University of California at Berkeley

Uncertainty Quantification and Finite Element Method: uqFEM V1.0.0

Frank McKenna & Nikhil Padhye

University of California at Berkeley,

Berkeley, California 94720-1710

e-mail:fmckenna@ce.berkeley.edu

SimCenter NHERI
Center for Computational Modeling and Simulation

Abstract

The uqFEM (uncertainty quantification based finite element method) is an open source software tool that allows engineers to perform reliability based structural analysis pertaining to natural hazards. It has been developed at SimCenter (https://simcenter.designsafe-ci.org/) at University of California, Berkeley. The SimCenter is part of Natural Hazards Engineering Research Infrastructure (NHERI) program, funded by the National Science Foundation. This documentation summarizes the usability and capabilities of the first version of this software tool (which we call uqFEM V1.0.0). Major enhancements are expected to be included in future releases of uqFEM. The integration of uncertainty quantification, optimization and finite element methods are expected to facilitate novel research and development efforts at intersection of these topics for variety of natural hazards and other applications.

Licenses and Copyright Notices

Uncertainty Quantification and Finite Element Method: uqFEM V1.0.0 [*]

Contents

[*]Frank McKenna is the chief developer for uqFEM V1.0.0.

1 Introduction to uqFEM

Finite element analysis under uncertainty is an important task in context to earthquake analysis of structures. An integrated tool that can carry out several tasks related to finite element simulations, uncertainty quantification and parameter estimation using optimization is now integrated in form of a package called uqFEM.

uqFEM is developed in QtCreator [1] and utilizes Python scripts to call different applications such as OpenSees (open source finite element software OpenSees, `http://opensees.berkeley.edu`), Dakota (open source repository for optimization and uncertainty quantification, `https://dakota.sandia.gov/`), and a free version of a popular finite element analysis program (FEAPpv, `http://projects.ce.berkeley.edu/feap/feappv/`). Users of uqFEM should independently install OpenSees, Dakota, and FEAPpv. Instructions for doing these are provided below.

What is OpenSees?

OpenSees (Open System for Earthquake Engineering Simulation) is a software framework for developing sequential parallel and grid-enabled finite element applications in earthquake engineering. It is written primarily in the object-oriented programming language C++. C++ wrappers are provided so that legacy and new procedures (elements, materials, numerical routines) written in other languages C, Fortran can be used [8]. The OpenSees can be downloaded and installed from `http://opensees.berkeley.edu/OpenSees/user/download.php`.

What is Dakota?

Dakota [5] is a code library that enables design exploration, model calibration, risk analysis, and quantification of margins and uncertainty with computational models. The Dakota toolkit provides a flexible, extensible interface for other simulation codes. In particular, Dakota comprises of optimization with gradient and nongradient-based methods, uncertainty quantification with sampling, reliability, stochastic expansion, and epistemic methods, parameter estimation using nonlinear least squares (deterministic) or Bayesian inference (stochastic); and sensitivity/variance analysis with design of experiments and parameter study methods. These capabilities may be used on their own or as components within advanced strategies such as hybrid optimization, surrogate-based optimization, mixed integer nonlinear programming, or optimization under uncertainty.

Dakota is a powerful system with diverse set of capabilities, and an advanced knowledge of its working is not necessary to perform analysis using uqFEM. We will refer the relevant portions of Dakota manuals, as needed. The Dakota can be installed from `https://dakota.sandia.gov/download.html`. *Users should install Dakota 6.8 for usage on personal computers.* Dakota has an option for a graphical user interface installation, but we do not recommend this option for usage with uqFEM. The Dakota website has nice tutorials and a variety of manuals aimed at beginners to power users (available at https://dakota.sandia.gov/documentation.html).

What is FEAPpv?

FEAP is a general purpose finite element analysis program which is designed for research and educational use. Source code of the full program is available for compilation using Windows (Intel compiler), LINUX or UNIX operating systems, and Mac OS X based Apple systems (GNU and Intel compilers). The FEAP program includes options for defining one, two, and three dimensional meshes, defining a wide range of linear and nonlinear solution algorithms, graphics options for displaying meshes and contouring solution values, an element library for linear and nonlinear solids, thermal elements, two and three dimensional frame (rod/beam) elements, plate and shell elements, and multiple rigid body options with joint interactions. Constitutive models include linear and finite elasticity, viscoelasticity with damage, and elasto-plasticity. A small version of the system, called FEAPpv, is available free of any charge [2].

The system may also be used in conjunction with mesh generation programs that have an option to output nodal coordinates and element connection arrays. In this case it may be necessary to write user functions to input the data generated from the mesh generation program. One can download the free executable/source for FEAPpv from http://projects.ce.berkeley.edu/feap/feappv/.

What is DesignSafe?

DesignSafe [3] is the web-based research platform of the NHERI Network that provides the computational tools needed to manage, analyze, and understand critical data for natural hazards research (see https://www.designsafe-ci.org/#research). uqFEM can run on the DesignSafe cloud. Users can create an account on the DesignSafe by registering at https://www.designsafe-ci.org/account/register/. Users must send an e-mail to *nheri-simcenter@berkeley.edu* with their User ID to seek access to DesignSafe computing resources. This allows the users to utilize computational capabilities of DesignSafe for intensive jobs. The DesignSafe cloud experiences large user loads and this can cause jobs to stay in queue for long times. See the supplement video for running uqFEM V1.0.0 on the DesignSafe cloud.

2 Installation and Execution of uqFEM

We are providing the self contained executable for uqFEM and this tool can work in conjunction with Dakota, OpenSees and FEAPpv. To execute uqFEM tool no other software installation is required if the users utilize DesignSafe cloud resources. The earlier mentioned items such as OpenSees, Dakota and FEAPpv are only needed if the users want to run uqFEM locally [1].

The source code for uqFEM can be obtained from https://github.com/NHERI-SimCenter/uqFEM.git [4]. Only advanced users are recommended to compile uqFEM application from the source directly.

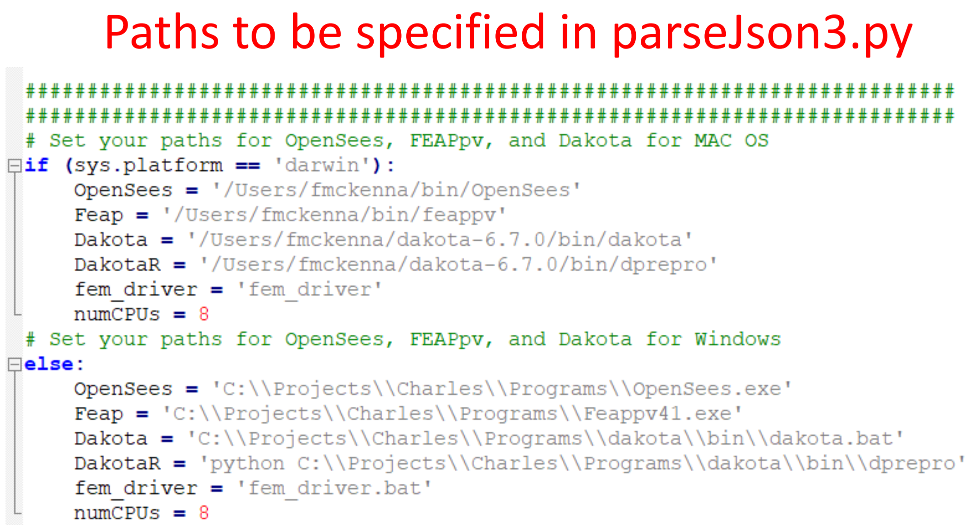
2.1 Running uqFEM V1.0.0 on DesignSafe Cloud

Users should open the uqFEM application and login to the DesignSafe cloud. The DesignSafe cloud has pre-installed applications such as Dakota, FEAPpv and OpenSees. While using DesignSafe cloud the only place where the path needs to be specified is in the sample JSON files (see next sections).

2.2 Running uqFEM V1.0.0 on local computer

As stated earlier, to run uqFEM on a local machine one needs to independently install OpenSees, Dakota and FEAPpv. Additionally, please install Python version 3.6. A particular package named NumPy is also needed to run the default post-processing scripts https://solarianprogrammer.com/2017/02/25/install-numpy-scipy-matplotlib-python-3-windows/. One can install Numpy on Windows command prompt by typing *python -m pip install numpy*.

The path to the executable for OpenSees, Dakota and FEAPpv, must be specified in Python script named parseJson.py. See Figure 1.



Paths to be specified in parseJson3.py

```
###########################################################################
###########################################################################
# Set your paths for OpenSees, FEAPpv, and Dakota for MAC OS
if (sys.platform == 'darwin'):
    OpenSees = '/Users/fmckenna/bin/OpenSees'
    Feap = '/Users/fmckenna/bin/feappv'
    Dakota = '/Users/fmckenna/dakota-6.7.0/bin/dakota'
    DakotaR = '/Users/fmckenna/dakota-6.7.0/bin/dprepro'
    fem_driver = 'fem_driver'
    numCPUs = 8
# Set your paths for OpenSees, FEAPpv, and Dakota for Windows
else:
    OpenSees = 'C:\\Projects\\Charles\\Programs\\OpenSees.exe'
    Feap = 'C:\\Projects\\Charles\\Programs\\Feappv41.exe'
    Dakota = 'C:\\Projects\\Charles\\Programs\\dakota\\bin\\dakota.bat'
    DakotaR = 'python C:\\Projects\\Charles\\Programs\\dakota\\bin\\dprepro'
    fem_driver = 'fem_driver.bat'
    numCPUs = 8
```

Figure 1: Snapshot of parseJson3.py script. The users should specify the paths for Dakota, FEAPpv and OpenSees.

[1]uqFEM V1.0.0 is unable to execute FEAPpv on a Windows local machine. However, it executes successfully on MAC OS or DesignSafe cloud.

2.3 Sample Examples provided with uqFEM V1.0.0.

Default examples can be run by loading the provided sample JSON files. These examples are self-contained, i.e., they contain the model files and input files for uqFEM V1.0.0. These are contained in the following provided folders:

1. ./exampleFEAP
2. ./exampleBayesCalibration
3. ./exampleCalibration
4. ./exampleOpenSees

The provided JSON files contain necessary information and populate the uqFEM user-interface upon loading. Typically these JSON files contain: an input script/file (specifying the details of a structural model), post-processing script (for computing a desired quantity based on the output from the finite element simulation), choice of the FEM solver, choice of uncertainty quantification or optimization method, name/type/distribution of the input variables, etc. It is also important to set the paths under "fem" label (in these JSON files) based on the current location of the respective files on a user's computer (whether running locally or on the DesignSafe cloud). If default JSON objects are not used then the user must specify all these details in the uqFEM user-interface manually. We shall demonstrate the working of these examples in the following sections. It is advisable that users get familiarized with the default examples provided and then set up their own examples.

3 Capabilities of uqFEM V1.0.0

The current capabilities of uqFEM can be summarized as shown in Figure 2. OpenSees, FEAPpv and Dakota can be downloaded and installed as stated in the earlier section. These open source softwares are already installed at DesignSafe cloud and job tasks can be submitted to the DesignSafe (new users can create an account by registering at `https://www.designsafe-ci.org/account/register/`). Currently we are limited to only use OpenSees and FEAPpv as the FEM solvers and Dakota methods for uncertainty quantification and optimization. Efforts are underway in developing a generic interface so that uqFEM can be linked with any general FEM solver, uncertainty quantification and optimization methods.

uqFEM V1.0.0, currently, can carry out sampling, parameter estimation, and Bayesian calibration on the structural models. We briefly review different algorithms for these purposes.

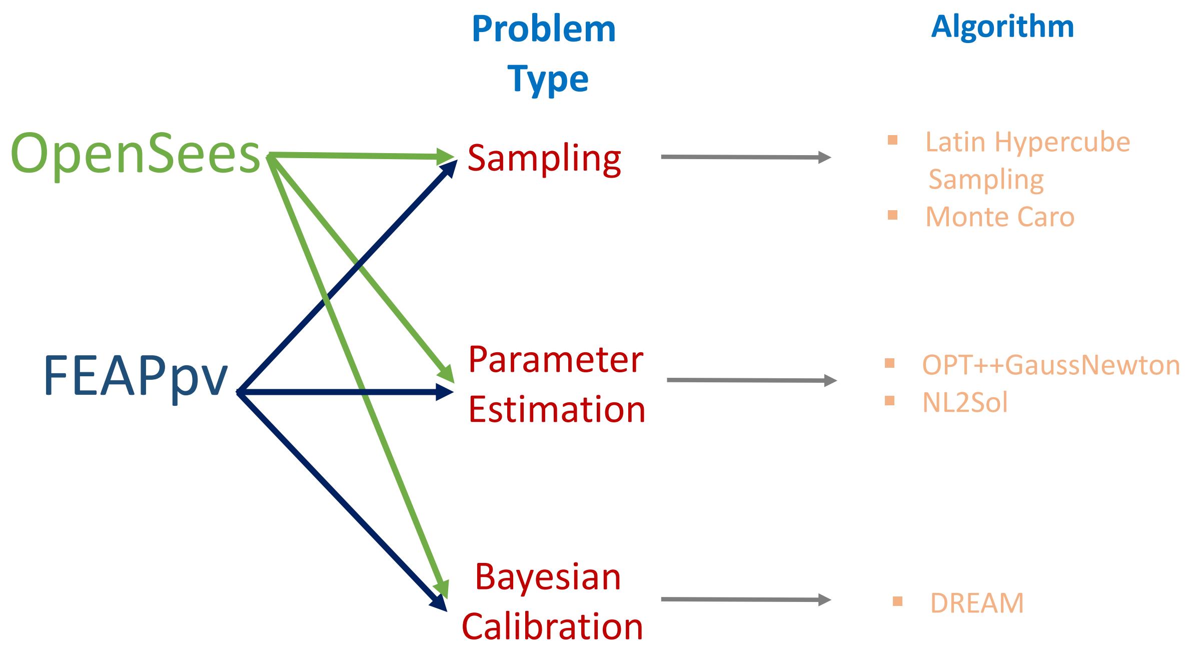


Figure 2: Available methods in uqFEM V1.0.0.

3.1 Sampling Methods:

(a) **Latin Hypercube Sampling**: Latin hypercube sampling (LHS) is a statistical method for generating a near-random sample of parameter values from a multidimensional distribution. The sampling method is often used to construct computer experiments or for Monte-Carlo integration. The method was proposed by McKay [7]. We recommend using sufficient number of samples when using this method. We demonstrate the application of this method with an example in Section 4.1. In future, UQ-FEM users will also be able to define their correlation matrices for random variables. For the non-deterministic sampling methods, users will be able to supply the correlation matrix that specifies rank correlations between the variables. See [6] for further details .

(b)**Monte Carlo:** Monte Carlo methods (or Monte Carlo experiments) are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. Their essential idea is using randomness to solve problems that might be deterministic in principle. Monte Carlo simulation furnishes the decision-maker with a range of possible outcomes and the probabilities they will occur for any choice of action.

3.2 Calibration (Parameter Estimation) Methods:

(a) **OPT++GaussNewton**: Gradient-based nonlinear programming optimizers for unconstrained, bound-constrained, and nonlinearly constrained minimization based on Gauss-Newton Method. A parameter estimation problem described in Section 4.2 is solved using this method.

(b) **NL2SOL**: Method of optimization for unconstrained and bound-constrained problems. It uses a trust-region method (and thus can be viewed as a generalization of the Levenberg-Marquardt algorithm) and adaptively chooses between two Hessian approximations, the Gauss-Newton approximation alone and the Gauss-Newton approximation plus a quasi-Newton approximation to the rest of the Hessian. Even on small-residual problems, the latter Hessian approximation can be useful when the starting guess is far from the solution. On problems that are not over-parameterized (i.e., that do not involve more optimization variables than the data support), NL2SOL usually exhibits fast convergence.

3.3 Bayesian Calibration Method:

In Bayesian calibration a “prior distribution” on a parameter is updated through a Bayesian framework involving experimental data and a likelihood function. In Bayesian methods, uncertain parameters are characterized by probability density functions. These probability density functions define the permissible parameter values - the support, as well as the relative plausibility of each permissible parameter value. In the context of calibration or any inference step, the probability density function that describes knowledge before the incorporation of data is called the prior. When data are available, the likelihood function describes how well each parameter value is supported by the data. Bayes Theorem is used for inference: to derive the plausible parameter values, based on the prior probability density and the data. The result is the posterior probability density function of the parameters. It is interpreted the same way as the prior, but includes the information derived from the data.

(a) **Dream:** This is popularly known as DiffeRential Evolution Adaptive Metropolis (DREAM) algorithm [9, 10] for Bayesian estimation. For the DREAM method, one can define the number of chains used with the chains specification. The total number of generations per chain in DREAM is the number of samples divided by the number of chains. The number of chains randomly selected to be used in the crossover each time a crossover occurs is crossover chain pairs. There is an extra adaptation during burn-in, in which DREAM estimates a distribution of crossover probabilities that favors large jumps over smaller ones in each of the chains. Normalization is required to ensure that all of the input dimensions contribute equally. In this process, a discrete number of candidate points for each crossover value is generated, which can be specified. The threshold control is the convergence tolerance for the Gelman-Rubin statistic, which governs the convergence of the multiple chain process. The integer jump step forces a long jump every jump step generations. Credible and prediction intervals can be calculated by specifying probability levels, and statistics regarding the posterior may be calculated by specifying posterior stats.

4 Running Examples using uqFEM

The snap shot of the main window upon launching the uqFEM is shown in Figure 3. Users need to provide the relevant data in different uqFEM modules. In FEM Selection module (as selected and shown in Figure 3), the users can either choose OpenSees or FEAPpv from the drop down menu. Accordingly, the path for the sample model file needs to be provided in the "Input Script" field, and a script that can post process the output from the FEM simulations needs to be specified.

The sample model file for the finite element simulation contains problem specific parameters, which we will be assigned different values during uncertainty quantification and optimization. Thus, it is important to properly set up the FEM model and associated problem parameters in the "Input Variables" section. Next, we demonstrate detailed set up and execution of uqFEM through three examples. These example templates are also provided in form of JSON files. The users are advised to run these three examples and gain familiarity with uqFEM before setting up their own problem. In order to run the provided examples users should take special care in setting the local paths correctly in the JSON files under the "fem" label. These paths should be set to the current location of the respective files on a user's computer.

Lastly, it is important to specify the path of executable files of Dakota, OpenSees, and FEAPpv in the following Python script parseJson3.py (found in /uqFEM/localApp/) if running uqFEM on a local machine.

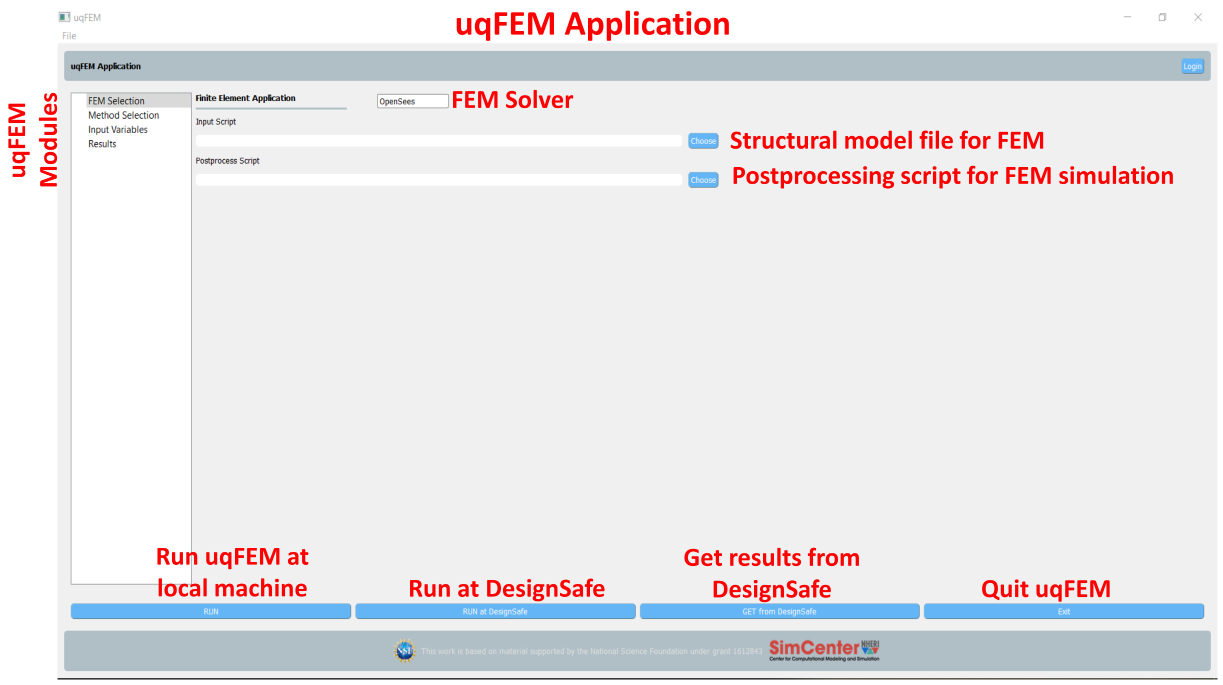


Figure 3: uqFEM V1.0.0 main window after launching the software.

4.1 Example 1: Uncertainty quantification in structural response through stochastic sampling

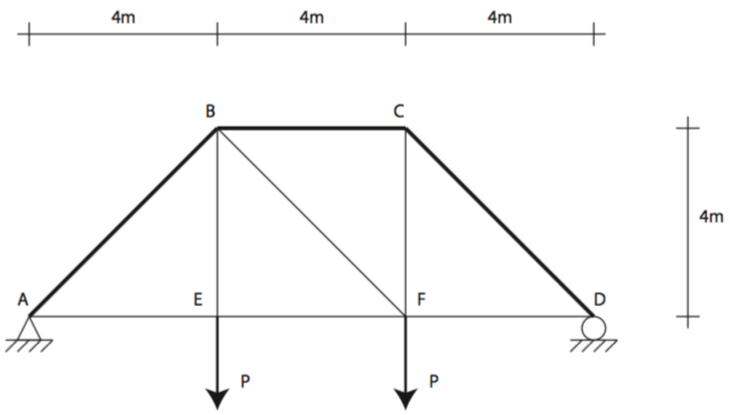


Figure 4: Structural model for uncertainty quantification.

A linear elastic truss shown in Figure 4. This structure has uncertain properties and all obeying a normal distribution:

1. Elastic modulus: $\bar{E} = 205\ kN/mm^2$, $\sigma_E = 15\ kN/mm^2$.
2. Load: $\bar{P} = 25\ kN$, $\sigma_P = 2.5\ kN$.
3. Cross-section for 3 upper bars (AB, BC and CD in Figure 4): $(\bar{A}_u) = 500\ mm^2$, $\sigma_{Au} = 25\ mm^2$.
4. Cross-section for the 6 other bars (AE, EF, FD, BE and CF in Figure 4): $\bar{A} = 250\ mm^2$, $\sigma_A = 10\ mm^2$.

We assume that the random variables are independent (i.e., zero covariance), and aim to estimate the mean and standard deviation of the vertical displacement at point F ($\bar{V}_F$ and σ_{vF}, respectively), with 95% confidence intervals for both the statistics. In this example, we use Dakota in conjunction with OpenSees. The set up example problem, populated fields and sample results are show in Figures 5, 6 and 7.

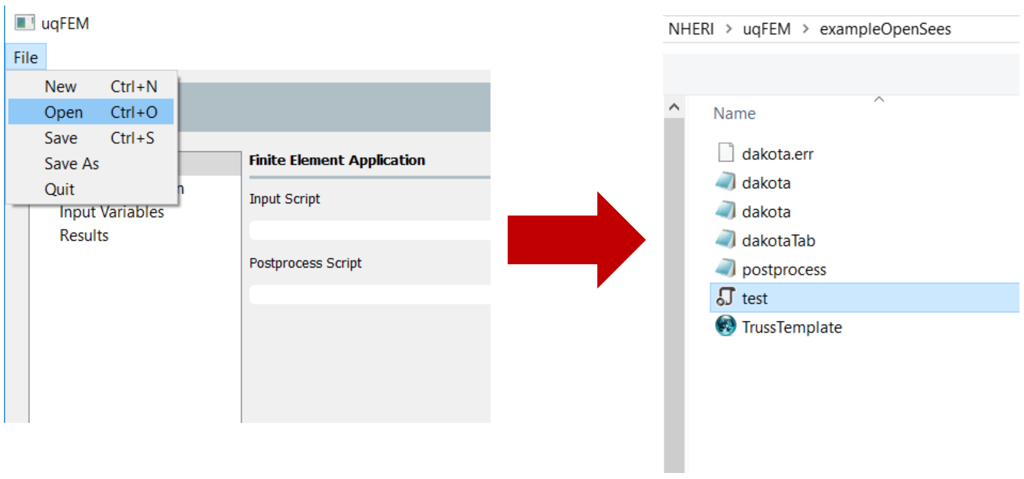
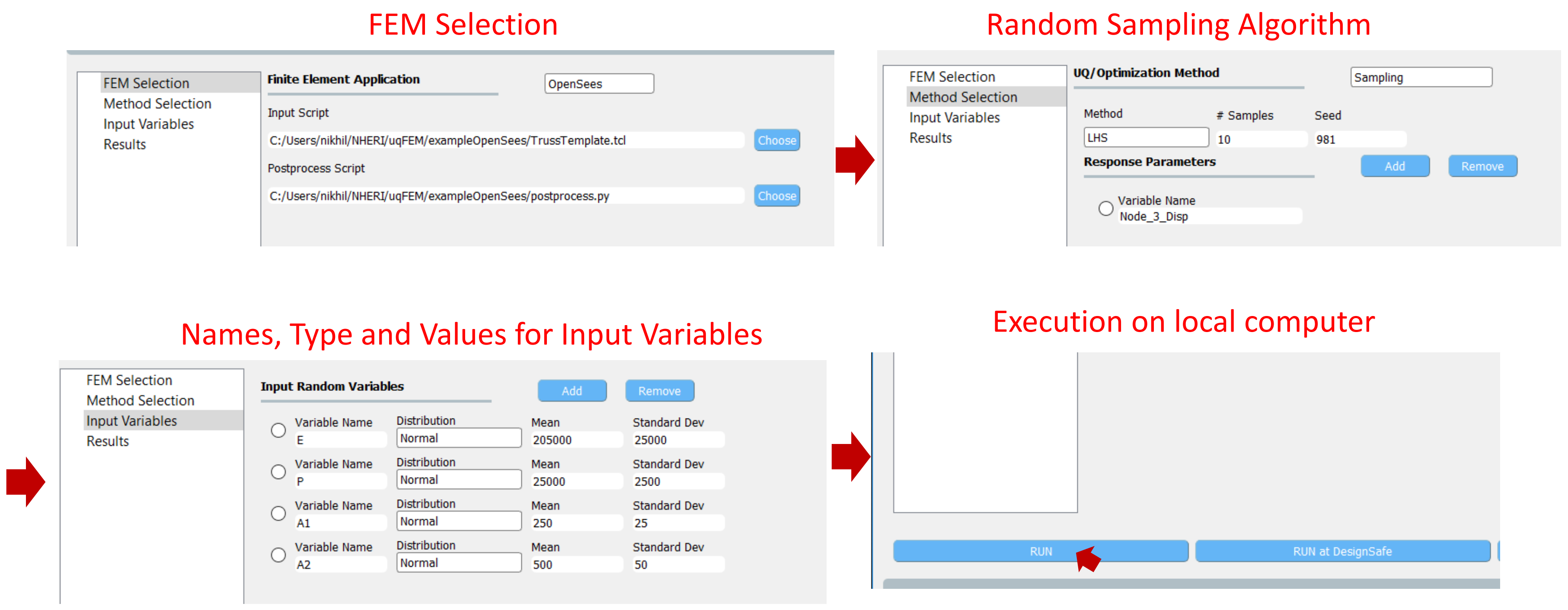


Figure 5: Loading the default sampling example by loading the JSON file “test” provided in uqFEM/exampleOpenSees for Example 1. This is done by selecting File>Open>test.json.



Finite Element Application: OpenSees

Input Script: C:/Users/nikhil/NHERI/uqFEM/exampleOpenSees/TrussTemplate.tcl

Postprocess Script: C:/Users/nikhil/NHERI/uqFEM/exampleOpenSees/postprocess.py

UQ/Optimization Method: Sampling

Method	# Samples	Seed
LHS	10	981

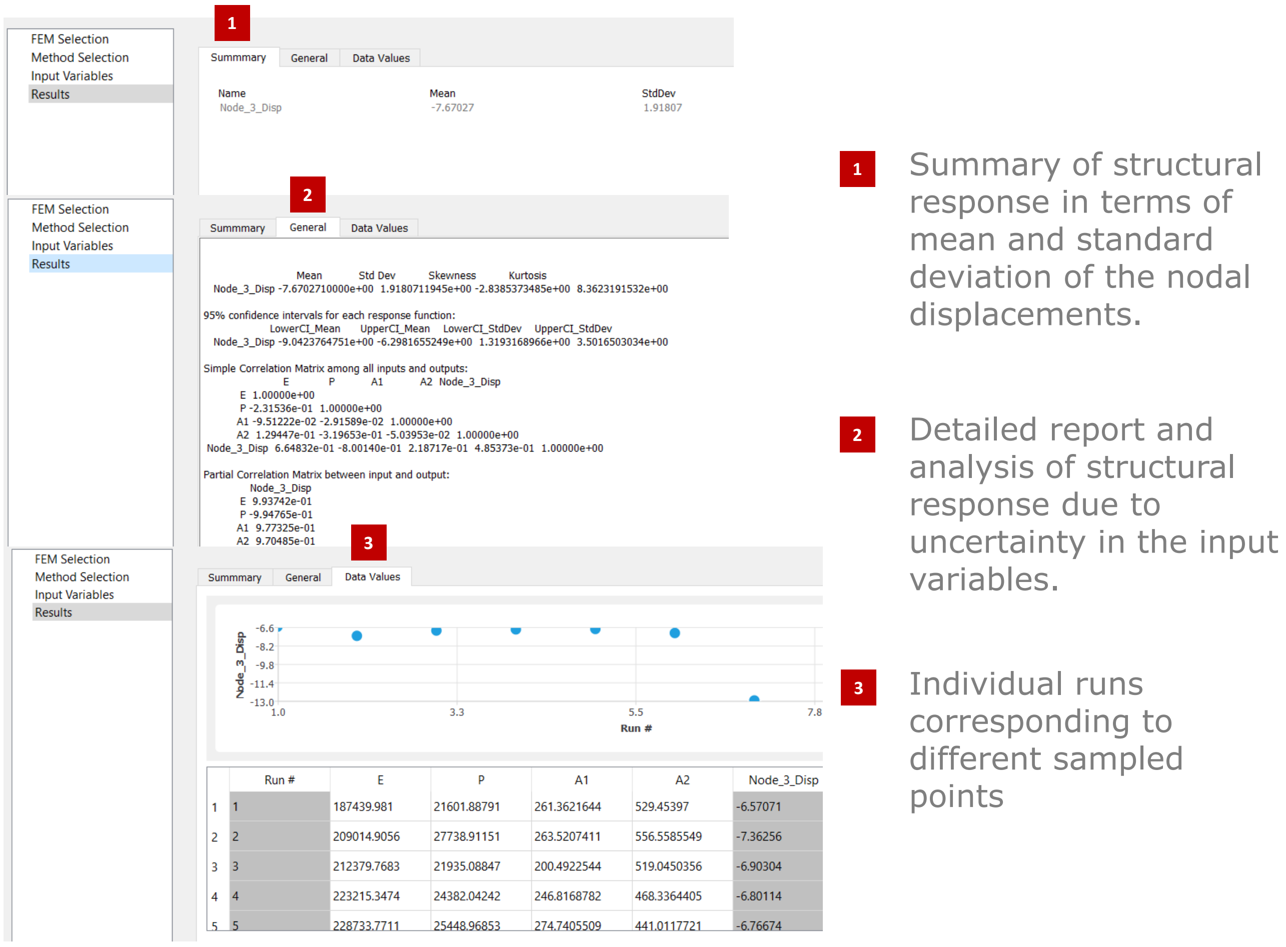
Response Parameters

Variable Name: Node_3_Disp

Input Random Variables

Variable Name	Distribution	Mean	Standard Dev
E	Normal	205000	25000
P	Normal	25000	2500
A1	Normal	250	25
A2	Normal	500	50

Figure 6: Auto-population of different modules of uqFEM based on the JSON “test” file for Example 1.



Name	Mean	StdDev
Node_3_Disp	-7.67027	1.91807

```
                      Mean          Std Dev         Skewness         Kurtosis
  Node_3_Disp -7.6702710000e+00  1.9180711945e+00 -2.8385373485e+00  8.3623191532e+00

95% confidence intervals for each response function:
                 LowerCI_Mean      UpperCI_Mean    LowerCI_StdDev    UpperCI_StdDev
  Node_3_Disp -9.0423764751e+00 -6.2981655249e+00  1.3193168966e+00  3.5016503034e+00

Simple Correlation Matrix among all inputs and outputs:
                     E            P            A1            A2  Node_3_Disp
          E  1.00000e+00
          P -2.31536e-01  1.00000e+00
         A1 -9.51222e-02 -2.91589e-02  1.00000e+00
         A2  1.29447e-01 -3.19653e-01 -5.03953e-02  1.00000e+00
 Node_3_Disp  6.64832e-01 -8.00140e-01  2.18717e-01  4.85373e-01  1.00000e+00

Partial Correlation Matrix between input and output:
             Node_3_Disp
          E  9.93742e-01
          P -9.94765e-01
         A1  9.77325e-01
         A2  9.70485e-01
```

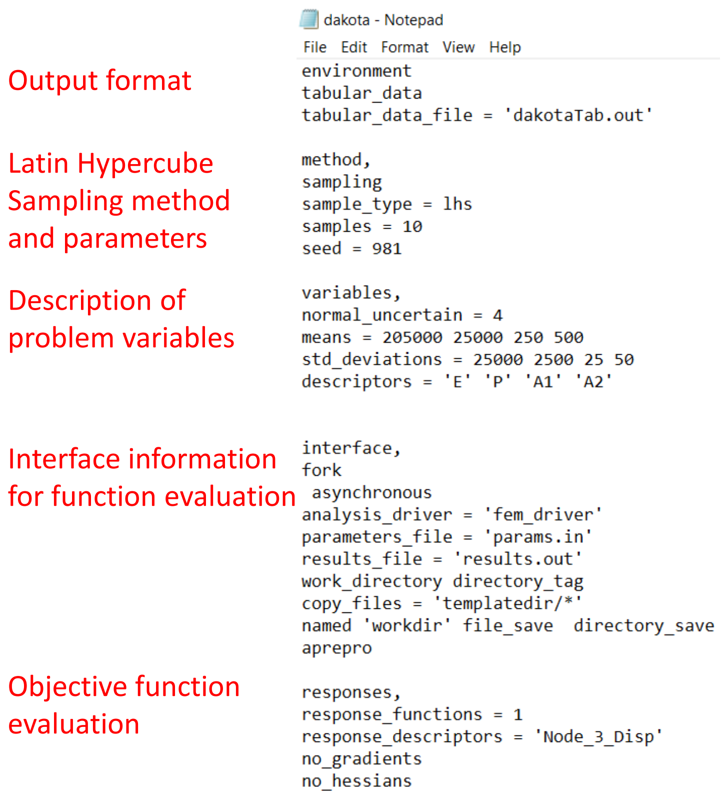
	Run #	E	P	A1	A2	Node_3_Disp
1	1	187439.981	21601.88791	261.3621644	529.45397	-6.57071
2	2	209014.9056	27738.91151	263.5207411	556.5585549	-7.36256
3	3	212379.7683	21935.08847	200.4922544	519.0450356	-6.90304
4	4	223215.3474	24382.04242	246.8168782	468.3364405	-6.80114
5	5	228733.7711	25448.96853	274.7405509	441.0117721	-6.76674

1 Summary of structural response in terms of mean and standard deviation of the nodal displacements.

2 Detailed report and analysis of structural response due to uncertainty in the input variables.

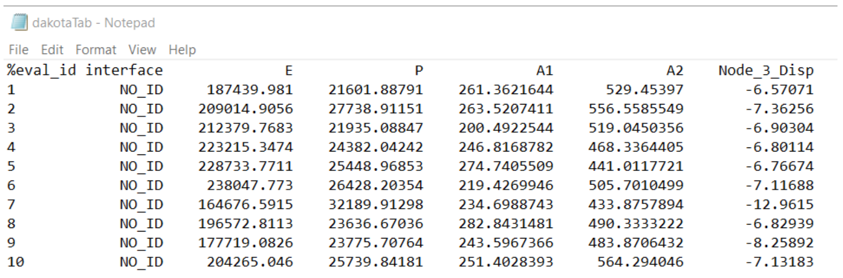
3 Individual runs corresponding to different sampled points

Figure 7: Results from stochastic structural response for Example 1. Aggregate response in terms of mean and standard deviation is provided along with detailed stochastic analysis and individual run details.


dakota - Notepad
File Edit Format View Help
Output format
Latin Hypercube Sampling method and parameters
Description of problem variables
Interface information for function evaluation
Objective function evaluation

```
environment
tabular_data
tabular_data_file = 'dakotaTab.out'

method,
sampling
sample_type = lhs
samples = 10
seed = 981

variables,
normal_uncertain = 4
means = 205000 25000 250 500
std_deviations = 25000 2500 25 50
descriptors = 'E' 'P' 'A1' 'A2'


interface,
fork
 asynchronous
analysis_driver = 'fem_driver'
parameters_file = 'params.in'
results_file = 'results.out'
work_directory directory_tag
copy_files = 'templatedir/*'
named 'workdir' file_save  directory_save
aprepro

responses,
response_functions = 1
response_descriptors = 'Node_3_Disp'
no_gradients
no_hessians
```

Figure 8: Screen shot of input file for Dakota generated by uqFEM V1.0.0. The generated file is in *correct* Dakota format.



```
%eval_id interface             E             P            A1            A2     Node_3_Disp
1             NO_ID    187439.981   21601.88791   261.3621644     529.45397       -6.57071
2             NO_ID   209014.9056   27738.91151   263.5207411   556.5585549       -7.36256
3             NO_ID   212379.7683   21935.08847   200.4922544   519.0450356       -6.90304
4             NO_ID   223215.3474   24382.04242   246.8168782   468.3364405       -6.80114
5             NO_ID   228733.7711   25448.96853   274.7405509   441.0117721       -6.76674
6             NO_ID    238047.773   26428.20354   219.4269946   505.7010499       -7.11688
7             NO_ID   164676.5915   32189.91298   234.6988743   433.8757894       -12.9615
8             NO_ID   196572.8113   23636.67036   282.8431481   490.3333222       -6.82939
9             NO_ID   177719.0826   23775.70764   243.5967366   483.8706432       -8.25892
10            NO_ID    204265.046   25739.84181   251.4028393    564.294046       -7.13183
```

Figure 9: Screen shot of data generated from running Dakota independently using the input file. The generated results match with those showed in uqFEM V1.0.0 *correctly*. This also verifies our code.

4.2 Example 2: Parameter Estimation using Optimization

This problem is concerned with parameter estimation for a three story frame subject to harmonic ground motion using OpenSees. The model consists of two parameters: damping ratio (in the range [0.01, 0.05]) and factor for the weight to the roof of the structure (in the range [0.5, 1.0]). Using some specific values of these two parameters the time-history response of the structure is computed using OpenSees. Then, the chosen parameters are treated as unknown for a parameter estimation exercise and Dakota is employed to carry out this task. The goal is to propose an error mismatch problem and identify the true parameters so as to minimize this error. The workflow is shown in Figure 10.

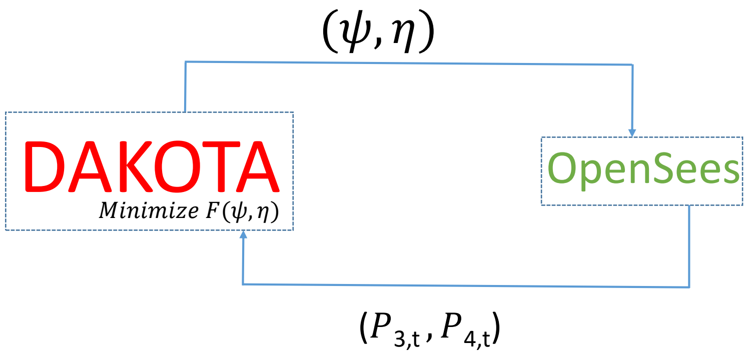


Figure 10: Calibration of parameters using Dakota-based optimization.

The original data is referred to as measured (and denoted by M), and the data generated through OpenSees software, for a particular combination of damping ratio (ψ) and roof factor (η), is referred to as predicted (and denoted by P). Predicted data is a function of ψ and η, i.e., $P(\psi, \eta)$. In particular, we use the following notation:

$P_{3,t}$: predicted value of displacement (using OpenSees) for floor 3 at time t;

$P_{4,t}$: predicted value of displacement (using OpenSees) for floor 4 at time t;

$M_{3,t}$: measured value of displacement (shown to professor) for floor 3 at time t;

$M_{4,t}$: measured value of displacement (shown to professor) for floor 3 at time t;

In order to find the correct values of ψ and η, we propose to solve an error minimization problem, such that the error mismatch between the predicted and observed values is minimal. This minimization is carried out by treating ψ and η as problem variables. The bounds for ψ and η are chosen as

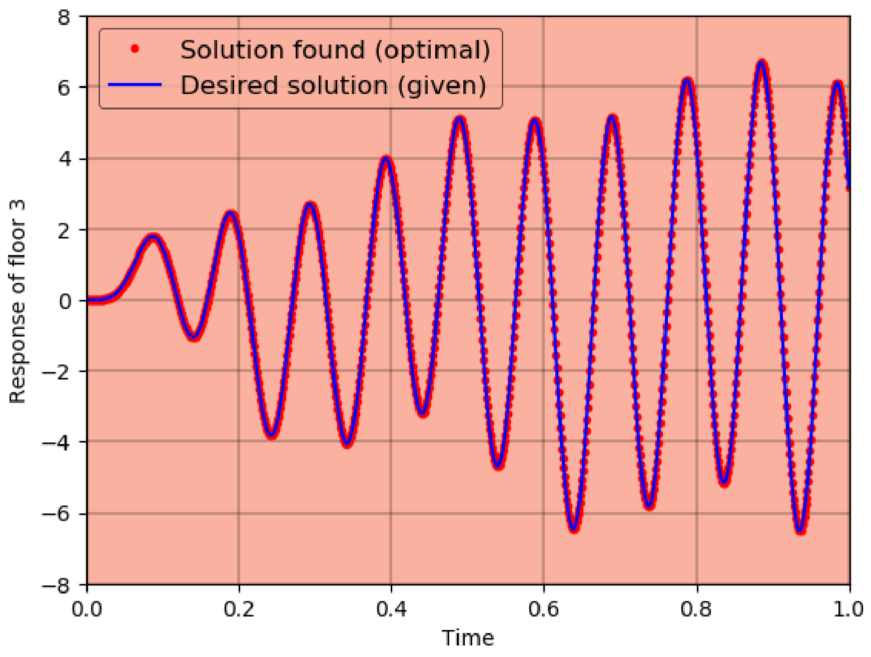


Figure 11: Comparison of response for floor 3 between optimal solution with $(\psi, \eta) = (0.0349, 0.7749)$.

[0.01,0.05] and [0.5,1.0], respectively. Formally, we write:

$$\begin{aligned}
&\underset{\psi,\eta}{\text{minimize}} && F(\psi,\eta) = \sum_t (P_{3,t} - M_{3,t})^2 + (P_{4,t} - M_{4,t})^2 \\
&\text{subject to:} \\
&\psi \in [0.01, 0.05] \\
&\eta \in [0.5, 1.0]
\end{aligned} \tag{1}$$

We use Dakota (with OPT++GausNewton) to carry out the optimization. Figures 13, 14 and 15 show the results after running the uqFEM. Finally, figures 11 and 12 show comparison of response for floor 3 and 4, respectively, of the optimal solution and reference solution (provided). We have precisely matched the desired response.

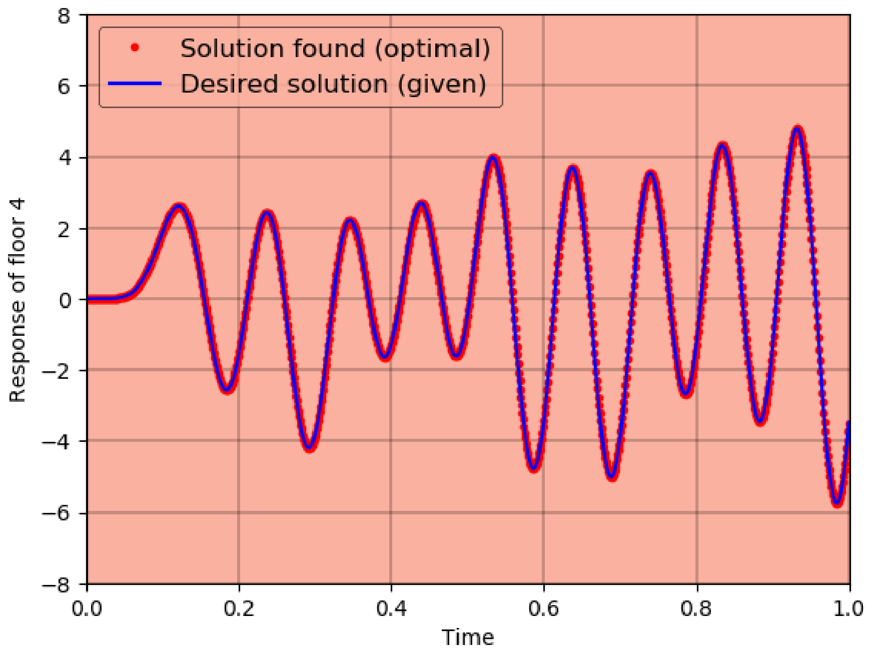


Figure 12: Comparison of response for floor 4 between optimal solution with $(\psi, \eta) = (0.0349, 0.7749)$.

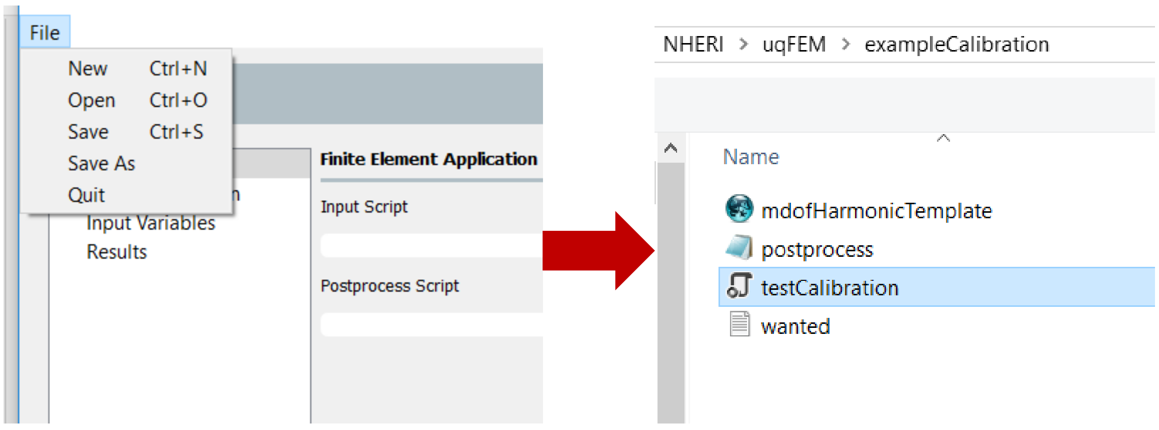
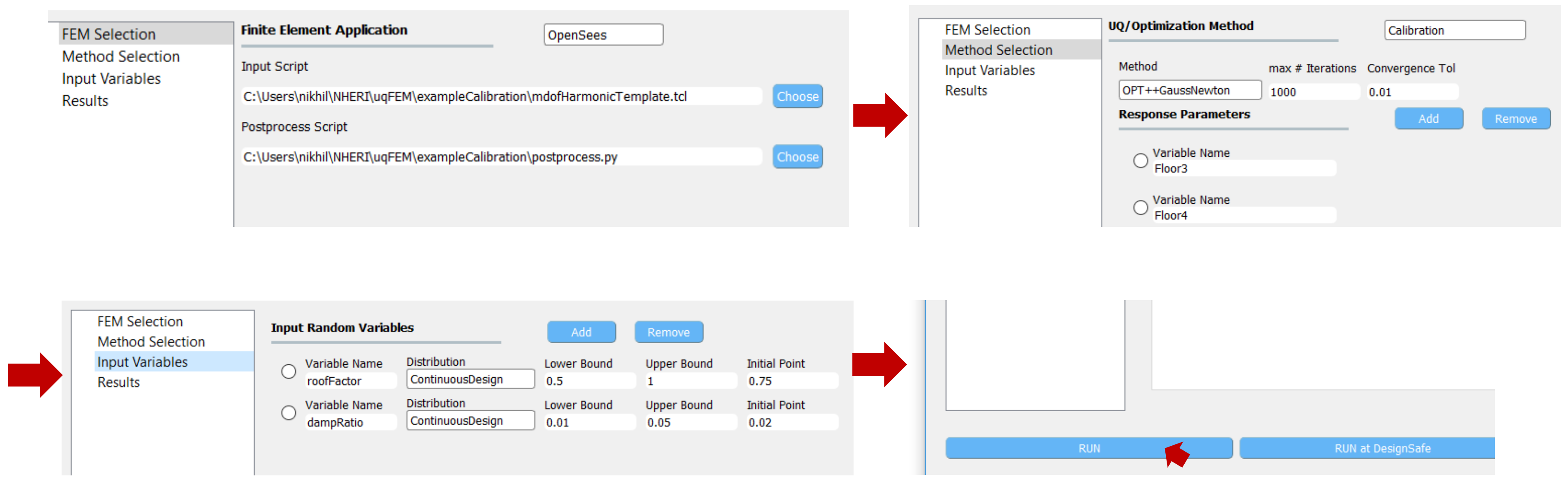


Figure 13: Loading the default calibration (parameter estimation) example by opening the JSON file "testCalibration" provided in uqFEM/exampleOpenSees for Example 2. This is done by selecting File>Open>testCalibration.json.



Finite Element Application: OpenSees

Input Script: C:\Users\nikhil\NHERI\uqFEM\exampleCalibration\mdofHarmonicTemplate.tcl

Postprocess Script: C:\Users\nikhil\NHERI\uqFEM\exampleCalibration\postprocess.py

UQ/Optimization Method: Calibration

Method	max # Iterations	Convergence Tol
OPT++GaussNewton	1000	0.01

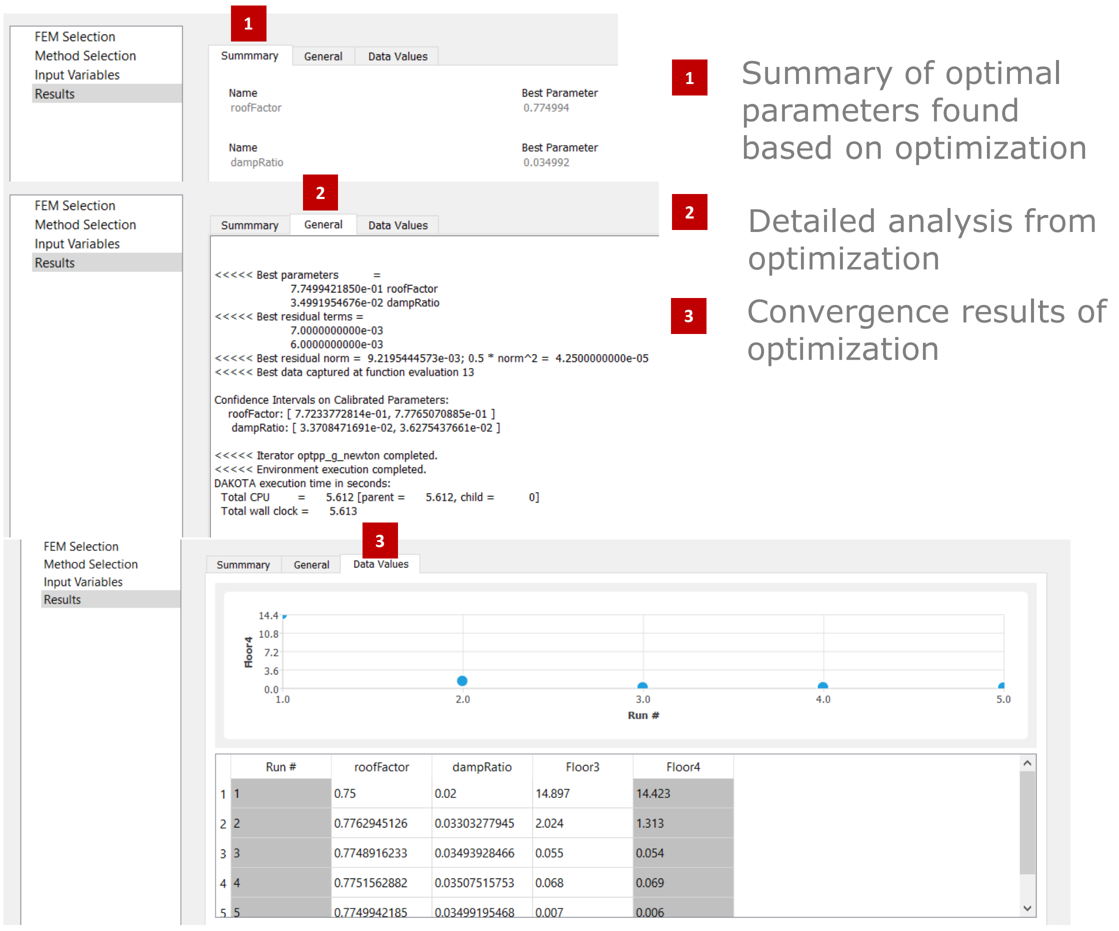
Response Parameters

Variable Name
Floor3
Floor4

Input Random Variables

Variable Name	Distribution	Lower Bound	Upper Bound	Initial Point
roofFactor	ContinuousDesign	0.5	1	0.75
dampRatio	ContinuousDesign	0.01	0.05	0.02

Figure 14: Auto-population of different modules of uqFEM based on the JSON "testCalibration" file for Example 2.


FEM Selection
Method Selection
Input Variables
Results
1
Summmary
General
Data Values
2
3
Floor4
Run #
1 Summary of optimal parameters found based on optimization
2 Detailed analysis from optimization
3 Convergence results of optimization

Name	Best Parameter
roofFactor	0.774994
dampRatio	0.034992

```
<<<<< Best parameters          =
                      7.7499421850e-01 roofFactor
                      3.4991954676e-02 dampRatio
<<<<< Best residual terms =
                      7.0000000000e-03
                      6.0000000000e-03
<<<<< Best residual norm =  9.2195444573e-03; 0.5 * norm^2 =  4.2500000000e-05
<<<<< Best data captured at function evaluation 13

Confidence Intervals on Calibrated Parameters:
    roofFactor: [ 7.7233772814e-01, 7.7765070885e-01 ]
     dampRatio: [ 3.3708471691e-02, 3.6275437661e-02 ]

<<<<< Iterator optpp_g_newton completed.
<<<<< Environment execution completed.
DAKOTA execution time in seconds:
  Total CPU        =      5.612 [parent =      5.612, child =          0]
  Total wall clock =      5.613
```

	Run #	roofFactor	dampRatio	Floor3	Floor4
1	1	0.75	0.02	14.897	14.423
2	2	0.7762945126	0.03303277945	2.024	1.313
3	3	0.7748916233	0.03493928466	0.055	0.054
4	4	0.7751562882	0.03507515753	0.068	0.069
5	5	0.7749942185	0.03499195468	0.007	0.006

Figure 15: Results from error minimization problem for parameter estimation in Example 2. Parameter vales along with detailed stochastic analysis and individual optimization run details are provided.

Online Introductory Resources

- MIT OpenCourseWare on Monte Carlo Simulation: https://www.youtube.com/watch?v=OgO1gpXSUzU
- MIT OpenCourseWare on Confidence Interval: https://www.youtube.com/watch?v=rUxP7TM8-wo
- MIT OpenCourseWare on Sampling and Standard Error: https://www.youtube.com/watch?v=soZv_KKax3E
- Latin Hypercube Sampling: https://www.youtube.com/watch?v=HORZ1uezuuw
- Mote Carlo Algorithm: https://www.youtube.com/watch?v=mATm9eCtJaQ
- A begginer's Guide to Monte Carlo Markov Chain MCMC Analysis 2016: https://www.youtube.com/watch?v=vTUwEu53uzs

References

[1] https://www.qt.io/download.

[2] http://projects.ce.berkeley.edu/feap/feappv/.

[3] https://www.designsafe-ci.org/.

[4] Code for nheri simcenter uqFEM application: A desktop application to add uncertainty quantification and optimization routines to finite element applications. https://github.com/NHERI-SimCenter/uqFEM.

[5] Brian M. Adams and et al. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.7 user's manual. https://dakota.sandia.gov/sites/default/files/docs/6.7/Users-6.7.0.pdf, 2017.

[6] Ronald L Iman and William-Jay Conover. A distribution-free approach to inducing rank correlation among input variables. *Communications in Statistics-Simulation and Computation*, 11(3):311–334, 1982.

[7] Michael D McKay, Richard J Beckman, and William J Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[8] Frank McKenna. Introduction to opensees and tcl. http://opensees.berkeley.edu/OpenSees/workshops/parallel/IntroductionOpenSees.pdf.

[9] Cajo JF Ter Braak. A markov chain monte carlo version of the genetic algorithm differential evolution: easy bayesian computing for real parameter spaces. *Statistics and Computing*, 16(3):239–249, 2006.

[10] Jasper A Vrugt, Cajo JF Ter Braak, Hoshin V Gupta, and Bruce A Robinson. Equifinality of formal (dream) and informal (glue) bayesian approaches in hydrologic modeling? *Stochastic environmental research and risk assessment*, 23(7):1011–1026, 2009.