select and specify integrated solutions during design, and then to compare against performance measurement and verification during commissioning and operation. New methods and computational tools, beyond one-time simulation of nominal energy performance, are required to quantify the achievable performance. However, brute-force Monte Carlo methods-based analyses of building energy simulations are computationally prohibitive and unaffordable. Within DOE (specifically NNSA) and DoD (DARPA/DSO) there are several large R&D activities developing rapid uncertainty quantification (UQ) tools and methodologies. This project applied and adapted these tools and frameworks to building energy simulation with the specific objective to make them accessible, affordable and easy-to-use for building designers. For the next year, the team will focus on adapting, and evaluating existing UQ tools on either HPC or cloud infrastructures for two uses cases for buildings retrofit design.

1.2 Uncertainty Analysis

Uncertainty analysis methods either converge slowly (e.g. Monte Carlo) or they suffer from dimensionality curse (e.g. stochastic collocation). Because of these inherent properties of uncertainty quantification methods, the computational cost grows exponentially with the size of the problem. This means that simply throwing in more computational

power will not enable one to analyse a large system. For example, to perform uncertainty quantification for 200 uncertain building parameters, taking into consideration parameter interactions up to the order of 10 and using full collocation grid, it would take roughly 351 billion years for computation to complete on a single processor machine. There is no physical solution in sight that can compress such simulation to run within a reasonable amount of time. Therefore, it is necessary to develop mathematical methods that scale better with the size of the problem [1, 2].

Current state of the art methods for uncertainty quantification do not remove the curse of dimensionality altogether, they simply ameliorate the consequences of the curse. Still, big practical improvements have been accomplished recently by increasing number of uncertain parameters that can be studied simultaneously to several hundreds from less than ten. Particularly effective are methods for detecting uncertain parameters to which system is not sensitive. These methods scale linearly and can help reduce the size of the problem significantly. There are other techniques, such as quasi-random number generation, which can significantly improve convergence of Monte Carlo methods, as well as sparse grid methods, which can significantly reduce amount of probabilistic samples needed to cover given volume in parameter space. To be effective, uncertainty analysis process needs to be carefully tailored to the system under investigation. Therefore, the onus of the research should not be only on developing new uncertainty analysis methods, but also on developing processes that utilize those methods in the most efficient way.

The Uncertainty Quantification Tool is intended to provide a complete solution for uncertainty quantification, sensitivity analysis and calibration of building models and also to become platform for development of novel methods for uncertainty analysis. The tool should be designed to use standard building simulators, such as TRNSYS or EnergyPlus, to produce probabilistic samples, so that same models that are used for deterministic simulations can be used for uncertainty analysis. Finally, the tool should be flexible enough to allow the user to tailor uncertainty analysis process to the problem at hand.

1.3 Summary of Year 1 Accomplishments

In Year 1 process flow for uncertainty quantification, sensitivity analysis and model calibration was designed. The process relies on a standard buildings simulator to collect probabilistic data required for the analysis and it is not simulator specific. All Year 1 analyses were performed using TRNSYS as the buildings simulator and DOE medium size office benchmark building model. Key components of the process were implemented in C++ within Uncertainty Quantification Tool. In addition to that, unit tests were created for each component, so that implemented algorithms can be verified automatically whenever changes are made to the tool.

Sensitivity analysis for hundreds of parameters in a whole building model was demonstrated and key building parameters were identified. The results supported assertion that only a handful of parameters significantly affects energy performance of a typical building. Meta model (sometimes referred to as reduced order model) was obtained form

from sensitivity analysis with little additional cost. The meta model was obtained in closed analytical form and it was used for computationally intensive uncertainty quantification and model calibration. Performing calibration with full TRNSYS model was not computationally feasible. Quantitative measures for the accuracy of sensitivity analysis were defined and included in the computation. They were used to assess if the meta model approximated well the original system or additional analysis was required.

Model calibration was performed for subset of parameters identified by sensitivity analysis to affect building energy performance significantly. Calibrated values were then substituted back in the original TRNSYS model and results were compared. The calibration reduced model prediction error from as much as 45% to about 5%.

The Year 1 results suggest that it is possible to perform efficient whole building model calibration with turnaround time of 1-2 days, provided reasonable computational resources are available. Same computational tools can be used for performance optimization. First order Sobol analysis captured $> 90\%$ of overall sensitivity in the DOE benchmark building model. It is unlikely this would always be the case, so higher order analyses (such as collocation methods) need to be implemented within the tool, as well. Furthermore, process flow control needs to be added to guide user through the analysis and provide available analysis methods at every step of the way.

All methods implemented so far apply to parametric uncertainties only. Uncertain processes, such as weather or occupancy patterns affect building energy performance significantly and need to be included in the analysis in the near future. Quantifying effects of uncertain processes is an open research problem and is far more challenging than parametric uncertainty quantification.

Overview of the Uncertainty Analysis Tool and implemented algorithms is given in Chapter 2. Detailed description of the Year 1 accomplishments and results is presented in Chapter 3. Proposed work for Year 2 is outlined in Chapter 4.

2 Uncertainty Analysis Tool

2.1 General Process

Commonly used buildings simulation packages rely heavily on legacy code and embedding numerical methods within such packages is difficult if not impossible. Fortunately, efficient uncertainty analysis can be carried out entirely with black-box methods. These methods use repeated simulations of deterministic model to produce statistical data needed for uncertainty quantification. Each deterministic simulation is performed with input parameters perturbed to reflect uncertainty associated with them.

General black-box uncertainty analysis process has three key steps:

- Generating probabilistic parameter samples.
- Running simulations for each sample.
- Performing analysis on simulation data

Parameter perturbations are done according to some probabilistic rule. For example, parameters can be selected randomly from input probability density function. Each deterministic simulation is run for different sample of input parameters. Running deterministic simulations is the most time consuming part of the process. Typically, the rest of the process takes less time than only one deterministic simulation run. Output data from simulations is then analysed in order to assess and quantify effects of input uncertainty. Analysis can be as simple as calculating average value of all simulation trials, or it can be an elaborate multi-stage process [3].

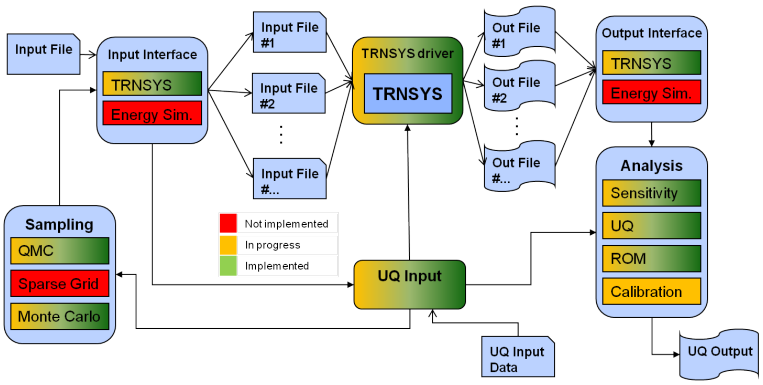

Figure 2.1: Process flow chart for uncertainty analysis

The current process flowchart is shown in Figure 2.1. Input information is provided by the user and part of it is pulled from the simulator input file. Based on user's choice of the analysis type and related parameters, a sampling strategy is created. The simulation engine interface then creates input files and scripts necessary to run all required deterministic simulations. The simulation engine driver manages input files, execution schedules and storage of raw output files from the simulator. Information necessary for the analysis is extracted from the output files by the simulator output interface and passed to the analysis block. Currently, the tool can perform basic sensitivity analysis, create a meta model and carry out model calibration. As the tool develops it should include more elaborate analysis process that provides feedback to the user about quality of the obtained results and suggests possible secondary analysis to further refine results.

Secondary analysis is particularly important in cases where sensitivities are dominated by multiple parameter interactions and cannot be well approximated by a sum of single parameter sensitivities. There are no general purpose methods that can handle a large number of uncertain parameters in situations like that, but in many cases relevant to buildings design a sequence of different analyses can be tailored to the problem at hand to produce accurate results at reasonable computational cost.

2.2 Input Data

All data necessary for the analysis is stored in a single class. Other parts of the code access that data by reference or, for small subsets, make temporary copies. The input data is supposed to be accessed and set by user interface. That part of the code has not been developed yet and temporary interface is used to set up computations. Input data consists of:

- Names of sampling and analysis methods
- General parameters required to set up those methods (e.g. number of Monte Carlo simulations required for the analysis)
- Data specific for each uncertain input. Currently this includes:
  - Uncertain input ID
  - Nominal value
  - Tolerance
  - Type of uncertainty
  - Polynomial expansion order (optional)

The inputs from first two items are entered through a text file. Uncertain inputs data is entered using TRNSYS graphic user interface and then read from the TRNSYS input file generated by the interface.

TRNSYS mark-up uses exclamation mark to denote a comment in the input file. Everything entered after the exclamation mark is ignored by the simulation engine.

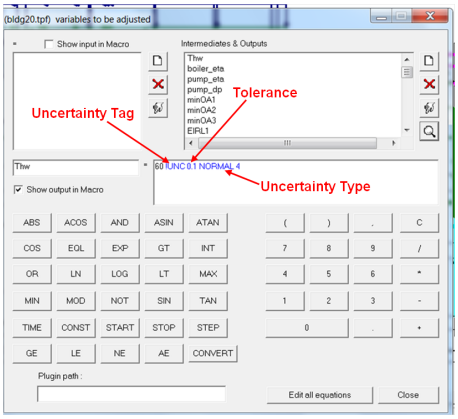

Figure 2.2: TRNSYS graphic user interface is used to enter uncertain inputs data. The data is stored as a comment in the TRNSYS input file.

Graphic user interface allows user to enter comments next to variables as shown in Figure 2.2. To take advantage of this feature, we defined tag "!UNC" to denote uncertain input data entry. For each uncertain input user adds a comment that begins with this tag and is followed by value for the tolerance, uncertainty type tag and polynomial expansion order separated by blank spaces. Solver interface can extract this information from TRNSYS input file and store it for the analysis.

Tolerance is entered as a relative tolerance, a number in interval $(0, 1)$. The absolute tolerance is calculated by multiplying this number to the nominal value of the uncertain parameter. If the nominal value is zero, then the tolerance is interpreted as the absolute tolerance expressed in the same units as the nominal value (this feature is not yet implemented).

Currently three types of uncertainty are supported, based on probability density function that describes the uncertainty. These are normal (Gauss), uniform and exponential uncertainty. Corresponding tags are all capital "NORMAL", "UNIFORM" and "EXPONENTIAL".

2.3 Sampling

In order to perform analysis, sufficient amount of probabilistic data needs to be produced by running repeated deterministic simulations. Each deterministic simulation is run with uncertain parameters perturbed according to some mathematical prescription. In a simplest case, we choose parameters by sampling them randomly from probabilistic density functions for each uncertain input (Monte Carlo method). The sampling range and the type of the probabilistic density functions need to be set by the user. These uncertain input properties are not always known. Sometimes equipment or material manufacturers will conduct extensive material testing and compile deviations from nominal values into a probability density function. More often, however, no reliable information is available,

so the user needs to conduct testing on his/her own or estimate uncertain input properties based on prior experience. Currently the tool supports Monte Carlo sampling, as well as two quasi Monte Carlo methods.

Random sample $x^i = (x_1^i, \ldots, x_d^i)$ is computed under assumption that all uncertain inputs are uncorrelated, so uncertain parameter values x_j^i are calculated independently as

$$x_j^i = \bar{x}_j + \varepsilon_j \rho_j^{-1}(\xi_j^i) \tag{2.1}$$

where $\bar{x}_j$ is nominal value of the uncertain parameter, ε_j is relative tolerance, ρ_j is cumulative probability density function describing type of the uncertainty, and $0 < \xi_j < 1$ is a random number. The difference between Monte Carlo and quasi Monte Carlo methods is in the way random numbers ξ_j^i are generated. The former uses plain random number generator, while the latter uses quasi random number sequences, which ensure more uniform sampling space coverage.

We refer to $x^i = (x_1^i, \ldots, x_d^i)$ as a physical sample, and to $\xi^i = (\xi_1^i, \ldots, \xi_d^i)$ as mathematical sample.

2.3.1 Monte Carlo Sampling

Random numbers ξ_j^i are generated one by one, independently one from another, when using Monte Carlo sampling. There are numerous methods for generating random numbers. A good random number generator will produce a sequence of numbers negligible correlations between them. A downside of this approach is that sampling space will not be uniformly covered, and some clustering of samples will be noticeable. Because of that convergence of Monte Carlo methods is slow and numerical error scales as $1/\sqrt{N}$, where N is the number of random samples.

2.3.2 Quasi Monte Carlo Sampling

Quasi random number sequences are designed to improve uniformity of the sampling space coverage. Random samples depend on dimensionality d of the sampling space, and components of each sample $\xi^i = (\xi_1^i, \ldots, \xi_d^i)$ are generated simultaneously. The numerical error of quasi Monte Carlo method is theoretically evaluated to be of order $(\log N)^d/N$ [citation], where N is the number of samples. In some cases it is observed that the error diminishes even faster, at $1/N$ rate. Quasi random number sequences are typically based on arrays of integer numbers, which are rescaled to the interval $[0, 1)$ by suitable division. The integer number arrays often have correlations at certain dimensions, which can produce numerical artefacts in the final result. This is remedied by bit scrambling of integer number arrays. Each integer in the array has its bits permuted according to some rule. Devising methods for producing quasi random numbers that cover sampling space uniformly and show little correlation amongst themselves is a subject of active scientific research. In the Uncertainty Analysis Tool we implemented Joe-Kuo variant of Sobol sequence [4] and Halton sequence with Kocis-Whiten scrambling [5].

2.4 Solver Input Interface

Once physics samples are computed they are read by the solver input interface, who then generates input files for the simulation engine. The interface needs to parse original input file for the building model and replace nominal values for the uncertain parameters with the sampled values. In the future solver input interface will generate shell scripts required to run all the samples on hardware architecture selected by the user.

2.4.1 TRNSYS

Unlike most of other tools TRNSYS uses two text files to store its input data. The main input file (typically with extension "dck") contains the information about building equipment, output files and input file with building envelope data. The TRNSYS model can be structured in a way that all uncertain parameters in the main input file have unique names, and that uncertainty input data can be entered as comments (Figure 2.2). The input file with building envelope parameters has somewhat different structure, most notably it does not have unique parameter names, and there is no simple way to tag uncertain parameters in there using TRNSYS graphic user interface. As a temporary solution, the input user interface expects third input file with uncertainty information for the building envelope. This file needs to be prepared by the user. TRNSYS input interface will generate pairs of equipment-envelope input files for each probabilistic sample.

2.5 Simulation Engine Driver

This module should manage simulation jobs scheduling, output storage, stopping and restarting jobs, getting job reports and restarting failed jobs. Current implementation provides option to run all simulations as a serial process or to run a shell script on an arbitrary machine.

2.6 Solver Output Interface

Simulation engine will typically store its output in text files. Solver output interface needs to parse those files and extract information needed for the analysis. In the case of TRNSYS there is no standard output file format. Current implementation of the output interface is designed to use output format of the DOE Benchmark Building model. In the future a standard output format for uncertainty analysis needs to be defined and appropriate documentation including templates needs to be provided for TRNSYS developers. It should be then up to the model developer to add TRNSYS output for uncertainty analysis.

2.7 Sensitivity Analysis

Typical building model has hundreds of uncertain parameters and because of that sensitivity analysis methods that can handle high dimensional problems are at the core of the building performance analysis. Variance based methods are particularly important since even when they fail to capture all sensitivities in the system, they can still provide a quantitative measure of the amount of sensitivity in the system that has not been accounted for.

The strategy we use for sensitivity analysis is to first account for sensitivities due to single parameter perturbations. Such computation scales *linearly* with the number of uncertain parameters, and large problems can be attacked by providing more computational power. From there we find first order sensitivity indices and amount of sensitivity not captured by first order analysis. If unaccounted sensitivity is a small fraction of the overall sensitivity, then all necessary information for further analysis is contained within sampling data. Otherwise, higher order sensitivity analysis needs to be done, preferably after some model reduction based on first order analysis results.

A model $f(x)$, were $x = (x_1, \ldots, x_d)$ and $0 \leqslant x_i \leqslant 1$, can be expanded in terms of analysis of variance (ANOVA) decomposition as

$$f(x) = f_0 + \sum_{i=0}^{d} f_i(x_i) + \sum_{i=0}^{d} \sum_{j<i} f_{ij}(x_i, x_j) + \ldots + f_{1,2,\ldots,d}(x_1, x_2, \ldots, x_d) \tag{2.2}$$

where

$$\int_0^1 f_{i_1 i_2 \ldots i_s}(x_{i1}, x_{i2}, \ldots, x_{is}) dx_{i_k} = 0, \ \forall k : \ 1 \leqslant k \leqslant s \tag{2.3}$$

From this definition it follows that f_0 is the mean value of $f(x)$. Variance expansion is then given as

$$D = \sum_i D_i + \sum_i \sum_{j<i} D_{ij} + \ldots + D_{1,2,\ldots,d} \tag{2.4}$$

where

$$D_{i_1 i_2 \ldots i_s} = \int_0^1 f^2_{i_1 i_2 \ldots i_s}(x_{i1}, x_{i2}, \ldots, x_{is}) dx_{i_1} \ldots dx_{i_s} \tag{2.5}$$

is a partial variance and

$$D = \int_0^1 (f(x) - f_0)^2 dx \tag{2.6}$$

is the total variance. Sobol sensitivity index is defined as a fraction of total variance

$$S_{i_1 i_2 \ldots i_s} = \frac{D_{i_1 i_2 \ldots i_s}}{D^2} \tag{2.7}$$

By definition all Sobol indices sum up to one.

$$\sum_i S_i + \sum_i \sum_{j<i} S_{ij} + \ldots + S_{1,2,\ldots,d} = 1 \tag{2.8}$$

First order Sobol index is defined as

$$S_i = \frac{D_i}{D} \tag{2.9}$$

When $S_i \approx 1$ the system is sensitive only to x_i. Sum of all first order indices gives the fraction of uncertainty due to single parameter perturbations. The value of this sum helps determine if higher order sensitivity analysis is necessary to assess overall system sensitivity. Another useful quantity is first order *total* Sobol index, which is defined as

$$T_i = \frac{D_i + \sum_j D_{ij} + \sum_j \sum_{k<j} D_{ijk} + \ldots + D_{1,2,\ldots,d}}{D^2} \tag{2.10}$$

When $T_i \approx 0$ the system is not sensitive to x_i. Total Sobol index is typically used to eliminate parameters that the system is not sensitive to from further analysis. Finally, derivative based first order sensitivity index is defined as

$$v_i = \int_0^1 \left(\frac{\partial f}{\partial x_i}\right)^2 dx \tag{2.11}$$

Relationship between the three types of first order sensitivity indices is:

$$0 \leqslant S_i \leqslant T_i \leqslant v_i \leqslant 1 \tag{2.12}$$

Derivative based sensitivity index is the upper bound of the total Sobol index, but in practice it is much cheaper to evaluate, and is used instead of the total Sobol index to eliminate low sensitivity parameters.

Mean value f_0 and total variance D are computed directly from the simulation data. Partial variances could be calculated from separate sets of simulation data where only a subset of uncertain parameters is perturbed. However, computationally more efficient way to compute partial variances is response surface method where the data for the mean and total variance calculation can be reused. Let us assume that model $f(x)$ can be expanded in terms of orthonormal polynomials ϕ_k and consider response surface in the form

$$f_j(x_j) = \sum_{k=0}^{\infty} a_{jk}\phi_k(x_j), \tag{2.13}$$

where

$$\int_0^1 \phi_i(x)\phi_j(x)\rho(x)dx = \delta_{ij}. \tag{2.14}$$

For simplicity and without loss of generality we will assume that $\rho(x) = 1$. Partial variance then can be expressed in terms of polynomial expansion coefficients as

$$D_j = \sum_{k=1}^{\infty} (a_{jk})^2, \tag{2.15}$$

and therefore Sobol first order index can be computed as

$$S_j = \frac{1}{D}\sum_{k=1}^{\infty}(a_{jk})^2. \tag{2.16}$$

If polynomial expansion converges, we can make a cut-off at suitable order P and approximate Sobol index as

$$\hat{S}_i = \frac{1}{D}\sum_{k=1}^{P} a_k^2. \tag{2.17}$$

Obviously, only a finite approximation at relatively small order is useful for practical applications. Response surface for the entire system is

$$f(x) \approx \sum_{j=1}^{d}\sum_{k=0}^{P} a_{jk}\phi_k(x_j) \tag{2.18}$$

Polynomial coefficients can be calculated from simulation data using for example least square fit. The number of polynomial coefficients to find, and therefore computational cost, grows linearly with the dimension of the system d.

2.8 Model Calibration

The sensitivity analysis and response surface (reduced-order model) of the original function $f(x)$ can then be used to calibrate the full-order model; that is, given a set of field or experimental measurements, the analysis could be used to compute the parameters that maximize the agreement of the model with the data. The response surface of the function $f(x)$ was described as

$$y = \tilde{f}(x) = \sum_{i=1}^{d}\sum_{k=0}^{P} a_k\phi_k(x_i) \approx f(x). \tag{2.19}$$

In general, the original function can be a vector $\mathbf{f}(x) = \big(f_1(x), f_2(x), \ldots, f_N(x)\big)$, in which case the response function is

$$y_n = \tilde{f}_n(x) = \sum_{i=1}^{d}\sum_{k=0}^{P} a_k^n\phi_k^n(x_i) \approx f_n(x), \qquad n = 1, 2, \ldots, N. \tag{2.20}$$

The calibration procedure, illustrated in figure 2.3, can be described in the following steps:

- For each output, order the parameters in the decreasing order of their Sobol indices. Compute their cumulative Sobol indices, and retain the parameters $\tilde{x}$ with a cumulative index smaller than a cut-off, say 0.95. Alternatively, retain all parameters with corresponding Sobol indices greater than a certain user-defined cut-off, say 0.05. Redefine the response surface (2.20) by retaining only terms containing the reduced set of parameters.

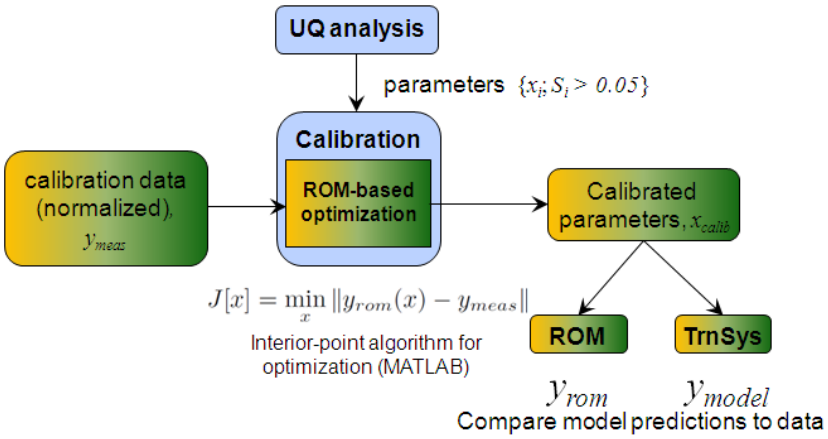

Figure 2.3: Schematic of model calibration process, using the results of a sensitivity analysis.

- Given measurements c_i, the calibration problem can be formulated as:

$$\min_{\tilde{x}} J[\tilde{x}] = \sum_{i=1}^{N} \alpha_i (y_i(\tilde{x}) - c_i)^2, \tag{2.21}$$

$$0 < \tilde{x}_i < 1, \qquad i = 1, 2, \ldots, d. \tag{2.22}$$

  The constants α_i are used to normalize the outputs with respect to certain nominal values; here, these are chosen to be the values of the function at the center of the domain $[0, 1]^d$. The optimization problem (2.22) can be solved using standard off-the-shelf solvers such as IPOPT [6] or NLOpt [7]. In this work, we just present a demonstration of the capability that is to be built into the tool-chain in the future. For that purpose, we use standard algorithms built into MATLAB®; in particular, we use the function fmincon, with an interior-point algorithm to solve (2.22).

- Compare the predictions of the calibrated model to the measurement data. That is, if the solution of the optimization problem (2.22) is $\bar{x}$, compute the error $e = \|\mathbf{f}(\bar{x}) - \mathbf{c}\|/\|\mathbf{c}\|$.

3 Year 1 Accomplishments

Basic groundwork for building analysis tools has been laid in the Year 1 of the project. Process for uncertainty quantification, sensitivity analysis and model calibration was defined and key software components were developed. Also, key infrastructure was created to support future development.

All demonstrations of the tool capabilities were done using a TRNSYS model of DOE medium-size office benchmark building. The benchmark building is a three storey building with total inside area of 15,000 ft^2, divided into 15 heating/cooling zones. A picture of the benchmark building model is shown in Figure 3.1.

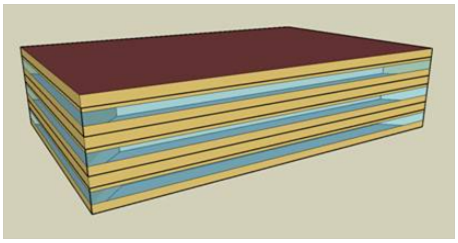

Figure 3.1: Model of DOE medium-size office benchmark building.

3.1 Prototype Process and Support Infrastructure

The guiding principle in defining process for uncertainty analysis was to reuse as much of existing tools and resources as possible and to make the process easy to adopt by general buildings community. The Uncertainty Analysis Tool was designed as a wrapper around a standard buildings simulator so that same models can be used for both, deterministic simulations and uncertainty analysis. The tool does not depend on the specific simulator. With a suitable plug-in any building simulator can be interfaced to the Uncertainty Analysis Tool. In Year 1, TRNSYS plug-in was delivered together with the rest of the code. Intermediate data is stored in text files (Figure 2.1), so it can be reused in different analyses. Collecting simulation data necessary for the uncertainty analysis is extremely time-consuming process, and storing intermediate results allows one to collect that data in stages and have more flexibility in managing computations. Detailed description of the uncertainty analysis process is given in Section 2.1.

Preliminary implementation of the Uncertainty Analysis Tool already has key sensitivity analysis capabilities built in. The tool was designed in a modular fashion, which allows for adding new features seamlessly. Model calibration is delivered in a form of Matlab prototype and it is not yet fully integrated with the rest of the code.

Unit tests were created for each segment of the process using test cases from peer reviewed publications [4, 5, 8, 9]. Unit tests are run automatically upon each rebuild of the tool and provide immediate verification of all algorithms used in uncertainty analysis.

3.2 Demonstration of Sensitivity Analysis

Sensitivity analysis was performed for 217 uncertain parameters with respect to 16 outputs in DOE benchmark building model. The outputs are annual energy consumptions of different parts of the building. Total of 4,500 quasi Monte Carlo simulations was run, using TRNSYS as the simulation engine. First order sensitivity analysis (Figure 3.2) shows that there are few parameters that the system is sensitive to; there are only 19 parameters that contribute to 10% or more of overall sensitivity for at least one output.

For example, annual gas consumption sensitivity is almost entirely dominated by the efficiency of the boiler (67%) followed by the hot water temperature (8%). All other parameters combined contribute to less than 25% of the gas consumption sensitivity. The sum of all first order Sobol indices for gas consumption is 99.5%, suggesting that higher order sensitivities are negligible and no further sensitivity analysis is needed. From this one can further deduce that model for gas consumption can be reasonably well calibrated by adjusting only two uncertain parameters.

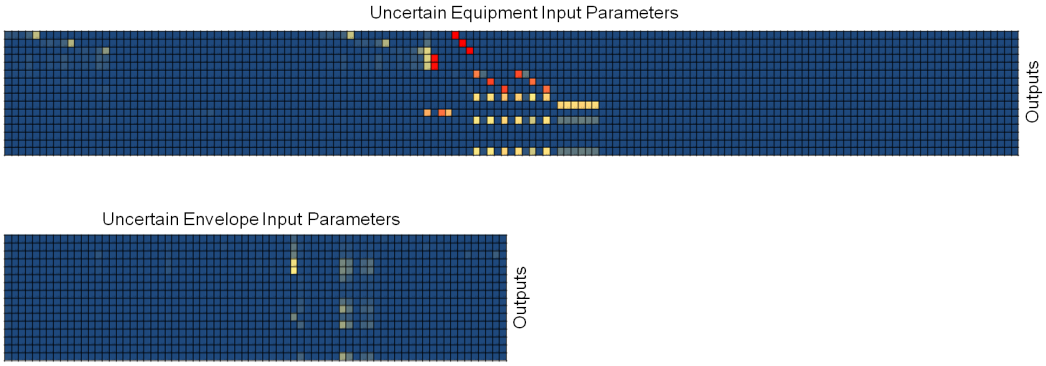

Figure 3.2: First order Sobol indices for 217 uncertain input parameters calculated for 16 system outputs. Color code denotes value of the Sobol index with dark blue indicating small and red large value of the index. Results for equipment parameters and building envelope parameters are shown on separate panels. Parameter and output labels cannot be shown on this scale.

In order to assess accuracy and reliability of results, computations of sum of first order Sobol indices and relative error of the least square fit were included in the sensitivity analysis. These two quantitative measures help determine whether the analysis was successful and help decide if and what additional analysis needs to be performed. Figure 3.3 shows these measures evaluated for all system outputs.

Large error in response surface fit may indicate that there were too few probabilistic

samples to get an accurate fit or that the form of the response surface is not suitable for the problem at hand. The latter means that either some important uncertain parameters were omitted from the analysis or that higher order sensitivities (those due to combined effects of several parameter perturbations) contribute significantly to the overall sensitivity.

If the sum of first order Sobol indices is significantly smaller than one, that suggests higher order sensitivities need to be investigated, and possibly some important parameters were not included in the analysis. If the sum is very close to zero, then it is quite certain that analysis needs to be expanded to cover additional parameters. It is highly unlikely that sensitivity is contained in higher orders, while single parameter perturbation do not affect the system significantly.

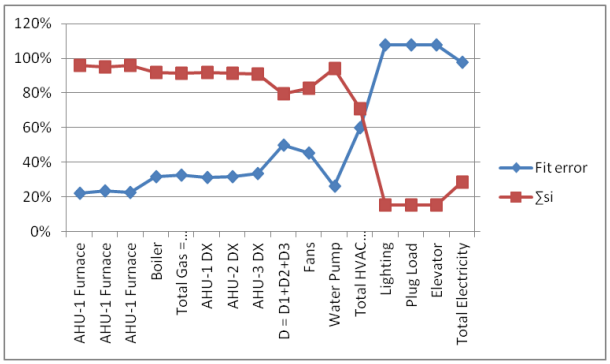

Figure 3.3: Quality of sensitivity analysis is checked by evaluating response surface fit error and sum of all Sobol indices.

In the example given in Figure 3.3 it is shown that sensitivity analysis was successful for all but three outputs, since in those cases response surface error was small. Furthermore, first order Sobol indices sum up almost exactly to one, suggesting that no further analysis is required. The three outputs where sensitivity analysis failed are energy consumptions of lighting, plug loads and the elevator. The sum of first order sensitivity indices for each of those is zero, suggesting that parameters that affect these three outputs were not included in the analysis. Upon closer look one can verify that this is indeed the case. Energy consumption in these cases depends almost exclusively on usage schedules (e.g. how long during the day the lights are on) rather than on any static parameters. Currently the tool supports only static parameter sensitivity analysis, so lighting, plug load and elevator schedules were not included in the analysis, and this is what the two quality measures indicated.

3.3 Demonstration of Model Calibration

Here, we describe the process involved in model calibration, using the response surface and results of sensitivity analysis described in section 3.2. The process itself was outlined

for an arbitrary model in section 2.8, and here we describe its application to a Trnsys model of the DOE benchmark building.

- The response surface $\tilde{\mathbf{f}}(x)$ of the model $\mathbf{f}(x)$ is a polynomial function of all the original 217 parameters. The Sobol indices, which are an indicator of the importance of the parameters to the measured outputs, can be used to reduce the number of parameters defining the response surface. Here, we retain only the parameters with Sobol indices $S_i > 0.05$ for any of the given outputs, which sharply reduces the number of parameters from 217 to 29.

- Next, define the response surface in terms of the reduced number of parameters, $\tilde{\mathbf{f}}(\tilde{x})$. Given the measurements of the 16 outputs c_i, define the cost function similar to that in (2.22). Use Matlab function fmincon to solve (2.22) and compute the reduced number of calibration parameters.
  In this case, since field or experimental data was not available, we compute this as follows. For the sensitivity and uncertainty analysis using the sampling method, we evaluated the Trnsys model at multiple samples (4500 in number), and the resulting outputs define a distribution, with a mean μ_i and variance σ_i^2 that can be numerically computed. The distribution of total gas consumption, along with the mean and one standard deviations are shown in figure 3.4. We assume that the calibration data are defined as $c_i = \mu_i + \beta\sigma_i$, where $\beta = 1.5$ is an arbitrarily chosen parameter. We considered a wide range of values of β for calibration and report a representative one. However, note that the calibration data should be within the spread of the output distribution for meaningful results; for data outside of this range, the response surface is no longer accurate.

- The calibration parameters are then substituted in the full Trnsys model and the error in predicting all the outputs is computed. A comparison of the error, both before and after calibration, is shown in figure 3.5. The calibration data, along with the model predictions (before and after calibration) are shown in figure 3.6.

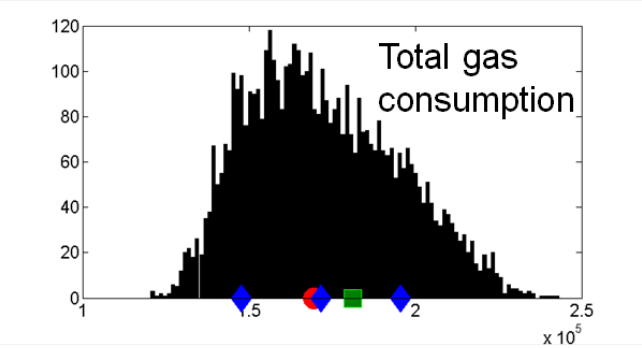

Figure 3.4: Histogram of total gas consumption, obtained from the 4500 samples used for sensitivity analysis. Also shown are the calibration data (green, square), the nominal output (red, circle), and the mean and single standard deviations (blue, diamond)

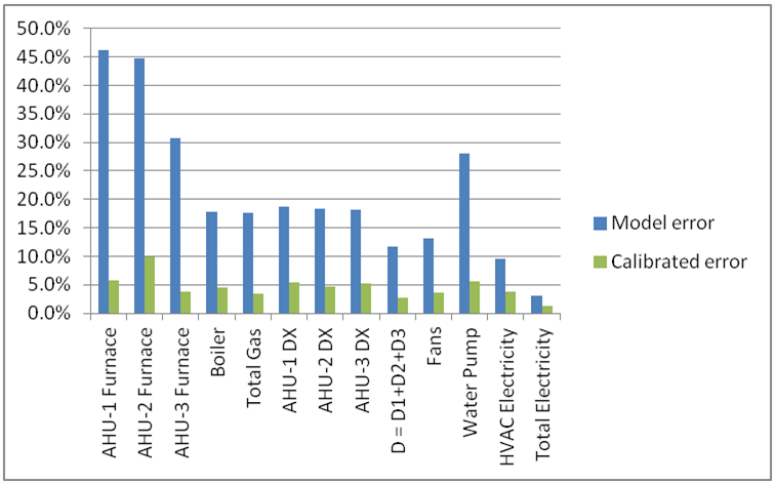

Figure 3.5: DOE Energyplus benchmark model calibration: error in predicting various outputs, before (blue) and after (green) calibration.

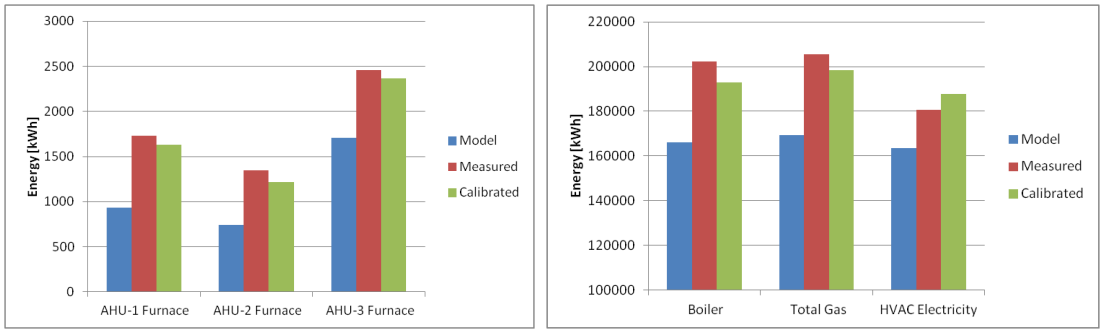

Figure 3.6: DOE Energyplus benchmark model calibration. The figure compares the model predictions of five different quantities, before (blue) and after (green) calibration, with the assumed measurements (red).

4 Next Steps

The UTRC team proposes to develop two sets of energy performance simulation use cases using existing (from year 1) or newly created TRNSYS or EnergyPlus models involving building retrofit design for:

1. Building 14 or 101 (conventional retrofit)
2. Building 661 (deep retrofit)

In collaboration with team members, existing uncertainty quantification and parametric sensitivity analysis tools will be applied to the above use cases. This will include development of rapid parameter sampling techniques, techniques to capture uncertainties associated with dynamic parameters, and standardized interfaces for automated data exchange between tools and existing energy simulation tools already in use by design practitioners. A process model for how the user would interact with tools during various stages of the retrofit design process will be developed, demonstrating the appropriate places for conducting performing rapid sensitivity and uncertainty analysis. This will be followed by a collaborative effort using the two use cases to assess the benefit, suitability and readiness of the tools within existing energy simulation frameworks and retrofit design process.

Bibliography

[1] Bryan Eisenhower, Zheng ONeill, Satish Narayanan, Vladimir A. Fonoberov, and Igor Mezic. A methodology for meta-model based optimization in building energy models. *Energy and Buildings*, (0):–, 2011.

[2] Zheng ONeill, Bryan Eisenhower, Shui Yuan, Trevor Bailey, Satish Narayanan, and Vladimir Fonoberov. Modeling and calibration of energy models for a dod building. In *Proceedings of ASHRAE Annual Conference*, Montreal, Quebec, Canada, 2011.

[3] Xiu Yang, Minseok Choi, Guang Lin, and George Em Karniadakis. Adaptive anova decomposition of stochastic incompressible and compressible flows. *Journal of Computational Physics*, 231(4):1587 – 1614, 2012.

[4] Stephen Joe and Frances Y. Kuo. Constructing sobol sequences with better two-dimensional projections. *SIAM J. Sci. Comput.*, 30:2635–2654, 2008.

[5] Ladislav Kocis and William J. Whiten. Computational investigations of low-discrepancy sequences. *ACM Trans. Math. Softw.*, 23:266–294, June 1997.

[6] Andreas Wachter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[7] Steven G. Johnson. The nlopt nonlinear-optimization package.

[8] I.M. Sobol, S. Tarantola, D. Gatelli, S.S. Kucherenko, and W. Mauntz. Estimating the approximation error when fixing unessential factors in global sensitivity analysis. *Reliability Engineering & System Safety*, 92(7):957 – 960, 2007.

[9] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, third edition, 1999.