PSUADE Short Manual
(Version 2)

Charles Tong
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94551-0808
September, 2022

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344.

Contents

1 Introduction **4**
- 1.1 A Quick Start . . . 4
- 1.2 PSUADE Capabilities . . . 5

2 Installation **7**
- 2.1 Linux . . . 7
- 2.2 MacOSX . . . 8
- 2.3 Windows (as of 2017) . . . 8

3 Using PSUADE **9**
- 3.1 Batch Mode . . . 9
  - 3.1.1 The Input Section . . . 10
  - 3.1.2 The Output Section . . . 11
  - 3.1.3 The Method Section . . . 11
  - 3.1.4 The Application Section . . . 12
  - 3.1.5 The Analysis Section . . . 14
- 3.2 Command Line Mode . . . 16
- 3.3 Calling to the PSUADE Library . . . 18
- 3.4 PSUADE-Generated Response Surface Functions . . . 18
- 3.5 PSUADE as a Response-Surface Server . . . 19

4 Examples **19**
- 4.1 Uncertainty Analysis . . . 20
- 4.2 Dimension Reduction Methods . . . 22
  - 4.2.1 Data-Centric Dimension Reduction Methods . . . 23
  - 4.2.2 Dimension Reduction Methods for Near-Linear Models . . . 24
  - 4.2.3 Dimension Reduction for Non-Linear Models . . . 25
- 4.3 Response Surface Analysis . . . 27
  - 4.3.1 Basic Steps in Response Surface Analysis . . . 27
  - 4.3.2 Suggestions on Response Surface Analysis . . . 32
  - 4.3.3 Specialized Response Surfaces . . . 32
- 4.4 Response Surface-Based Uncertainty Analysis . . . 33
- 4.5 Quantitative Sensitivity Analysis . . . 34
  - 4.5.1 Main Effect Analysis Using A Large Raw Sample . . . 35
  - 4.5.2 Main Effect Analysis Using Replicated Latin Hypercube . . . 36
  - 4.5.3 Sensitivity Analysis Using the FAST Method . . . 37
  - 4.5.4 Main Effect Analysis Using RLH on Response Surfaces (RS) . . . 38
  - 4.5.5 Main Effect Analysis Using Direct Numerical Integration on RS . . . 39
  - 4.5.6 Higher-order Global Sensitivity Analysis . . . 40
  - 4.5.7 Global Sensitivity Analysis for Correlated Inputs . . . 40
- 4.6 Mixed Aleatory-Epistemic Uncertainty Analysis . . . 41

4.7 Statistical Inference . . . 42
4.7.1 Statistical Inference Essentials . . . 43
4.7.2 Discrepancy Modeling . . . 44
4.7.3 An PSUADE Example . . . 45
4.7.4 Another PSUADE Example . . . 48
4.7.5 More About MCMC . . . 49
4.8 Two-Stage Uncertainty and Sensitivity Analysis . . . 49
4.8.1 Stage 1: Statistical Inference . . . 49
4.8.2 Stage 2: Uncertainty Propagation and Sensitivity Analysis . . . 51
4.9 Numerical Optimization . . . 53
4.9.1 Optimization with Continuous Variables . . . 53
4.9.2 Optimization With Mixed Types of Variables . . . 55
4.9.3 Optimization With Linear Constraints . . . 56
4.9.4 Optimization With Inequality Constraints . . . 57
4.9.5 Other Optimization Algorithms . . . 57
4.9.6 Response Surface-based Optimization . . . 57
4.9.7 Two-Level Optimization . . . 58
4.9.8 Optimization Under Uncertainty (OUU) . . . 58
4.10 Experimental Design . . . 61
4.10.1 Optimal Design of Experiments for Simulation Campaigns . . . 61
4.10.2 Optimal Design for Experimental Campaigns . . . 65
4.11 Miscellaneous Capabilities . . . 66
4.11.1 Sobol' Analysis for Correlated Inputs . . . 66
4.11.2 Search for Optimally Space-filling Designs . . . 68
4.11.3 Sample Quality Check . . . 68
4.11.4 Use of Response Surface Index File . . . 68
4.11.5 Adaptive Sampling Refinement . . . 69
4.11.6 Reliability Analysis . . . 69
4.11.7 Python Plotting Tools . . . 70
4.11.8 Stand-alone Response Surface Interpolators . . . 71
4.11.9 The Use of Integer Variables . . . 71
4.11.10 Sample Editing . . . 71
4.11.11 Use of Configuration Table . . . 72
4.11.12 High-Dimensional Input Reduction . . . 72

5 UQ Methodology **72**
5.1 UQ Objectives and Problem Specification . . . 73
5.2 Identification of Uncertainty Sources . . . 75
5.3 Characterization of Sources of Uncertainty . . . 76
5.4 Propagation of Uncertainties . . . 77
5.5 Analysis of Uncertainties . . . 80
5.6 Documentation/Reporting . . . 80
5.7 Reduction of Uncertainties . . . 80

6 Summary **81**

1 Introduction

PSUADE (Problem Solving environment for Uncertainty Analysis and Design Exploration) is a software package for performing various uncertainty quantification (UQ) computational tasks such as uncertainty analysis (UA), sensitivity analysis (SA), parameter study, high-dimensional reduction, response surface analysis, numerical optimization, statistical inference, optimal experimental design, etc. It comprises three major components: a suite of sampling methods, a job execution environment, and a collection of analysis/optimization tools. This document describes how to set up and use these UQ tools. Detailed UQ mathematics can be found in the theory manual.

1.1 A Quick Start

Follow the instructions in this section and you should be able to build and run PSUADE (on a simple example) in a few minutes on a Linux-based system (depending on the speed of your hardware). For building PSUADE executables on other platforms (MAC, Windows), please refer to detailed instructions in later sections.

1. Download the PSUADE source code (e.g., PSUADE_v2.X.tar.gz)
2. tar xvfz PSUADE_v2.X.tar.gz
3. cd PSUADE_v2.X
4. mkdir build
5. cd build
6. ccmake .. (Select packages by typing 'c'. Then use the arrow keys to move up and down the list, and type 'enter' to select or edit. To install the executable somewhere else other than 'build/bin', set the install directory accordingly. Also, ccmake automatically searches and set the default compilers. To select different compilers, type 't' and change the corresponding fields. Afterward, type 'c' (may be twice) and then 'g' to save and exit. (NOTE: Version 2.X also utilizes 'OpenMP' for some of its computationally expensive analysis such as cross validation for most response surface types, bootstrapped analysis such as quantitative sensitivity analysis, and multi-domain response surface types such as multi-domain RBF. If 'OpenMP' is available on your computer system, it can be utilized by typing 't' to go to the advanced mode and adding '-fopenmp -DPSUADE_OMP' to the 'CXX_FLAGS').
7. make (to create the 'psuade' executable and libraries in the buiid/bin directory), or
8. make install (to install PSUADE in the install directory other than 'build/bin'.)

9. To test your installation, do:

   (a) cd PSUADE_v2.X/Examples/SimpleUQ/Bungee/Basic

   (b) cc -o simulator simulator.c -lm

   (c) ../../../build/bin/psuade psuade.in (this is to verify that the executable runs correctly).

What you have just done are to build the PSUADE executable and perform uncertainty analysis on a simple example. At the end PSUADE prints out some summary statistics; and all sample input and output data will have been stored in the 'psuadeData' file (the default sample file produced by PSUADE). Later on in this document more details about how to create PSUADE input files ('psuade.in' in this case) and how to create Matlab/Scilab graphics will be given.

1.2 PSUADE Capabilities

PSUADE supports non-intrusive uncertainty quantification through sampling (it does have a few features to support semi-intrusive methods such as derivative-based methods). Some of the available sampling methods are:

- Monte Carlo (MC) and quasi-Monte Carlo (LPTAU),
- Latin hypercube (LH) and orthogonal arrays (OA, OALH),
- Morris one-at-a-time (MOAT), its variants, and other parameter screening designs,
- Central composite designs (CCI4, CCI5, etc),
- Factorial (FACT) and fractional factorial (FF4, FF5),
- Fourier Amplitude Sampling Test (FAST),
- Sparse grid,
- Other space-filling designs (e.g. METIS),
- Support for several standard input probability distributions (used only with MC sampling), and
- Some capabilities for adaptive sampling.

Once the sample points have been generated, they are evaluated by running the user-supplied simulation codes. PSUADE provides a simple mechanism to accomplish this via a runtime environment, which performs the following tasks when invoked (by, e.g. 'psuade psuade.in'): for each sample point,

- Write the values of a sample point to a parameter file (number of inputs in the first line, and input values in subsequent lines).
- Call the user simulation code (provided by users in the PSUADE input file - keyword is driver) with the parameter file as its first argument (and the name of the output file as its second argument).
- The user simulation code is expected to read in the parameter values from the parameter file, substitute the values in his application's input decks, run the simulation, extract target output quantities from simulation output files, and write them to the output file specified as the second argument when PSUADE calls the user program. Thus, the user program can be a simple program such as the ones compiled from simulator.c in some of the examples, or a complex super-script performing preprocessing (inserting parameter values into the simulation), model evaluation (submitting and running simulation on the user-designated system), and postprocessing (extracting and writing simulation outputs).
- PSUADE detects the presence of the output file and reads in the simulation outputs of interest.
- PSUADE then continues with the next sample point until all sample points have been processed (Optionally, PSUADE can process multiple sample points at the same time using 'asynchronous' or 'ensemble' mode).
- PSUADE then takes in all sample data (both inputs and outputs) and stores them in the PSUADE_IO section in the 'psuadeData' file.
- Finally, PSUADE analyzes the sample based on user requests, if any, given in the PSUADE input file (the ANALYSYS section), e.g. uncertainty or sensitivity analysis.

PSUADE supports many types of analysis such as

- Parameter screening (several of these methods have been tailored for different types of simulation model and different scenarios),
- Response surface construction and validation (including adaptive response surface methods),
- Basic uncertainty and correlation analysis (for raw samples or response surfaces),
- Mixed aleatory-epistemic uncertainty analysis,
- Global sensitivity analysis (first-order, second-order, and total-order Sobol' sensitivities),
- Statistical inference (with discrepancy modeling option)

- Hypothesis testing,
- Deterministic numerical optimization (for various types such as with bound or inequality constraints, with derivatives, or with mixed continuous-integer variables),
- Optimization under uncertainty (for various types),
- Graphical analysis (e.g. scatter plots via Matlab/Scilab), and
- Optimal experimental (e.g. D-, A-, G-optimal, etc.) design.

In addition, in version 2.X, parallel processing (shared memory and distributed) has been added to some analysis methods (such as bootstrapped Sobol' analysis, some response surface methods, and Bayesian inference) for faster turnaround time. Finally, there are other advanced features in PSUADE which are under active research and are not described in this document.

2 Installation

In this section we describe installation procedures for three different operating systems.

2.1 Linux

As described in the last section, installation of PSUADE on Linux-based systems is straightforward. After 'unzipping' and 'untarring' the downloaded file, go into the PSUADE directory and do the following:

```
[Linux] mkdir build
[Linux] cd build
[Linux] (optional) setenv FC <your preferred Fortan compiler>
[Linux] ccmake ..
    hit 'c'
    If you need to change the compiler, hit 't' and find the
       CMAKE_C_COMPILER and CXX fields and modify them.
    hit 'c'
    hit 'c' again until you are able to hit 'g'.
    hit 'g' to generate an exit

    * If you do not have ccmake, do :
    * cmake ..
[Linux] make
```

At the end of this installation, the PSUADE executable will have been created in the 'build/bin' directory. Note that since 'BUILD_SHARED' is the default option, the executable will use the shared libraries in the build/lib directory, so it is important to keep the

libraries at the same directory and be accessible by users. For installing the libraries and executables somewhere other than the 'build/bin' directory, set the CMAKE_INSTALL_PREFIX field in ccmake and then issue 'make install instead.

2.2 MacOSX

Building PSUADE executable from source files on MacOS is similar to building it on Linux systems. The major difference is that the Mac compilers may spit out more warnings, and possibly compiler errors that prevent a successful build. A build session is given below:

Step 1: Run ccmake

```
[MacOS] mkdir build
[MacOS] cd build
[MacOS] ccmake ..
    hit 'c'
    hit 't' to go to advanced options.
      In my case ccmake has picked up cc:
        CMAKE_C_COMPILER                /usr/bin/cc
      You can keep this the same, or you can change it to, e.g.:
        CMAKE_C_COMPILER                /usr/bin/gcc

    Once all compilers have been verified, hit 'c' until you can hit 'g',
    then hit 'g' to generate and exit.
```

Step 2: Build Executable

To build the PSUADE executable, do the following in the 'build' directory:

```
[MacOS] make (for the default installation directory) or make install
```

At the end of this installation, the PSUADE executable will have been created in the 'build/bin' directory (or some user-specified installation directory).

2.3 Windows (as of 2017)

Building PSUADE executable from source files on Windows requires 'cmake', and 'mingw' (preferably including 'gfortran'). If you desire to build an installable package, you will need NSIS. The session described below was as of 2017 and it may no longer be relevant:

Step 1: Check Compilers

First make sure you have 'cmake' version 2.8 or higher installed on your system. Then,

```
Start the 'cmake-gui' program.
Select your PSUADE source tree, and where you want it to be built.
Click 'configure'.
Select MingGW make files.
Click 'Generate'.
```

Step 2: Build Executable

Open a command line window, either 'powershell' or 'cmd', then do:

```
cd builddir
c:\mingw\bin\mingw-make.exe (It should build for a while)
```

Step 3: Install

You can now install PSUADE by running

```
c:\mingw\bin\mingw-make.exe install
```

Now continue to read this manual and follow the instructions to get a simple application running.

3 Using PSUADE

PSUADE operates primarily in one of the two modes: batch or command line mode, the latter of which allows more user interaction with PSUADE. This section is concerned mostly with these two modes. Nevertheless, this section also covers other uses of PSUADE.

3.1 Batch Mode

In batch mode, PSUADE interacts with users via a few files, the first of which is the PSUADE input file (e.g. 'psuade.in' below) that is to be created and run via

```
[Linux] psuade psuade.in
```

This 'psuade.in' file should normally begin with the keyword PSUADE as the first line, followed by 5 sections (although the ANALYSIS section can be omitted), and finally with the last line having the keyword END. The formats of these 5 sections are described next (for a concrete example, read the 'psuade.in' file in the Examples/SimpleUQ/Bungee/Basic directory).

3.1.1 The Input Section

The input section allows users to specify the number of inputs, their names, their ranges, and their probability distributions. Specifically, it is enclosed in an **INPUT** block. An example is given as follows:

```
INPUT
   dimension = 4
   variable 1 X1  =   0.0   1.0
   variable 2 X2  =   0.0   1.0
   variable 3 X3  =   0.0   1.0
   variable 4 X4  =   0.0   1.0
   PDF 1 T  0.0   1.0
   PDF 2 N  0.0   1.0
   PDF 3 S  sample3 1
   PDF 4 N  0.0   1.0
   COR 2 4  0.5
   num_fixed = 1
   fixed 1 Z1 = 1.0
END
```

In this example the number of inputs is 4, their names are X1, X2, X3, and X4 (NOTE: the variable indices are 1-based), and their lower and upper bounds are 0's and 1's, respectively. Defining probability distributions (PDF) for the inputs is optional (the default is uniform U with lower and upper bounds specified in the variable declaration). If the PDF is either normal (N) or lognormal (L), the mean and standard deviation must also be provided. If the PDF is triangular (T), the mean and half-width must be provided. Other available distributions are beta (B), exponential (E), gamma (G), Weibull (W), and user-provided samples (S). This last option (S) allows probabilities to be represented by a pre-generated sample file ('sample3' above) such as a posterior sample created from statistical inferences (the integer following the sample file is the index specifying which column in the sample file is to be used for 'X3'). An example of the 'S' type sample file can be found in Examples/PDFTest ('sample1'). The 'S' option also facilitates the use of discrete variables even though PSUADE assumes all variables to be continuous (since sample points are drawn from this user-given sample). 'COR' specifies the correlation between two inputs and it is currently available only for normal distributions (that is, for correlation to work in PSUADE, both input 2 and 4 have to be of PDF type 'N'). It should be noted that even if the prescribed input PDF is other than uniform, the input ranges should be still be carefully defined, because all sample points outside the input ranges will be 'clipped' (For example, if you define 'X1' to be normal with zero mean and standard deviation equal to 1, defining the input bound to be $[0, 1]$ will constrain the sample values to be in this range only, which is not truly Gaussian).

A new addition (Version 2.X) to the INPUT section is the fixed parameters (see example above). These fixed parameters are not directly used by PSUADE (e.g. no sample values are generated for the fixed parameters), but they are passed back to the user executable

(driver) when simulations are launched. This additional feature has been found useful in some applications.

3.1.2 The Output Section

The OUTPUT section is similar to the INPUT section but simpler, in the sense that only the dimension and the names (and not ranges or PDF's) of the output variables need to be specified. An example OUTPUT section specification is:

```
OUTPUT
   dimension = 3
   variable 1 Y1
   variable 2 Y2
   variable 3 Y3
END
```

3.1.3 The Method Section

The METHOD section specifies the selected sampling method and additional information on sampling. For example,

```
METHOD
   sampling = LH
   num_samples = 600
   num_replications = 60
   num_refinements = 0
   randomize
   random_seed = 129932931
END
```

In this example, the sampling method is Latin hypercube, the sample size has been set to 600, and no refinement is used (refinement is an advanced feature for adaptive sampling and is described in detail later). When the number of replications is larger than 1, it is called replicated Latin hypercube which is useful for certain global sensitivity analysis. In this example, with 60 replications, the number of levels for the Latin hypercube samples is 600/60 = 10 (that is, 60 Latin hypercube samples of size 10 will be created). Also, the randomize flag has been turned on to tell the sampling method that random perturbation should be added to the sample. Optionally, the random number generator seed can be provided. This is useful if you would like your sampling experiments to be replicable (having the same random seed every time the experiment is repeated should give identical results.)

Some of the other sampling methods available in PSUADE are (refer to the theory manual or examine your sample data file, 'psuadeData', for more details):

```
MC    - Monte Carlo
```

```
LPTAU - a quasi-random sequence
FACT  - full factorial design
MOAT  - Morris one-at-a-time screening
LH    - Latin hypercube
OA    - Orthogonal Array
OALH  - Orthogonal Array-based Latin hypercube
FAST  - Fourier Amplitude Sampling Test (FAST)
BBD   - Box Behnken design
PBD   - Plackett Burman design
FF4   - Fractional factorial of resolution IV
FF5   - Fractional factorial of resolution V
FF5   - Fractional factorial of resolution V
CCI4  - Central composite (circumscribed) of resolution V
CCI5  - Central composite (circumscribed) of resolution V
METIS - full space-filling based on domain decomposition
SPARSEGRID - a sparse grid method
```

3.1.4 The Application Section

The application section sets up the user-provided simulation executable and other runtime features. An example is given below.

```
APPLICATION
   driver = ./testmain
   opt_driver = NONE
   aux_opt_driver = NONE
   ensemble_run_mode
   ensemble_driver = NONE
   ensemble_opt_driver = NONE
   max_parallel_jobs = 1
   max_job_wait_time = 1000000
END
```

Here driver points to the executable to be used as the simulation model. opt_driver and aux_opt_driver point to the executables (set it to NONE or not declaring opt_driver if you are not running optimization) for numerical optimization. aux_opt_driver is needed for the 'SM' and 'MM' optimizers, which are two-level optimization methods that utilize two simulation models with different fidelities). Again, the user simulation code can just be a simple program or a complex super-script performing preprocessing, actual model evaluation, and postprocessing. driver can also be a PSUADE sample data file (and the simulation model will be the response surface built from this sample), as will be shown later.

After the creation of a sample based on information from the INPUT, OUTPUT, METHOD, and APPLICATION sections, PSUADE proceeds with launching the jobs. If the keyword

max_parallel_jobs is set to 1 (or not set), the sequential mode is turned on. In this mode, PSUADE schedules the simulation runs sequentially beginning with sample point 1. To run job i, PSUADE first creates an input parameter file (called 'psuadeApps.in.i'). This file contains in its first line the input dimension, followed in subsequent lines by values of the input parameters for the i-th sample point. An example of this input file (with 2 input variables) created by PSUADE is (called 'psuadeApps.in.9' for sample 9, for example):

```
2
0.345
1.429
```

PSUADE then internally calls driver (make sure the simulation code has execute permission on) with two arguments:

```
./testmain psuadeApps.in.9 psuadeApps.out.9
```

The driver program is expected to take the input parameter values from 'psuadeApps.in.9', perform function evaluation, and write the outputs to the file specified as the second argument ('psuadeApps.out.9'). An example of the content of an output file (e.g. 3 output variables) to be created by user programs is (note that, as opposed to the input, the dimension is not needed):

```
3.12
15.9
100.4
```

If max_parallel_jobs is set to a number larger than 1, the asynchronous job scheduling mode will be turned on. In this mode, multiple 'psuadeApps.in.i' files are created, and driver is launched max_parallel_jobs times simultaneously. Since asynchronous execution uses the Linux background processing mechanism, make sure that max_parallel_jobs is set moderately (at most about $10-20$), or the system may be clogged up with too many concurrent processes. Another caution is that since multiple jobs are running concurrently, make sure they do not create temporary files with same name in the same execution directory or they may suffer from interference (or 'aliasing' in computer science terminology). Also, in the parallel mode, the keyword max_job_wait_time is used for fault detection and recovery (that is, PSUADE waits for a job to complete until the time has expired, at which point PSUADE considers this job to be a failed run and re-launches it.)

For fast simulations (such as 'simulator.c' in Bungee/Basic of the Examples/SimpleUQ directory), cycling through the sample points one by one will require file input/output (open the parameter file, read in the sample point, evaluate, and write to the output file) that may consume much more time than the actual simulations themselves. To reduce the I/O overhead, PSUADE provides the ensemble_driver (and ensemble_opt_driver for numerical optimization) that can be called in place of driver (and opt_driver) to facilitate 'group processing' when *ensemble_run_mode* is turned on. In this mode, PSUADE writes multiple sample points (as specified by the max_parallel_jobs variable) in the parameter file and the user executable is

expected to process all sample points before returning the results to PSUADE. An example is given in the Bungee/Basic directory (uncomment the ensemble lines in 'psuade.in', compile 'ensemble_simulator.c' and run 'psuade psuade.in').

Some of the other options in this section are:

```
launch_only         - launch all jobs without waiting for results
gen_inputfile_only  - generate all psuadeApps.in.$i$ files only
limited_launch_only - launch max_parallel_jobs jobs and terminate
```

launch_only is useful when you can run all the jobs at the same time on your machine but each job takes a long time to run (for example, when the jobs are launched on a backend HPC system). In this case PSUADE terminates after all jobs have been launched (that is, PSUADE does not wait for job completion), and when all the jobs are completed, PSUADE can be launched again to harvest the results.

gen_inputfile_only is useful when job scheduling is handled outside of PSUADE. In this case PSUADE is launched initially to create the input files for all sample points (that is, all 'psuadeApps.in.i'. Subsequently, users can take these input files, run simulations using their preferred job scheduler, and create all 'psuadeApps.out.i'. Finally, PSUADE can be called again to harvest the results.

limited_launch_only is useful in a semi-automated job scheduling environment (e.g. on the LC machines at LLNL where too many job submissions at once may overly burden the job scheduler). In this case PSUADE is called initially to launch a small number of jobs (e.g. 100 out of 1000 in total). The job execution scripts for these 100 jobs should be equipped with capabilities to launch more jobs when they are completed (e.g. the dependency auto-submission mechanism on the LC machines). This 'domino' job launching (also called 'chain mode') will continue until all 1000 jobs have been completed in an automated fashion. After that, PSUADE can be called again to harvest the results (as expected, this 'domino' effect cn be disrupted if some jobs are abruptly terminated before triggering the next in the chain, so be cautious).

3.1.5 The Analysis Section

The ANALYSIS section specifies the type of analysis to be performed and the analysis settings to be used. An example given below (Moment) computes summary statistics on output number 1 (output_id = 1). Lines beginning with '#' are treated by PSUADE as comment lines (the commented options will be explained later).

```
ANALYSIS
   analyzer method = Moment
   analyzer threshold = 5.000000e-04
   analyzer output_id  = 1
   #analyzer rstype = MARS
   #optimization method = bobyqa
   #optimization num_local_minima = 1
```

```
   #optimization use_response_surface
   #optimization num_fmin = 1
   #optimization fmin = 0
END
```

Some of the available analysis methods are:

```
Moment         - mean, standard deviation, skewness, kurtosis
MainEffect     - Sobol' sensitivity indices
TwoParamEffect - second order Sobol' sensitivity indices
RSFA           - response surface (RS) analysis (curve fitting)
MOAT           - Morris one-at-a-time screening analysis
Correlation    - correlation analysis
Integration    - numerical integration using the data points
FAST           - Fourier Amplitude Sampling Test analysis
FF             - fractional factorial main and interaction analyses
PCA            - principal component analysis
RSMSobol1      - RS-based first order sensitivity analysis
RSMSobol2      - RS-based first second sensitivity analysis
RSMSobolG      - RS-based group main effect analysis
RSMSobolTSI    - RS-based total sensitivity analysis
```

When response surfaces are used together with the selected analysis method, the response surface type (rstype above) may have to be specified (for some analysis methods, rstype is prompted at runtime; for others, it will be fetched from this rstype setting). Some of the available response surface types are (some of these are not included in the releases):

```
MARS            - multi-variate adaptive splines (by Friedman)
MARSBag         - MARS with bootstrapped aggregation
MMARS           - multi-domain MARS for very large samples
linear          - linear regression
quadratic       - second-order regression
cubic           - third-order regression
quartic         - fourth-order regression
legendre        - Legendre regression (different polynomial orders)
user_regression - regression with user-specified basis functions
GP1             - Gaussian process (Mackay's TPROS)
GP2             - Gaussian process (PSUADE's local implementation)
MGP3            - multi-domain Gaussian process (for large samples)
TGP             - treed-Gaussian Process by Gramacy and Lee
SVM             - support vector machine
Kriging         - an universal Kriging method
sum_of_trees    - an implementation of the sum-of-trees method
ANN             - artificial neural network
```

```
KNN             - K nearest-neighbors
RBF             - radial basis functions
MRBF            - multi-domain RBF
splines         - 2D or 3D splines (on factorial designs)
selective_regression - user-specified subset of polynomials
sparse_grid_regression
```

In addition, for performing numerical optimization, a few related options have to be specified (the commented lines from the above example). In the example, the selected optimization method is bobyqa by Michael Powell. Other available optimization methods are cobyla (optimization with inequality constraints again by Michael Powell), newuoa (optimization with unbounded constraint by Michael Powell), lincoa (optimization with equality constraints by Michael Powell), LBFGS (derivative-based optimization), OUU (optimization under uncertainty), SCE (pattern search optimization that can be used for mixed integer programming), Nomad (external library for continuous, integer, or binary variables), minpack (an external optimization package), and sm/mm (two-level space-mapping and manifold-mapping methods by David E. Ciaurri). The keyword num_local_minima tells PSUADE how many optimal or sub-optimal minima (say k) to identify (from the initial sample that should have size at least k) for multi-start searches. If user_response_surface is used, the sample data will first be used to create a response surface before searching for minima in the initial sample. num_fmin (default is 1) tells PSUADE to identify the best subset of num_local_minima sub-optimal points. Users can also tell PSUADE the optimal value to look for via fmin (for example, for least-squares optimization the obvious minimum is 0.)

3.2 Command Line Mode

PSUADE allows users to 'interactively' perform some of the analyses (e.g. statistical inference can only be performed in this mode). In this mode, all simulations should first be run in batch mode (i.e., turning off all analysis methods in the PSUADE input file). Once all simulations have been completed, the 'psuadeData' file will contain all sample inputs and the corresponding outputs enclosed in the PSUADE_IO section. This file (which needs to be renamed to prevent being overwritten) is to be loaded in the command line mode for analysis. Command line mode is launched by calling

```
[Linux] psuade
```

without any argument. There are different categories of commands. Use the help command to explore these categories:

```
Help topics:
      info          (Information about the use of PSUADE)
      io            (File read/write commands)
      stat          (Basic statistics)
      screen        (Parameter screening commands)
```

```
rs            (Response surface analysis commands)
uqsa          (Quantitative UA/SA commands)
calibration   (Statistical Calibration commands)
odoe          (Optimal experimental design)
kpca          (Commands that support KPCA-based MCMC)
plot          (Commands for creating plots)
setup         (Commands to set up PSUADE work flow)
edit          (Commands to edit resident sample data)
misc          (Miscellaneous commands)
advanced      (Advanced analysis and control commands)
<command -h> (Help for a specific command)
```

For example, to see the available commands for uncertainty and sensitivity analysis, type help uqsa, which will display the following commands:

```
Commands for RS-based uncertainty/sensitivity analysis:
(to see more uqsa commands, use 'help uqsa long')
(to see details of each command, use '-h' option)
      rsua       (RS-based UA)
      rsuab      (RS-based UA with bootstrap)
      rsmeb      (RS-based McKay main effect + bootstrapping)
      rsieb      (RS-based McKay pairwise effect + bootstrap)
      rssobol1b  (RS-based Sobol' main effect + bootstrap)
      rssobol2b  (RS-based Sobol' 2-way analysis + bootstrap)
      rssoboltsib(RS-based Sobol' total effect + bootstrap)
      aeua       (RS-based aleatoric-epistemic analysis)
      soua       (RS-based 2nd order analysis: PDF variation)
      NOTE: use 'help uqsa long' for more detailed information
```

The commands in all categories are described in more detail in the reference manual. Also, you can use the '-h' option to examine each individual commands. An example of a help menu for the rsua command is:

```
psuade> rsua -h
rsua: uncertainty analysis on response surface
Syntax: rsua (no argument needed)
This command perform uncertainty analysis on the response surface
built from the LOADED sample. Uncertainty analysis is performed
using a user-provided sample created beforehand in PSUADE data
format or a PSUADE-generated sample. If a stochastic response surface
(e.g. Kriging, MARSB, or polynomial regression) is selected, its
response surface uncertainty will also be shown in the PDF and CDF
plots produced by this command.
NOTE: Turn on master mode to select between average case and worst
      case analysis.
```

Another example: after you have completed a set of runs, a PSUADE data file will have been created (and to be renamed to 'simdata'). To create scatter plots for the data in the command line mode, do:

```
[Linux] psuade
*************************************************************
*      Welcome to PSUADE (version 2.1.0)
*************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration (2.1.0)
=============================================================
psuade> load simdata
psuade> splot
matlabsp.m is now available for scatter plots.
psuade> quit
[Linux]
```

You can now launch Matlab to display the scatter plot. You can also create Scilab files by first issuing the scilab command (Note: Scilab plotting tools are less well-developed than Matlab plotting tools within PSUADE) before running splot. There are also some Python plotting tools in the 'Python' directory (details will be given later).

The command line mode contains commands not only for statistical analysis, but also for sample manipulation and visualization (Use help to explore the rich collection of commands).

3.3 Calling to the PSUADE Library

In addition to batch and command line modes, PSUADE also has the capability to be used as a library. Specifically, PSUADE functions such as uncertainty assessment, global sensitivity analysis, and numerical optimization can be called directly from a user program when it is linked with the PSUADE shared library (The linking is straightforward if the user program is in C++, and the shared object files are located in the build/lib directory). The usage of this PSUADE functionality is described in detail in the reference manual.

3.4 PSUADE-Generated Response Surface Functions

One unique feature offered by PSUADE is a collection of response surface interpolation subroutines that can be linked directly into users' simulation codes. This feature is useful when it is desirable to add a physics sub-module directly into a simulation code but only the data characterizing the physical phenomenon and not the governing equation is available. In this case, the data will be used to train a surrogate (or response surface), which will then be used as the sub-module (possibly with prediction uncertainty) in the simulation code. PSUADE provides many such surrogates in C++ and/or Python. More details can be found in the section on response surface analysis.

3.5 PSUADE as a Response-Surface Server

Response surfaces are used everywhere within PSUADE for various purposes. A user, on the other hand, may want to use PSUADE to build a response surface that is very time-consuming to create (e.g. building a Gaussian process model with 5000 sample points); and he wants to build it once and use it many times. One solution is described in the last section (PSUADE-generated C++ code for response surface interpolation), but if that is not feasible, another solution is to use PSUADE as a response surface interpolation server. In this mode, PSUADE is launched in the command line mode, a sample file is loaded, a response surface for this sample is created using the rscreate command, and subsequently, this response surface can be evaluated repeatedly using the rseval_m command, which waits for a request from a client program via a file, evaluates the requested sample points, and returns the results to the client via another file. This protocol continues until the PSUADE session is terminated. An example is given in the Examples/PsuadeAsServer directory.

4 Examples

PSUADE provides many tools for answering UQ questions concerning the simulation model. For example, given a computational model simulating some physical processes,

1. What is the impact of parameter uncertainties on the model output of interest? (Uncertainty analysis)

2. Is there a small subset of parameters accounting for most of the output variabilities? If so, what is this subset? (Parameter screening/down-select)

3. How to construct a relationship between some input parameters and the model outputs of interest? (Response surface or surrogate modeling)

4. How much of the output uncertainties can be attributed to individual parameters? (Global sensitivity analysis)

5. What are the parameter values that give the best model performance when my simulation model has uncertainties? (Numerical optimization under uncertainty)

6. What are the parameter values that best fit the available experimental data in view of different sources of uncertainties? (Statistical inference)

7. In light of parameter uncertainties, what is the probability that the model outputs fall outside acceptable ranges? (Risk analysis)

8. What physical experiment(s) should be conducted next to minimize prediction uncertainties? (Optimal experimental design)

9. What simulation experiment(s) should be performed next to improve a response surface (or surrogate model) that mimic the computational model? (Adaptive sampling)

10. How to visualize uncertainty data? (for purposes of identifying outliers, observing trends, presenting to stakeholders, etc.)? (Uncertainty visualization)

In the following we provide a few examples to show in more detail how to set up and run PSUADE. PSUADE has many other advanced features for handling complex multi-physics models.

4.1 Uncertainty Analysis

This section shows how to perform a simple uncertainty analysis on the following Bungee function (in the Examples/SimpleUQ/Bungee/Basic directory:

$$Y = H - 2Mg/(1.5 * \sigma)$$

where the parameter ranges are set accordingly. To compute the basic statistical moments of this function assuming all inputs are uniformly distributed in their respective ranges, we select the LPtau design with a sample size of 300 (an arbitrary pick). Let's say the simulations are to be run in sequential mode (since the simulations are computationally inexpensive). The corresponding PSUADE input file (say, 'psuade.in') is:

```
PSUADE
INPUT
   dimension = 3
   variable 1 H   =   40   60
   variable 2 M   =   67   74
   variable 3 sig =   20   40
END
OUTPUT
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = LPTAU
   num_samples = 300
   randomize
   random_seed = 128121211
END
APPLICATION
   driver = ./simulator
END
ANALYSIS
   printlevel 1
END
END
```

Here randomize in the METHOD section specifies that random perturbations are to be added to the sample; and random_seed is defined so that repeating this exercise will produce the same result (since randomization and perturbation added to the Latin hypercube sample depends on the seed of the random number generator). driver points to an executable, which takes a sample point from a parameter file (first argument), uses it to run a simulation, and writes the simulation output to the output file (second argument). driver can be in any language provided that it can be executed (by the Linux system command). Our example is the following simulation program ('simulator') compiled from 'simulator.c'.

After these files have been prepared, run PSUADE:

```
[Linux] psuade psuade.in
```

and, at the completion of the runs, a result file called 'psuadeData' will have been created. Next, launch PSUADE in command line mode, load the sample, and run uncertainty analysis:

```
[Linux] psuade
psuade> load psuadeData
psuade> ua
...
```

And at the end, the following statistical measures will be displayed:

```
...
* outputID =          1
*       Sample mean          =   1.7957e+01
*       Sample std dev       =   8.6932e+00
*       Sample skewness      =  -2.2206e-01
*       Sample kurtosis      =   2.4823e+00
```

In addition, a Matlab file called 'matlabua.m' will have been created for visualizing the output distribution. This command line approach is recommended because this approach provides many other functionalities to analyze the same data set (e.g. ca for correlation analysis).

The above uncertainty analysis can also be performed in batch mode by:

1. Adding a line analyzer method = Moment in the ANALYSIS section of 'psuade.in', and
2. Running the command: 'psuade psuade.in'.

In this example, we observe that even for computing simple statistical moments, a sample size of 300 may be too small. You may increase the sample size and re-run to see if the result changes much.

PSUADE also provides an approach that iteratively add more sample points and check statistics. This is shown in the Examples/SimpleUQ/Bungee/Iterative directory. Specifically, an additonal line num_refinements = 5 can be inserted in the ANALYSIS section. This will cause PSUADE to iteratively increase the sample size and re-anaylze. Iterative refinement can be demonstrated by running

```
[Linux] psuade psuade.in
```

and tracking the statistics displayed after each refinement.

4.2 Dimension Reduction Methods

Many UQ analyses suffers from the notorious 'curse of dimensionality' (tens to hundreds or more), which often results in prohibitively high computational cost and inaccurate analysis results. As such, effective and efficient dimension reduction methods are needed to enable feasible UQ. Dimension reduction techniques can be applied to model inputs or outputs. Techniques for high-dimensional outputs are usually based on the method principal component analysis (PCA) and its variants. While PSUADE provides a simple PCA method for output dimension reduction, it provides an exotic collection of input dimension reduction methods, which are described in this section. Input dimension reduction in this context pertains to a set of high-dimensional and independent inputs. Dimension reduction for correlated inputs (e.g. the kernel PCA method) is covered in another section.

The objective of input dimension reduction in our current context is to identify the most important parameters that drive model output uncertainty (again, other input dimension reduction approaches exist, such as finding a low-dimensional manifold embedded in the high-dimensional input space). Thus, it can be considered as a type of sensitivity analysis or subset selection. The major difference in the current context is that our objective is to employ relatively inexpensive methods to quickly identify the 'subset' of the most important parameters for subsequent analysis that are more computationally intensive (e.g. response surface analysis, variance-based sensitivity analysis). As such, we are more interested in 'qualitative' sensitivity analysis (thus also called parameter down-selection or screening) rather than 'quantitative' sensitivity analysis, which apportions a higher precision real-value sensitivity index to each parameter.

There are several parameter screening methods that may be useful under different scenarios. In this section we provide a description of these methods and their applicability; as well as their usage.

4.2.1 Data-Centric Dimension Reduction Methods

This class of methods is relevant when a data set (with inputs and observations/outputs) is readily available (thus, there is no room for intelligent sampling design to reveal parameter importance, and the data set is large (tens or hundreds of thousands or more). Some of these screening methods are:

Delta test : This method works best with relatively large (hundreds to thousands) random or quasi-random samples. Since this method involves minimization of certain 'noise' function, it may take long time to run this analysis. Screening results may be confirmed by using different sample sizes or by another method (e.g. sum-of-trees). This method is not suitable for very large input parameter set (e.g. > 50, maybe unless the data size is very large and this method is to be run on high performance computers). The calling sequence is:

```
[Linux] psuade
psuade> load dataSet <assume dataSet is in PSUADE format>
```

```
psuade> delta_test
...
```

Eta test : This method, as with with Delta test, works best with relatively large (hundreds to thousands) random or quasi-random samples. This method involves searching for nearest neighbors for all sample points in all dimensions, so its computational time grows linearly with the number of inputs and quadratically with the sample size. Screening results may be confirmed by using different sample sizes, different number of neighbors (selectable in the analysis expert mode), or by another method (e.g. sum-of-trees). The calling sequence is:

```
[Linux] psuade
psuade> load dataSet <assume dataSet is in PSUADE format>
psuade> eta_test
...
```

Sum-of-trees method : This method works well with relatively large random or quasi-random samples (thousands or more). It uses bisection techniques to form unbalanced trees and estimates sensitivities based on frequencies of bisection in each input parameter.

```
[Linux] psuade
psuade> load dataSet <assume dataSet is in PSUADE format>
psuade> sot_sa
...
```

Correlation Analysis : The classical Pearson correlaton analysis assumes that the model input-output relationship is more or less linear. The analysis result is a ranked list of correlation coefficients that measure the strengths of relationship between the model output and the individual inputs. Based on this list, users can select the subset that is deemed the most sensitive.

If the model input-input relationship is not linear but exhibits monotonic behavior (non-decreasing or non-increasing), a variant called the Spearman coefficients may be used as a 'qualitative' sensitivity measure (there is another set of coefficients called the Kendall coefficients). For general nonlinear input-output relationships, these types of analysis may give erroneous results. Thus, for general models this method should be supplemented with other methods to confirm a good subset selection.

Correlation analysis can be launched via either the batch or command line mode. The command line mode calling sequence is:

```
[Linux] psuade
psuade> load dataSet <assume dataSet is in PSUADE format>
psuade> ca
...
```

Since both methods are suitable for large samples, you are encouraged to run both of them and compare ranking results.

4.2.2 Dimension Reduction Methods for Near-Linear Models

If a data set is not already available and the simulation model is computationally expensive to evaluate, the choice of sampling design should depend on one's knowledge of the simulation models and how much computational resources are available. For linear or near-linear models, PSUADE currently provides the following screening methods:

Plackett-Burman : This is a gradient-based local sensitivity analysis design (and the corresponding lsa analysis). This method requires only $m + 1$ simulations (m is the number of uncertain parameters). To use this method, first create a Plackett-Burman sample (PBD), run simulations, and analyze with:

```
[Linux] psuade
psuade> load dataSet <assume dataSet is in PSUADE format>
psuade> lsa
...
```

Fractional Factorial (FF) : If there are also interaction terms (e.g. the model equation consists also of terms involving two or more inputs, then this class of designs and the corresponding (ff analysis may be useful. There are a number of fractional factorial sampling methods, the suitable choice of which depends on the types of parameter interactions assumed/inherent in the model. More exotic FF samples are needed to avoid certain types of 'confounding' (Look up fractional factorial designs to learn more). After an FF design has been created and run, the analyis goes like:

```
[Linux] psuade
psuade> load dataSet <assume dataSet is in PSUADE format>
psuade> ff
...
```

4.2.3 Dimension Reduction for Non-Linear Models

For general non-parametric models that may be nonlinear with significant input parameter interactions (higher order sensitivities), we recommend the Morris (MOAT) method, which is an effective variable selection method when the number of inputs is large (say, $10 - 100$'s). The cost of this method is about $4-10$ times (which is called the 'number of replications') the number of inputs. Because the Morris sampling is a structured sampling method, sample results can be analyzed systematically to detect irregular behaviors and anomalies. An example is given in the Examples/DimReduction/Morris20 directory.

In the following we show how to use PSUADE to set up the MOAT screening analysis. The PSUADE input file for a 20-dimension problem is given in the Examples/DimReduction/Morris20 directory):

```
PSUADE
INPUT
   dimension = 20
   variable  1 X1  =   0.0   1.0
   variable  2 X2  =   0.0   1.0
   ...
   variable 20 X20 =   0.0   1.0
END
OUTPUT
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = MOAT
   num_samples = 210
   randomize
END
APPLICATION
   driver = ./simulator
END
ANALYSIS
   analyzer method = MOAT
   printlevel 3
END
END
```

Here the sample size should be a multiple (usually $4-10$, 10 in this example) of $m+1$ where m is the number of inputs. The driver program can be constructed in a similar manner as before (and thus is not to be given here). Again, PSUADE is launched with this input file and screening results will be displayed at completion.

Alternatively, the PSUADE input file can be run without the analyer method = MOAT line. In this case, the 210 simulations will be launched, and the results are stored in the file 'psuadeData'. Subsequently, screening analysis can be performed interactively by launching PSUADE in the command line mode and run the screening analysis:

```
[Linux] psuade
**************************************************************
*      Welcome to PSUADE (version 2.1.0)
**************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration (2.1.0)
==============================================================
psuade> load simdata <Note: renamed from psuadeData>
```

```
load complete : nSamples = 210
nInputs  = 20
nOutputs = 1
psuade> ana_expert
psuade> moat
... (MOAT results) ...
...
Create screening diagram ? (y or n) y
Enter matlab/scilab screening diagram file name : screen
MOAT screening diagram matlab file = screen.m
...
Create scatter plot ? (y or n) y
Enter matlab/scilab scatter plot file name : scatter
...
psuade> quit
[Linux]
```

The command to run screening analysis is called moat. In the example above, moat is preceded by ana_expert, which turns on more analysis option (If this mode is off, moat analyzes and creates a Matlab ranking plot only. If this mode is on, options are offered to provide more detailed information such as scatter plots of individual 'gradients'.) Thereafter, you can launch Matlab and run matlabmoatbs.m to view the Morris ranking and other plots. The final decision on the set of important parameters should be based on expert judgment (by modellers who know the simulation physics well, with the help from analysts who know how to interpret the MOAT results).

Due to the use of coarse sampling strategy (that is, small sample size), errors may be made in identifying important parameters. There are two types of errors: (1) Type I error or when unimportant parameters are treated as important; and (2) Type II or when important parameters are treated as unimportant. Type I error is benign resulting only in higher computational cost for subsequent analysis. Type II is much more serious and safeguard against this error should be employed. As such, our recommendataion for parameter screening follows a multi-algorithmic approach consisting of a frequentist analysis such as the Morris screening and other screening methods, such as ranking based on approximate response surfaces for small samples. For these alternative methods, a response surface analysis can be performed with parameter rankings based on the hyperparameters obtained from certain response surfaces. The two methods available in this class are mars_sa (based on the MARS response surface method) and gp_sa (based on Kriging). To assess whether these methods may be useful, we recommend first analyzing the response surface cross validation errors (described in next section) to make sure the response surface gives the correct 'trend' (error distribution centers around zero and the parity plot roughly captures the function behavior). These methods differ from the response surface analysis (to be discussed next) in that high accuracy is not a major concern (since the objective is to give rough estimates that capture the trends).

In a multi-algorithmic approach, each selected analysis proposes a ranked list of parameter importance. The final selection should be based on merging the ranked lists, and confirmed by subject matter experts.

PSUADE also supports Morris screening for more complex scenarios such when there are correlations between the input parameters in the form of inequality constraints. In this case, a modified MOAT analysis is more suitable. Specifically, a MOAT design based on no correlations is first constructed, followed by an 'adjustment' step to adapt to the constraints. An example is given in the Examples/DimReduction/MOATConstraint.

4.3 Response Surface Analysis

As we can see from the previous exercise, even a relatively simple test function may require large samples to compute the statistics to sufficient accuracies. To reduce the computational demands, one approach is to generate a small sample for this function and find out if the model input-output relationship can be described using a curve-fitting (or response surface/surrogate) method. In this section we show how to use response surface tools in PSUADE to find the best response surface (surrogate, emulator, etc.) model for a simulation model.

4.3.1 Basic Steps in Response Surface Analysis

To illustrate the steps, we use the example in Examples/SimpleUQ/Bungee/Basic. We run the simulation model with a quasi-random sample of size 300. We then rename the result sample file 'psuadeData' to 'simdata' so that it will not be overwritten by other PSUADE commands (since 'psuadeData' is the PSUADE default output file name). The following gives a snapshot of how to perform response surface analysis in PSUADE's command line mode:

```
[Linux] psuade
************************************************************
*      Welcome to PSUADE (version 2.1.0)
************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration (2.1.0)
============================================================
psuade> load simdata
...
load complete : nSamples = 300
                nInputs  = 3
                nOutputs = 1
psuade> rscheck (NOTE: this command performs response surface analysis)
... list response surface types ....
Enter you choice? 1 ('1' is for linear regression model)
... (some analysis results show be shown)
```

```
Perform crosss validation ? (y or n) y
Enter the number of groups to validate : (2 - 100) 20 (Use 10 - 20)
... (more informatin will be displayed)
Random selection of leave-out groups ? (y or n) y (generally say yes)
...
RSA: final CV error  =   1.276e-03 (avg unscaled)
RSA: final CV error  =   3.658e-01 (avg   scaled)
RSA: final CV error  =   1.152e+00 (rms unscaled)
RSA: final CV error  =   3.545e+00 (rms   scaled)
RSA: final CV error  =   3.229e+00 (max unscaled, BASE=2.587e+00)
RSA: final CV error  =   5.685e+01 (max   scaled, BASE=1.802e-02)
RSA: final CV error  =   8.525e-02 (max   scaled by (ymax-ymin))
...
CV error file is RSFA_CV_err.m
************************************************************************
psuade> quit
[Linux]
```

After loading the sample in the command line mode, the rscheck (or rsvalidate) command is launched to perform a response surface analysis. Questions will be prompted on which response surface method to use, which output to analyze, additional information for tuning the selected response surface (when the response surface expert mode is on by issuing rs_expert before rscheck), and whether to perform cross validation (CV) or not.

Following the procedure above (when CV is selected), a file called 'RSFA_CV_err.m' will be created in your working directory. The analysis results can be visualized when this file is run in Matlab (or Scilab if so selected prior to calling rscheck). Two plots will be displayed: the left showing the distribution of the cross validation errors and the right (called parity plot) showing how good the selected response surface method is. In addition, when the response surface analysis expert mode has been turned on (using the ana_expert on command) before running rscheck, another file called 'RSFA_training_error.m' will be created. To assess the 'goodness' of the response surface, we recommend examining the following quantities:

1. The 'final CV errors' similar to those shown above give important information. For example, if there is a large departure of the scaled 'rms' error from zero, a significant systematic bias may be present. If the scaled maximum error is large (> 1) with non-negligible base (e.g. if BASE=0.53 and max scaled error is 0.5, it means the maximum error has been found to be $0.53 * 0.5 = 0.265$ when the output is 0.53), the fitting error may be significant.

2. Running 'RSFA_training_error.m' using Matlab will show a few plots. One plot shows the error distribution when the response surface is evaluated at the training set (also called 'resubstitution test'). If there are significant errors in this step, the response surface may be declared 'unfit'. Another plot shows the 'training errors' for individual

sample points. Yet another plot displays the interpolation errors with respect to each input and is useful in assessing which input parameters may need more attention.

3. The left plot in Figure 1 of 'RSFA_CV_err.m' gives the distribution of the cross validation errors. The desirable distribution should be centered around zero with small spread. Again, if the center is far away from zero, it indicates possible systematic bias.

4. The right plot in Figure 1 of 'RSFA_CV_err.m' compares the CV predictions with the actual simulation data. Ideally all points should lie on the diagonal line. For this example problem, since the function is a fourth-order polynomial and you use quadratic polynomials, significant CV errors showing systematic bias should be observed. For response surfaces that also provide prediction uncertainties (You have to go into the file and set the variable 'errBoundOn' to 1), the uncertainties will be represented as green or red vertical bars on the data points, whereby green means the prediction uncertainties are within 2-*sigma* (that is, 95% confidence interval with Gaussian assumption) of the actual points. If you observe on the parity plot that the extreme (lowest and highest) points are off, this indicates possible 'extrapolation' error (the hold-out points are outside the convex hull of the selected training set) or systematic bias. In this case, a closer examination of these results with those of the resubstitution test should be done.

5. You can also change the 'morePlots' variable in 'RSFA_CV_err.m' to 1 and re-run in Matlab to create another plot showing the normalized CV errors (with respect to the output values). In our example problem, we observe that when quadratic regression is used, the result response surface incurs larger errors at the low end of the output range.

6. One more caveat: sometimes the cross validation errors may appear to be small, but the small errors may be deceiving. For example, if the sample outputs vary between 100 and 101, a scaled maximum CV error of 0.001 gives an absolute error of ≈ 0.1, which may not be significant compared to 100, but may be significant when we notice that the output range is small (1.0). Because of this possibility, an additional quantity is also given: max scaled by ymax-ymin, which should be small also.

After some preliminary analysis, it should be clear that fitting with linear regression does not lead to a satisfactory response surface. Since the test function is not a simple linear model, more exotic response surface methods such as Gaussian process should give better results. If you are not satisfied with all available response surfaces, you may

1. Experiment with your own basis functions using the user-defined regression option. In this case, the response surface function is expected to be in the form

$$Y = \sum_{i=1}^{n} a_i \phi_i(X).$$

where X is the set of uncertain parameters and a_i's are the coefficients to be determined using regression techniques within PSUADE. To use this option you need to provide PSUADE with the following information:

   (a) the number of terms n, and
   (b) an executable file that returns the values of all the terms $\phi_i(X)$ given X for all points requested by PSUADE.

2. Add more sample point via sample refinement (using the refine or a_refine functions, or just create another sample and append to the current sample using the loadmore command) until the response surface can be 'validated'.

Once you have identified a suitable response surface, it is ready for subsequent analysis. Note that there are two types of response surfaces provided by PSUADE - the 'deterministic' ones that predict only the output mean given the input values (splines, MARS, sparse grid, SVM), and the ones that also provide prediction uncertainties associated with the prediction means (polynomial and user-defined regressions, MARS with bootstrapping, Gaussian process, Kriging, sum-of-trees), so if you desire to include response surface uncertainties in subsequent analysis, select the ones that provide prediction uncertainties. In addition, when a response surface type can provide prediction uncertainties, a user may perform an additional response surface analysis: examining prediction errors over the parameter space. This can be achieved by first generating a large space-filling sample (say, 100k points using LPTAU), followed by writing the large sample in a file in the format that can be used by the subsequent command, then constructing a response surface, and finally propagating the sample through the response surface with prediction uncertainties (another qualitative way is to use the rssd_ua command to visualize prediction uncertainties with respect to a small number of parameters). The detailed steps are as follow:

1. Make a copy of the sample file (say, cp Sample newPsuade.in) where Sample contains the original sample for building response surface).
2. Strip the file 'newPsuade.in' of the PSUADE_IO section.
3. Set the sampling method to be LPTAU and sample size to be, say, 100000.
4. Make sure driver is set to 'NONE'.
5. Run PSUADE with newPsuade.in (in batch mode) to create 'psuadeData', which contains the large sample (input only).
6. Launch PSUADE, load 'psuadeData' and use iwrite LargeSample to write the large sample in a different format to a file called 'LargeSample'.
7. Launch PSUADE and load your original sample ('Sample').
8. Inside PSUADE, launch the command rscreate and select response surface type (one that provides prediction uncertainties).

9. Next, launch the command rseval and enter 'LargeSample' when prompted for the evaluation sample file. Select fuzzy evaluation.

10. When finished, a file called 'eval_sample.std' should have been created. Use read_std to read this file in command line mode and write LargeSampleResults to store the large sample with predictions (Output 1) and prediction uncertainties (Output 2) in PSUADE data format.

11. Now you can load 'LargeSampleResults' and use commands such as ua (for Output 2) to compute summary statistics and to create Matlab file for visualizing the distribution of prediction uncertainties, which gives you a sense of prediction uncertainties overall the parameter space (examine the mean, standard deviation, and max/min of this plot). You can also use splot to examine how prediction uncertainties behave at different parts of the parameter space.

There are a number of other response surface validation functions in PSUADE. In summary, PSUADE offers the following response surface validation tests:

- The k-fold cross validation in rscheck or rsvalidate;
- rstest_hs (hold-out test), which is a stripped-down version of the CV in rscheck;
- rstest_rcv, which is a more rigorous version of the CV in rscheck;
- rstest_ts (validation on the training set), which performs the same resubstitution test as in rscheck when ana_expert mode is on;
- sf rstest_gt (generalization test), which is the most rigorous validation test that checks how 'extrapolable' the method is on the training sample.

Use help rs in the command line mode to see the list of the available commands for response surface valiation. Make sure you run several tests to gain confidence in the validity of your chosen response surface.

4.3.2 Suggestions on Response Surface Analysis

There are some 30 response surface methods in PSUADE. Most of these methods are suitable for relatively low (say, < 20) input dimensions (except the class of RS methods for homogeneous inputs to be discussed later). A few important guidelines for fitting response surfaces are:

- Space-fillingness of the training sample is very important in building a response surface. If, for example, the corners are not covered sufficiently, uncertainty/sensitivity analysis or inference that explores the corners may result in larger errors.
- Some response surfaces are pegged to a specific sampling method. For example,the 2D and 3D spline methods rely on the use of factorial design.

- Some response surfaces are very computationally intensive (e.g. GP, Kriging). So if you have large samples, try the cheaper response surfaces (e.g. MARS) first.
- If you have large samples and cheaper response surfaces do not perform well, you can try using the multi-domain Kriging and/or GP methods, which are computationally less expensive than their single-domain counterparts. Even MARS may be too expensive for very large samples, so the multi-domain MARS is included to handle the very large sample cases.
- Always perform cross validation or holdout tests to determine choice of response surfaces and assess their prediction power (Re-substitution test may not be good enough since some response surface methods such as Kriging interpolate exactly at the sample points).
- There may exist large errors for some sample points in cross validation. This may be due to extrapolation errors - that the hold-out points are outside the convex hull of the training points. In this case, try the rstest_rcv to perform a more rigorous analysis.

4.3.3 Specialized Response Surfaces

In addition to the classical response surface types, PSUADE also provides a few specialized response surface types, as presented in this section.

RS for Homogeneous inputs : There are simulation models that have high-dimensional inputs (e.g. 50 or more) which are equally sensitive and thus cannot be down-selected. However, if the inputs are interchangeable, meaning that running the simulation model with any permutation of the inputs gives the same result, then a relatively inexpensive response surface may be feasible. An example is in agent-based models where the agents are exercising the same strategy but each will select his/her own parameter values. In this case, the class of homogeneous response surface can be considered. PSUADE provides 3 homogeneous response surface methods: homogeneous polynomial regression (based on the Legendre polynomials), homogeneous Kriging and also homogeneous GP. To experiment with this class of methods, go to 'Examples/RSTests' directory and then to the 'HomogeneousInps' sub-directory, compile the model 'simulator.c', and run either the 50- or 100-input case. Afterward, perform response surface analys (rscheck) with one of the 3 homogeneous response surface methods.

RS for Heteroscedastic Models : Another characteristics of agent-based models is heteroscedasticity, meaning that the simulation model is stochastic and that variances induced by stochasticity are different for each parameter setting. PSUADE provides two approaches for this type of problem:

1. Create separate response surfaces for the model output mean and standard deviation - by using the rm_dup (remove duplicates) command to create two samples : 'mean' and 'standard deviation'. Subsequently, each sample can be analyzed individually.

2. Create a compressed sample (from the large sample with replicated sample points due to stochastic simulations) into quantiles (an additional input will be created to quantify the quantiles and the final sample size will be the number of quantiles times the number of unique sample points), and then select any response surface (but if, in addition, all inputs are homogeneous, you can use the quantile GP to reduce computational complexity).

User-defined Nonlinear Regression : Since most models are nonlinear with respect to the inputs, PSUADE provides a nonlinear regression response surface type, the functional form of which is the summation of a finite number of terms corresponding to some nonlinear functions of any combinations of the inputs, i.e.,

$$Y = \sum_{i=1}^{n} a_i f_i(X)$$

where $f_i(X)$ are the nonlinear terms provided by users (in, for example, Python scripts) and a_i's are estimated by PSUADE using the training sample. PSUADE also provides prediction uncertainties. An example is given in the RSTests/UserRegression subdirectory of the Examples directory.

4.4 Response Surface-Based Uncertainty Analysis

To perform uncertainty analysis on a response surface with PSUADE, one should use the rsua command. The steps are:

1. Prepare, run, and validate a sample for building the response surface (say, the training sample file is 'simdata').

2. When you launch PSUADE and run the rsua command, you will be prompted to provide your own sample file to propagate uncertainty through the response surface, or request to generate a sample for you based on the the input distributions prescribed in 'simdata'. For the former, use the following steps to create a large sample:

   (a) Make a copy of 'simdata' (say, to 'ps2.in') and strip this file of its PSUADE_IO section.

   (b) Modify the INPUT section of 'ps2.in' if needed to prescribe input distributions (e.g. change input ranges or add distribution types to reflect the true input distributions).

   (c) Start PSUADE in command line mode, load 'ps2.in', and use the gensample command to create a large sample (say, of size 50000 or larger). Write this large to another file, say, 'uaSample'.

3. Now launch PSUADE in command line mode to run uncertainty analysis:

   (a) Load simdata,

(b) Run the rsua command,

(c) Enter uaSample when prompted for a sample file, if so selected, and

(d) When completed, a file called 'matlabrsua.m' will have been created.

4. Next, launch Matlab and run 'matlabrsua'.

(a) The top left plot gives the model output probability distribution,

(b) The bottom left plot gives the model output probability distributions when response surface uncertainties are also included,

(c) The right plot gives the corresponding cumulative probability distributions, and

Another command for response surface-based uncertainty analysis is the rsuab command, which creates many response surfaces using bootstrapping and the generates predictions from each response surface. If the response surface uncertainty is significant, the bootstrapped probability distribution curves will be fuzzy.

4.5 Quantitative Sensitivity Analysis

Quantitative sensitivity analysis are based on variance decomposition and include methods for main effect, pairwise interaction effect, group main effect, and total sensitivity analysis. Since quantitative sensitivity analysis requires a large sample, it is often performed on validated response surfaces, unless simulations are inexpensive.

Main effect analysis studies the first-order sensitivities of individual input parameter based on variance decomposition. Sensitivity (Sobol') indices can be computed using one of the following ways:

1. If a large sample (thousands or larger) has already been evaluated, and you would like to use this sample to estimate main effect (that is, without using response surfaces), you can just load the sample in command line mode and use the me (main effect) command to compute sensitivity indices;

2. If you want to use raw simulation data (because response surface fitting is not satisfactory) to compute sensitivity indices, and the simulation model is inexpensive, you can create and run a large replicated LH design and then use me to compute main effects;

3. Instead of using replicated Latin hypercube, create and run a FAST sampling design and analyze using the FAST (fast) method; or

4. Construct a response surface, create a large replicated Latin hypercube sample, evaluate this large sample with the response surface, and finally us me to compute the Sobol' indices,

5. Construct a response surface and use direct numerical integration to compute the Sobol' indices (rssobol1).

In the following example, we describe these approaches.

4.5.1 Main Effect Analysis Using A Large Raw Sample

When a large 'space-filling' sample is available (with inputs and outputs), main effects can be computed by binning and computing variance of condition expectations. So suppose the large sample is stored in 'simdata' (in PSUADE standard format), the following session shows how to compute sensitivity indices:

```
[Linux] psuade
************************************************************
*      Welcome to PSUADE (version 2.1.0)
************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration (2.1.0)
============================================================
psuade> load simdata
load complete : nSamples = 100000
                nInputs  = 8
                nOutputs = 1
psuade> me
...
Choose which output
============================================================
Input    1, normalized 1st-order effect = 7.17e-1 (raw = 3.34e-1)
Input    2, normalized 1st-order effect = 1.81e-1 (raw = 8.46e-2)
...
Input    8, normalized 1st-order effect = 2.99e-3 (raw = 1.39e-3)
Total VCE =  9.5e-1
************************************************************
MainEffect plot matlab file = matlabme.m
psuade> quit
[Linux]
```

At the conclusion of the session, the main effects will be displayed. In addition, a Matlab file ('matlabme.m') will also be available for visualizing main effects. This method supports correlated inputs in the sense that if correlation has been embedded in 'simdata', the final main effect result will capture this correlation.

4.5.2 Main Effect Analysis Using Replicated Latin Hypercube

Replicated Latin hypercube (RLH) design has the characteristics that make it a natural candidate for first-order Sobol' analysis (main effect can be computed using a large random sample, but the results are less accurate than RLH for the same sample size). In the current example, the input file ('psuadeLH.in') is given as follow (this can be found in the Examples/SimpleUQ/SobolG directory:

```
PSUADE
INPUT
   dimension = 8
   variable  1 X1  =   0.0   1.0
   variable  2 X2  =   0.0   1.0
   variable  3 X3  =   0.0   1.0
   variable  4 X4  =   0.0   1.0
   variable  5 X5  =   0.0   1.0
   variable  6 X6  =   0.0   1.0
   variable  7 X7  =   0.0   1.0
   variable  8 X8  =   0.0   1.0
END
OUTPUT
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = LH
   num_samples = 1000
   num_replications = 50
   randomize
END
APPLICATION
   driver = ./simulator
END
ANALYSIS
   analyzer method = MainEffect
END
END
```

Here the sample size is 1000 based on 50 replications of Latin hypercube each with $1000/50 = 20$ levels. To run this analysis, go to SimpleUQ/SobolG (in Examples) and run the following:

```
[Linux] cc -o simulator simulator.c -lm
[Linux] psuade psuadeLH.in
```

At the conclusion of the analysis, main effect statistics will be displayed. More information will be displayed if the printlevel level is set higher. In addition, a Matlab plot of the main effects ('matlabme.m') will have been created. This method has the limitation that the parameter space is expected to be a hyper-rectangle, although a modified method has been proposed to handle correlated inputs.

4.5.3 Sensitivity Analysis Using the FAST Method

The Fourier Amplitude Sensitivity Test (FAST) is a method for computing first-order and total-order Sobol' indices. It is applied to raw samples (rather than response surfaces constructed from raw samples) although the 'raw' samples can be evaluated using response surfaces instead of actual simulations. As an example, consider the input file ('psuadeFAST.in) below (which can be found in the SimpleUQ/SobolG directory):

```
PSUADE
INPUT
   dimension = 8
   variable  1 X1  =   0.0   1.0
   variable  2 X2  =   0.0   1.0
   variable  3 X3  =   0.0   1.0
   variable  4 X4  =   0.0   1.0
   variable  5 X5  =   0.0   1.0
   variable  6 X6  =   0.0   1.0
   variable  7 X7  =   0.0   1.0
   variable  8 X8  =   0.0   1.0
END
OUTPUT
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = FAST
   num_samples = 1000
END
APPLICATION
   driver = ./simulator
END
ANALYSIS
   analyzer method = FAST
END
END
```

The key differences in this file from the one in the last section are the sampling method and the analysis method (both are now FAST). The FAST sample size is fixed given the input dimension, so we can just give a number and let PSUADE suggest the correct size (1001 for 8 inputs). To run this analysis, go to Examples/SimpleUQ/SobolG and run the following:

```
[Linux] psuade psuadeFAST.in
```

At the conclusion of the analysis, main effect statistics will be displayed. More information will be displayed if the printlevel level is increased.

The same can be run in 2 steps: (1) run psuadeFAST.in with its analysis method turned off; and (2) start PSUADE in command line mode, load the 'psuadeData' file and run fast.

4.5.4 Main Effect Analysis Using RLH on Response Surfaces (RS)

Since a sample of size 1000 may not be sufficient to give reasonable results, we describe in the next example the use of the replicated Latin hypercube on response surfaces. First, run PSUADE on the file psuadeRS.in in SimpleUQ/SobolG to create a sample of size 1000 (this is an arbitrary sample size, but do try different sample sizes and examine the main effect results) for building a response surface. After renaming 'psuadeData' to something like 'simdata', run PSUADE again with 'psuadeRSME.in', which is given below (to select which response surface type to use, modify the rstype variable in 'simdata') to compute the sensitivity indices and obtain a few Matlab plot files (specifically, the scatter plots show how the output behaves with respect to each individual input and the bootstrapped plot include errors with each main effect).

```
PSUADE
INPUT
   dimension = 8
   variable 1 X1  =   0.0   1.0
   variable 2 X2  =   0.0   1.0
   variable 3 X3  =   0.0   1.0
   variable 4 X4  =   0.0   1.0
   variable 5 X5  =   0.0   1.0
   variable 6 X6  =   0.0   1.0
   variable 7 X7  =   0.0   1.0
   variable 8 X8  =   0.0   1.0
END
OUTPUT
   dimension = 1
   variable 1 Y1
END
METHOD
   sampling = LH
   num_samples = 100000
   num_replications = 50
   randomize
END
APPLICATION
   driver = ./simdata
END
ANALYSIS
   analyzer method = MainEffect
```

```
END
END
```

This procedure describes one powerful feature offered by PSUADE, namely, that swapping between the use of simulation model or response surface can be done by just changing one line (driver in the input script. Also, this analysis can also be performed via the command line interpreter: by running the above script without setting the analysis method, loading the result file, and use the me command). Note that me works for any sampling design (not just replicated LH) although the result will be less accurate in general.

4.5.5 Main Effect Analysis Using Direct Numerical Integration on RS

Another approach to first-order sensitivity analysis is to use direct numerical integration on response surfaces. Once the response surface has been validated and deemed satisfactory, The sample file (say, 'simdata') may be loaded into PSUADE's command line interpreter for analysis:

```
[Linux] psuade
*************************************************************
*      Welcome to PSUADE (version 2.1.0)
*************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration (2.1.0)
=============================================================
psuade> load simdata
load complete : nSamples = 1000
                nInputs  = 8
                nOutputs = 1
psuade> rssobol1
...
Choose which output
Choose response surface type
...
...
RSMSobol1: ordered normalized VCE:
...
RSMSobol1: Normalized VCE for input   1 = 7.210e-01
...
rssobol1 plot file = matlabrssobol1.m.
psuade> quit
[Linux]
```

At the conclusion of the session, the main effects will be displayed. In addition, a Matlab file will also be available for visualizing the main effects. This analysis is different from

me on response surfaces in that it supports correlated inputs (both inequality constraints and correlations in input distributions). There is another similar command called rssobol1b which provides error bars to each sensitivity index based on bootstrapping.

4.5.6 Higher-order Global Sensitivity Analysis

PSUADE also provides the functionalities to perform second-order (actually first- plus second-order) sensitivity analysis on raw sample data (that is, not response surface evaluations) using ie in command line mode. In this case, instead of using replicated Latin hypercube, replicated orthogonal array design will be more appropriate although any space-filling sampling design should work. To achieve sufficient accuracy, however, very large sample is needed, and hence it makes more sense to use response surfaces. To experiment with this analysis using response surfaces, you can run PSUADE with 'psuadeRSIE.in' in the SimpleUQ/SobolG directory.

PSUADE also provides the tsi command to perform total sensitivity analysis on a given sample. This sample needs to be even larger and with small number of parameters ($<= 21$) to give meaningful results. To experiment with this function, Again, the response surface-based method is recommended, which is the rssoboltsi method, which can be launched in a similar manner as rssobol1. Again, rssobol2b and rssoboltsib are the bootstrapped versions for rssobol2 and rssoboltsi.

4.5.7 Global Sensitivity Analysis for Correlated Inputs

Again, response surface-based sensitivity analysis methods offer more flexibility in handling non-uniform distributions and inequality input constraints. For example, you can experiment with different input distributions by modifying the input section of your psuadeRSME.in (or psuadeRSIE.in, etc.) file. If you use the command line interpreter to run rssobol1, rssobol2 or rssoboltsi, you can modify the input section of your simdata file that is used to build a response surface.

More sophisticated quantitative sensitivity analyses involving input correlations (governed by some inequality constraints, not joint PDFs) are available. For example, there are two ways to include input constraints into main effect analysis:

1. If you are using replicated Latin hypercube, say, in psuadeRSME.in, the steps are:
   (a) Generate a large sample and evaluate this sample using a response surface (e.g. using the 'simdata') file created previously) by running PSUADE with psuadeRSME.in which has its analysis method (analyzer method = MainEffect) disabled;
   (b) The 'psuadeData' file created by the previous step has a large sample evaluated with a response surface. The next step is to apply input constraints to filter out infeasible sample points from this file and write the filtered sample to another file (say, 'LargeSample');
   (c) Finally, launch PSUADE, load 'LargeSample', and run me.

2. Alternatively, you can also use numerical integration ('rssobol1'). In this case, you will need to turn on the 'rs_constraint' line in your training sample data file (simdata). The syntax is:

```
analyzer rs_constraint = constrSample indexFile Lbound Ubound
```

where 'constrSample' is another PSUADE sample, 'indexFile' file contains a subset of input indices for constraining, and 'Lbound' and 'Ubound' are lower and upper bounds of the feasible region. For example, if you desire to impose constraint on input 2 and 3 such that $0 < X_2 + X_3 < 1$ (say, X_1 is not involved in the constraints), then 'constrSample' should contain a sample for the function $Y = X_2 + X_3$; 'indexFile' should contain 2 and 3; 'Lbound'= 0; and 'Ubound'= 1. If X_1 is part of the 'constrSample' (say, the sample contains the functional relationship $Y = X_1 + X_2 + X_3$) but it is not involved in the constraints ($0 < X_2 + X_3 < 1$), then in order to disable $X1$, do the following: (1) the 'indexFile' file should contain input 1, 2, and 3, but the line corresponding to input 1 should begin with a '0' followed by a default value (0 in this case to mimic the desired function), and 'Lbound' and 'Ubound' should be set the same as before. To learn more about this topic, go to Examples/CorrelatedInps and follow the example.

4.6 Mixed Aleatory-Epistemic Uncertainty Analysis

Oftentimes, a simulation model has two types of uncertain parameters: aleatory and epistemic. Aleatory uncertainty describes natural randomness in a process (and so it is sometimes called variability rather than uncertainty) and is parametrized by a probability distribution (and each parameter value with nonzero probability is a valid value). As such, this type of uncertainty is not reducible, meaning that its probability distribution does not change with more experiments (but its characterization may improve with more experiments). Epistemic uncertainty, on the other hand, is due to a lack of knowledge of the true parameter values, which are usually unique (e.g. a physical constant is assumed to be unique but its true value may not be known exactly). Uncertainties for epistemic parameters are best described by ranges (upper and lower bounds) which look like uniform distributions, or union of disjoint ranges.

To quantify uncertainty in the aleatory-epistemic settings, it is not sufficient just to display a single output probability distribution. Rather, it should be an ensemble of probability distributions with each probability curve corresponding to the distributions due to variations in the aleatory parameters at a sample point drawn from the ranges of the epistemic parameters (this is sometimes called a probability-box or p-box).

PSUADE provides the functionality to perform the aleatory-epistemic analysis. Since this analysis is computationally intensive, it is currently applied to response surfaces only and not directly to simulation models. Hence, the first step in this analysis is to create and validate a response surface (say the training sample for response surface has been prepared

in 'simdata'). Next, prescribe probability distributions for the aleatoric parameters by modifying the 'INPUT' section in 'simdata' (For the epistemic variables, normally parameter ranges - lower and upper bounds - should suffice. For more complex scenarios where the possible ranges are union of disjoint ranges, they can be handle either by running this analysis multiple times each using a different range and then concatenate them, or by using the 'S' probability distribution type). Afterward, launch PSUADE in command line mode, load 'simdata', and run aeua. Users will be prompted to select which input parameters are epistemic. At the completion of this analysis, PSUADE will output a file called 'matlabaeua.m' for visualizing the ensemble of cumulative distributions.

Another similar analysis is the so-called 'second-order uncertainty analysis', which generates an ensemble of probability distributions as a result of uncertainties about the input distribution parameters. For example, let a certain parameter have a normal distribution with mean and standard deviation 1.2 and 0.5, respectively. Suppose there is uncertainty about the mean; then this analysis soua draws samples from this 'second-level' parameter uncertainties and generates distributions for each. At completion, this command will create a 'matlabsoua.m' file for visualizing the second-order uncertainties.

4.7 Statistical Inference

Consider a simulation model which computes the following fubnction:

$$Y = F(X; \theta, \omega)$$

where X is a model parameter (e.g. design or control parameter of a given physical system configuration); Y is the corresponding model output of interest (assume to be scalar to simplify discussion); and θ and ω are uncertain parameters that are initially prescribed with their 'prior' probability distributions. Suppose we also have a set of M observation data $D = \{X_i \tilde{Y}_i\}_{i=1}^{M}$ (X_i is the input value of the i-th experiment) for calibrating θ. One way to find the best θ is to perform a deterministic numerical optimization. If the experimental data D are noisy (in both X_i's and $\tilde{Y}_i$'s), an alternative is to use statistical inferences, the most popular of which is the one based on the Bayes rule. Due to the high simulation demands, statistical inference is often performed using surrogates (or response surfaces) instead of the actual simulation models.

4.7.1 Statistical Inference Essentials

To perform response surface-based statistical inference, we need the following:

1. an ensemble of pre-computed simulations (e.g. a Latin hypercube of size N) to be used for building a response surface ($\hat{F}(X; \theta, \omega$, where ω does not need to be a parameter if it has been fixed to ω^*) to approximate F (response surface is needed because a typical statistical inference requires many function evaluations; otherwise, the computational cost will be prohibitive),

2. calibration parameters (θ in this example) and their distributions (priors, e.g. uniform between 0 and 1),

3. uncertain parameters that are not to be calibrated (ω in this example) and their distributions (assume for simplicity that $\omega = \omega^*$ is a fixed value); and

4. an observation data set (D in this example) together with the corresponding observation uncertainties (which are usually associated with the experimental output $\tilde{Y}$, but may also be present in the experimental inputs X).

A few other decisions need to be made:

1. Which form of likelihood function is to be used (a popular one is the Gaussian function, which is the only one PSUADE offers, but users can prescribe their own too);

2. Whether discrepancy modeling is to be included in the inference (PSUADE also offers this option); and

3. Whether response surface errors are to be included in the inference.

Once these decisions have been made, statistical inference is now ready to be set in motion, following the Bayes' rule:

$$\pi(\theta) \sim L(D|\theta, \omega^*)p(\theta)$$

where $L(D|\theta, \omega^*)$ is the likelihood function, $p(\theta)$ is the prior distribution of θ; and $\pi(\theta)$ is posterior distribution of θ. If ω has a distribution and/or the experimental inputs are uncertain, an additional 'marginalization' is needed to 'integrate out' uncertainties associated with ω, so that the final posterior of θ is independent of ω. Popular algorithms to perform this inference are the Markov Chain Monte Carlo (MCMC) algorithm and its variants. PSUADE currently provides 3 options: Gibbs, Metropolis-Hastings, and brute-force. There may be other variants that are more efficient than these methods. Nevertheless, since our statistical inferences are based on the use of response surfaces rather than the actual simulation models, the efficiency impact is expected to be small.

There are other advanced features in PSUADE's MCMC method such as tuning MCMC parameters (e.g. number of chains for Gibbs-MCMC, termination criterion) and support for inferences in the presence of all four types of parameters: design, calibration, fixed, and uncertain (see examples below). In using the multi-chain Gibbs-MCMC, inference can be accelerated by running this scheme in parallel. This capability will require PSUADE be compiled using MPI, an option called 'PARALLEL_BUILD' that can be turned on during installation.

4.7.2 Discrepancy Modeling

If discrepancy modeling is to be used, the probabilistic prediction model (in place of the response surface $\hat{F}$ used with no discrepancy modeling) becomes

$$Y = \hat{F}(X; \theta, \omega^*) + \eta(X)$$

where $\eta(X)$ is the discrepancy function (a function of a subset or the full set of the design parameters X, if it is multivariate), which is constructed from taking the differences of the experimental data and predictions at the experimental conditions (X_i's):

$$\eta(X_i) = \tilde{Y}_i - \hat{F}(X_i; \theta^*, \omega^*); i = 1, \cdots, M.$$

Notice here that to construct a unique response surface for $\eta(X)$, θ^* is set to a pre-selected fixed value by users based on their expert knowledge or to some mode of its distribution (e.g. mean or peak). However, the requirement to pre-select θ^* has the undesirable 'confounding' effect, that the 'true' posterior distribution of θ has been modified because of this pre-selected choice of θ (other choices of θ^* may give different posteriors $\pi(\theta)$. Thus, including discrepancy modeling helps improve model prediction but not so much with model improvement (improvement in learning the best value of θ in the model, which may be acceptable anyway because including discrepancy modeling implies the user's assumption that the model is imprecise anyway). To use discrepancy modeling for model improvement, a suggestion is to try different θ^*'s and examine the behavior of the discrepancy functions for scientific intuition (for example, if the discrepancy is significant at lower temperatures, more efforts can be expended to improve the model at low temperatures.)

Discrepancy modeling is available in PSUADE, which creates internally a sample of size M corresponding to the differences between the observation data and the corresponding function values (that is, $\{X_i\eta(X_i)\}$) at the experimental design points X_i's. PSUADE's Bayesian calibration provides an option to store this discrepancy sample to a file for further examination (e.g. response surface analysis).

One final caution of using discrepancy modeling: that it should only be used if sufficient experimental data is available (a rule of thumb: M should be about 10 times the number of design parameters in η). In addition, once the discrepancy sample has been created by PSUADE after inference (the file name is 'psDiscrepancyModel'), perform a response surface analysis of the sample to see whether a good response surface can be constructed from this sample. The primary reason is that a discrepancy response surface of poor quality cannot be used with confidence in post-inference predictions.

4.7.3 An PSUADE Example

To give an example, let $M = 4$ be the number of observations at the design points $\{X_i\}_{i=1}^4$, and let the observation uncertainty be 0.1 (standard deviation). The observation data should be encoded into a 'specification' file, say, 'mcmcFile', that contains

```
4 1 1 1
1 <X_1> <Y_1> 0.1
2 <X_2> <Y_2> 0.1
3 <X_3> <Y_3> 0.1
4 <X_4> <Y_4> 0.1
```

The first line specifies that there are four observations, 1 model/observation output, and 1 design parameter (X in this example), which is parameter 1 in the INPUT section. Each of

the next four lines consists of the data set number (in order from 1 to 4 in this example), the corresponding design parameter value ($X_i, i = 1, \cdots, 4$), and the observation data value (Y_i) and its error (0.1 in this case) for experiment i. These information will be used by PSUADE to construct the likelihood function needed by the inference method:

$$L(D|\theta) \sim e^{\sum_{i=1}^{M}(\hat{F}(X_i;\theta)-\tilde{Y}_i)^2/\sigma_i^2}$$

where $sigma_i$ is the standard deviation of experiment i, and if discrepancy modeling is used, $\hat{F}$ should be replaced with $\hat{F}(X,\theta) + \eta(X)$. In addition, if response surface uncertainty is included, σ_i^2 should be replaced with $\sigma_i^2 + \tau_i^2$ where τ_i^2 is the response surface prediction standard deviation generated by the probabilistic response surfaces such as Gaussian process or polynomial regression. Note that the likelihood function is created by PSUADE when the specification file is provided. Later on in this section, we will describe how to create likelihood functions different from the one above.

To create a sample from the simulator (function F), we create a Latin hypercube sample of, for example, size 100. The PSUADE input file (say, 'psuadeRS.in') to create the Latin hypercube sample is:

```
PSUADE
INPUT
   dimension = 3
   variable   1 X     = 0.0  1.0
   variable   2 THETA = 0.0  1.0
   variable   2 OMEGA = 0.0  1.0
END
OUTPUT
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = LH
   num_samples = 100
END
APPLICATION
   driver = ./simulator
END
ANALYSIS
   printlevel 1
END
END
```

Again, simply compile the 'simulator.c' program, run the following command

```
[Linux] psuade psuadeRS.in
```

and then rename the result data file 'psuadeData' to, say, 'simdata'. After the preparation steps have been completed (make sure to validate your response surface), statistical inference can be launched by:

```
[Linux] psuade
**************************************************************
*      Welcome to PSUADE (version 2.1.0)
**************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration (2.1.0)
==============================================================
psuade> load simdata
load complete : nSamples = 100
                nInputs  = 2
                nOutputs = 1
psuade> ana_expert (use expert mode to turn on discrepancy mode)
psuade> rsmcmc
.....
===> Enter the spec file for building likelihood function : mcmcFile
.....
Output 1
Enter you choice (for response surface type) ? 0
.....
<say yes to discrepancy modeling, use default for other options)
MCMC BEGINS
 10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
.....
MCMC completed
**************************************************************************
MCMC: matlabmcmc2.m file (2-input analysis) is ready.
**************************************************************************
psuade> quit
[Linux]
```

If the inference is completed successfully, a Matlab (or Scilab) file called 'matlabmcmc2.m' ('scilabmcmc2.sci') will have been created that contains both the prior and posterior distributions for the purpose of visualization. Embedded in this Matlab file are some useful inference information. For example, the best negative log-likelihood value will be displayed in the title of the posterior plots. Large departure of this likelihood value from zero may indicate a strong systematic bias and thus the need for discrepancy modeling (if not already turned on). Also, the red vertical line in each single-parameter posterior plot (the plots on the diagonal) indicates the parameter value at the maximum likelihood point. Note that the maximum likelihood point does not necessarily reside at the peak of the individual single-parameter

probability distributions (for example, in the case of multi-modal posteriors) because these are marginalized distributions. In addition, an optional output from the inference is a posterior sample written to the 'MCMCPostSample' file. This file stores a large sample drawn from the parameters' posterior distributions. At the end of this file the maximum likelihood solution is included together with more information about the details of the likelihood function for further diagnosis (for example, these details can help pinpoint the anomalous experiments - the ones that give large negative log-likelihood value.)

One may ask: if the likelihood function is derived from many experiments each of which corresponds to a different training sample (so the step of loading a single training sample does not work) and a different output, can PSUADE perform inference on this problem? The answer is yes and here is how to do it:

- Instead of loading a single training sample ('simdata') containing multiple output variables, load different training samples for different outputs. This is done by first removing the whole PSUADE_IO section 'simdata' but keeping the input section (that is, the input parameters and ranges should be the same for all experiments although the model may be different) and the output section.

- After loading the 'modified' sample file, run 'rsmcmc' as before. PSUADE will recognize that a sample has not been given and the inference program will ask, for each output, a sample file that will be used to create the response surface for that output.

- After all output sample files have been entered and response surface types selected, the rest of the 'rsmcmc' command will be the same.

Another flexibility offered in PSUADE is the construction of the likelihood function. If users prefer to prescribe their own likelihood function, they can compute the negative log-likelihood value themselves (and multiply it by 2.0) as a scalar model output (by assimilating the experimental data instead of providing the data to PSUADE), and enter zero mean and standard deviation of 1.0 when prompted. PSUADE will take the user-provided negative log-likelihood and create the likelihood value by squaring its difference with the provided mean, dividing the result by the square of the provided standard deviation, multiplying it by $-1/2$ and finally exponentiating the result.

4.7.4 Another PSUADE Example

In a more complex scenario, suppose ω in the following function

$$Y = F(X; \theta, \omega)$$

is not to be fixed to some default value but is itself an uncertain parameter with a probability distribution derived from previous experiments and is not to be calibrated in the present experiments. Applying statistical inference in this case requires marginalization or integrating out with respect to the uncertainties of ω. PSUADE handles this scenario via the following steps:

1. Insert a line in the simulation sample file ('simdata' described previously) in the ANALYSIS section in the following form where 'rsIndexFile' has the form described in the next step :

```
analyzer rs\_index\_file = rsIndexFile'
```

2. Create the file 'rsIndexFile' in the following format:

```
3
1 1 0
2 2 0
3 999 1 PriorSample3
```

   Here the first line says that there are 3 parameters (X, θ and ω) in total. The next 3 lines describe the nature of the 3 parameters. Specifically, parameter 1 (X) and 2 (θ) are as they are, but parameter 2 is an uncertain parameter (but not a calibration parameter) whose probability distribution is represented in column 1 of the file 'PriorSample3', which should have the same format as that of a MCMC posterior sample file.

Once these 2 steps are completed, one only needs to follow the same MCMC command sequence as before. At the end of inference, 'MCMCPostSample' will contain the posterior sample for X, θ, and ω, but only the column corresponding to the calibration parameter (θ) is true posteriors to be sought.

4.7.5 More About MCMC

There is another command called mcmc that uses actual simulators instead of response surfaces. However, this method can be computationally expensive even when the simulator is fast because of the I/O overhead in running the simulator many times. If you have to use a simulation model (and not response surfaces), a remedy for this I/O bottleneck is to compile the model function (if the simulator is simple enough) into PSUADE and use driver = PSUADE_LOCAL to activate that function (this will speed up evaluation by reducing the I/O overhead but it requires inserting some code into PSUADE and re-compilation).

Yet another more advanced MCMC capability in PSUADE is one that can efficiently handle high-dimensional correlated inputs that can be characterized by an ensemble of inputs (as in multi-point statistics). PSUADE takes this ensemble and apply kernel principal component analysis (KPCA) to reduce the input dimension and the selected eigenvectors will be used as inputs for inference. The command to run this option is mcmc_with_dr (or MCMC with dimension reduction). This capability is still in experimental stage.

4.8 Two-Stage Uncertainty and Sensitivity Analysis

This section describes how to use PSUADE to perform a 2-stage (and generally for multi-stage) uncertainty quantification.

Given a simulation model (representing the full system) with 2 uncertain parameters θ_1 and θ_2 such that the model output Y is:

$$Y = F(\theta_1, \theta_2).$$

Suppose that the 'prior' probability distributions for θ_1 and θ_2 have been prescribed. In addition, suppose a focused experiment (representing a sub-system) is available to help 'calibrate' (or constrain) the probability distribution of θ_2 (and θ_2 only). The overall objective is to quantify the uncertainty and parameter sensitivities of the full system given experimental data for the sub-system to constrain θ_2 (while leaving the prior distribution of θ_1 undisturbed).

The steps to achieve the above objective consist of two stages: (1) to perform a statistical inference on the sub-system to calibrate the distribution of θ_2, and (2) to propagate the prior distribution of θ_1 and the 'posterior' distribution of θ_2 through the full system, and compute basic statistics for Y and sensitivities of θ_1 and θ_2.

We use the example given in Examples/HierarchicalUQ as a demonstration.

4.8.1 Stage 1: Statistical Inference

In this example, the experimental data for constraining θ_2 is given in the file called 'expdata2', which has the results of 10 experiments each with experimental uncertainty (that is, standard deviation) of 0.05.

To perform statistical inference (NOTE: the sequence of commands for this stage has been captured in the script called 'psScriptStage1.in'),

1. Prepare the sub-system model for statistical inference for θ_2 by
   - Compiling 'subsystem.c' to be the executable 'subsystem'.
2. Statistical inference requires the use of the actual simulation model or a response surface. To construct a response surface, we prepare a PSUADE input file 'psuadeSubsysRS.in', which has its content (it uses a factorial sampling design of size 10 in the range of $[0.4, 0.6]$):

```
PSUADE
INPUT
   dimension = 1
   variable 1 theta2  =  0.4   0.6
END
OUTPUT
   dimension = 1
```

```
   variable 1 Y
END
METHOD
   sampling = FACT
   num_samples = 10
END
APPLICATION
   driver = ./subsystem
END
ANALYSIS
   printlevel 1
END
END
```

- Run ensemble simulations with 'psuadeSubsysRS.in' and rename the default output file 'psuadeData' to 'subsys.psu'.

3. Perform statistical inference by launching PSUADE in command line mode, load 'subsys.psu', and run rsmcmc in to create a posterior sample for θ_2. The session looks like the following:

```
[Linux] psuade
************************************************************
*      Welcome to PSUADE (version 2.1.0)
************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration (2.1.0)
============================================================
psuade> load subsys.psu
load complete : nSamples = 10
                nInputs  = 2
                nOutputs = 1
psuade> ana_expert
analysis expert mode on
psuade> rsmcmc
.....
Enter MCMC scheme 2
==> Enter the spec file for building likelihood function : expdata2
Say 'no' to the next question.
Output 1
Enter you choice (for response surface type) ? 2
Enter 100000 and 20 to the next 2 questions.
Enter 1 and 0 next to select input 1 and terminate.
```

```
Say 'no' to discrepancy modeling.
.....
MCMC_BF: input   1 value at maximum likelihood = 5.610796e-01
MCMC_BF: input   1 mean    = 5.609426e-01
MCMC_BF: input   1 std dev = 7.416152e-03
.....
MCMC: matlabmcmc2.m file has been created.
.....
psuade> quit
[Linux]
```

4. Upon the completion of statistical inference on the sub-system, two files will have been created: 'MCMCPostSample' and 'matlabmcmc2.m' (or 'scilabmcmc2.m').

   - Launch Matlab and visualize the posterior distribution of θ_2.
   - 'MCMCPostSample' will be used in the next stage.

4.8.2 Stage 2: Uncertainty Propagation and Sensitivity Analysis

To perform uncertainty and sensitivity analysis, we will draw a large sample from both the prior distribution of θ_1 and the posterior distribution of θ_2.

1. Create a large sample for $theta_1$ by running psuade psuadeGenSample1.in (generate sample only and no simulation runs) and rename 'psuadeData' to 'prior1.psu'.

2. Convert 'MCMCPostSample' (the posterior sample from Stage 1) to PSUADE data format by launching PSUADE in command line mode and:

   - Read in the posterior sample (use 'iread MCMCPostSample', and
   - Write the sample into a file using PSUADE format (use 'write posterior2.psu').

3. Now we have two distribution files: 'prior1.psu' and 'posterior2.psu'. The next step is to combine the two distributions into a single large sample consisting of two parameters. This can be accomplishing by using the rand_draw2' command (say, to create a large sample of size $100k$).

```
[Linux] psuade
************************************************************
*      Welcome to PSUADE (version 2.1.0)
************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration (2.1.0)
============================================================
psuade> rand_draw2
```

```
Enter name of the first file : prior1.psu
Enter name of the second file : posterior2.psu
Size of the sample to be drawn : (1-2000000) 100000
Store random sample to : (filename) largeSample.psu
psuade> quit
[Linux]
```

4. Prepare the original 2-parameter full-system simulation model ('fullsystem.c') by compiling it (to become 'fullsystem').

5. Propagate the 2-parameter sample through the simulator by:

   - Setting the driver field in 'largeSample.psu' to be 'fullsystem',
   - Running PSUADE on 'largeSample.psu'. After the runs have been completed, rename the simulation output file 'psuadeData' back to 'largeSample.psu'.

6. Finally, launch PSUADE to compute uncertainties and sensitivities:

   - Run ua for uncertainty analysis, and
   - Run me for sensitivity analysis.

```
[Linux] psuade
**************************************************************
*      Welcome to PSUADE (version 2.1.0)
**************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration (2.1.0)
==============================================================
psuade> load largeSample.psu
psuade> ua
.....
psuade> me
.....
psuade> quit
[Linux]
```

7. In addition, you can compare the current uncertainty result with the one without using experimental data to assess how the use of experimental data affects the output uncertainty. The script 'psScriptOneStage.in' in PSUADE's Examples/HierarchicalUQ directory runs this analysis automatically.

4.9 Numerical Optimization

Numerical optimization, strictly speaking, does not belong to the realm of uncertainty quantification. However, since many response surface methods require optimization of their parameters, optimal design methods require optimizing some prescribed metrics, and optimization under uncertainty relates to UQ, therefore a number of optimization algorithms have been incorporated into PSUADE and can be used for user's model optimization needs.

There are several considerations in setting up numerical optimization. For example,

1. Is this a global optimization problem with many possible local minima? If so, should multi-start optimization be used?
2. What are the optimization variable types (continuous, discrete, categorical)?
3. Do constraints exist? If so, what types?
4. Does the model produce derivatives?
5. What should the initial guess(es) be?
6. Which optimizer to use?
7. What is the termination criterion?

In this section several several scenarios of how to use optimization and optimization under uncertainty are described.

4.9.1 Optimization with Continuous Variables

Let the function for numerical optimization be the two-dimensional Rosenbrock function:

$$Y = 100(X_2 - X_1^2)^2 + (1 - X_1)^2, \quad X_i \in [-2, 2].$$

The PSUADE input file for numerical optimization can be constructed as follow (here bobyqa is a public domain software developed by Michael Powell):

```
PSUADE
INPUT
   dimension = 6
   variable  1 X1  =   -2.0   2.0
   variable  2 X2  =   -2.0   2.0
   variable  3 X3  =   -2.0   2.0
   variable  4 X4  =   -2.0   2.0
   variable  5 X5  =   -2.0   2.0
   variable  6 X6  =   -2.0   2.0
END
OUTPUT
```

```
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = LPTAU
   num_samples = 5
END
APPLICATION
   driver = ./simulator
   opt_driver = ./simulator
END
ANALYSIS
   optimization method = bobyqa
   optimization num_starts = 3
   optimization max_feval = 10000
   optimization tolerance = 1.0e-4
   optimization print_level = 2
   optimization num_fmin = 1
END
END
```

The METHOD section specifies how initial guesses are to be generated. For example, if one random initial guess is to be used, the sampling method should be set to MC or LPTAU (a quasi-random sampling method) and the sample size (num_samples in the METHOD section) may be set to any positive integer (other sampling methods can be used, or users can provide initial guesses by filling the PSUADE_IO section themselves. For multi-start optimization, the number of starts can be set by using the num_starts keyword (or, equivalently, the obsolete keyword num_local_minima) in the ANALYSIS section. num_starts needs to be larger or equal to num_samples, since otherwise there will not be enough initial guesses. If, on the other hand, num_samples is larger than num_starts, PSUADE will select num_starts initial points from num_samples points that have the lowest objective function values.

In the example above, the analysis begins with the evaluation of the 5 quasi-random sample points. Subsequently, 3 (since num_starts = 3 sample points with the lowest output values are selected as the starting points for a multi-start optimization. Since it is a bound-constrained optimization, a suitable optimizer is bobyqa. The termination criteria are prescribed via the maximum number of function evaluations (10000 in this case) or a termination tolerance ($1e-4$). driver points to an executable called simulator (which is used to evaluate the initial quasi-random sample; and, if not provided, the selection of random initial guesses will be drawn randomly from the initial sample); and opt_driver points to an executable used in the actual optimization.

The above example is located in the Examples/Optimization/Rosenbrock directory. Simply compile the simulator.c file and then run 'psuade psuadeBobyqa.in' to see optimization in action.

There are other advanced features in optimization such as avoiding repeated function evaluations (this is very useful for restart in the case when the function evaluation is expensive).

There are other continuous optimization algorithms in PSUADE for handling different types of optimization:

cobyla: Nonlinear inequality-constrained continuous optimization (by M. Powell);

newuoa: Nonlinear optimization for unbounded parameter ranges (by M. Powell);

lincoa: Linear equality-constrained continuous optimization (by M. Powell);

minpack: Nonlinear optimization for least-squares problems with bounded constraints (by J. More, et al.);

L-BFGS-B-C: Bounded-constrained nonlinear optimization that utilizes derivatives (by Byrd, et al.);

SCE: Bound-constrained nonlinear optimization based on pattern search (by Duan), which can also handles mixed parameter types;

SM/MM: 2-level bound-constrained nonlinear optimization (by Ciaurri).

Usages of these different types of optimization are described in the examples in the Examples/Optimization directory.

4.9.2 Optimization With Mixed Types of Variables

There are two methods within PSUADE that can handle mixed types of variables: SCE and NOMAD. There is an example in the 'Examples/Optimization/MixedInteger' directory. Specifically, the scripts 'psScriptSCE3.in' and 'psScriptNomad3.in' capture how to run these 2 methods on a 30-parameter problem with 15 continuous and 15 discrete variables. Read the 'README' file for more information on how to set up and run these optimizations.

4.9.3 Optimization With Linear Constraints

Most of the optimization algorithms in PSUADE are for bound-constrained optimization, but it also include a linearly-constrained algorithm (by Michael Powell).

For example, given the following optimization problem:

$$\min_{X_1,X_2} [2X_1^2 + X_2^2]$$

with the equality constraint that $X_1 + X_2 = -1$ and also that $X_1, X_2 \in [-2, 2]$. The PSUADE input file for this scenario is:

```
PSUADE
INPUT
   dimension = 2
   variable  1 X1  =   -2.0   2.0
   variable  2 X2  =   -2.0   2.0
END
OUTPUT
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = MC
   num_samples = 1
END
APPLICATION
   driver = ./simulator
   opt_driver = ./simulator
END
ANALYSIS
   optimization method = lincoa
   optimization max_feval = 10000
   optimization tolerance = 1.0e-4
   optimization print_level = 2
END
END
```

In addition, another file, called 'psuade_lincoa_constraints' (a default file name that PSUADE looks for when LINCOA is called) needs to be prepared to set up the constraints. This file has the following content for the constraint given above:

```
PSUADE_BEGIN
1 2
1 1 1 -1
PSUADE_END
```

The keywords 'PSUADE_BEGIN' and 'PSUADE_END' are for internal checking purposes. The second line ('1 2') means that there is one constraint and it involves 2 variables. The third line specifies the constraint: '1' means the first constraint, the next '1 1' are the coefficients for X_1 and X_2 (that is, $X_1 + X_2$), and the last '-1' is the right hand side. This example can be found in the directory 'Optimiztion/LincoaTest'. Read the 'README' file for more information on how to set up and run these optimizations.

4.9.4 Optimization With Inequality Constraints

PSUADE has an optimization algorithm for nonlinear inequality constraints: COBYLA (also

by Michael Powell). The way to prescribe inequality constraints is different from that of LINCOA, in that the inequality constraint are embedded in the simulation driver (that is, the simulation model is expected to produce not only the objective function value, but also the values of all the constraints. For example, if one of the constraint is $X_1 + X_2 - 1 \geq 0$, it should first be rewritten to become $1 - X_1 - X_2 \leq 0$ (that is, all variables and constants are on the left hand side of the $\leq$ sign and 0 on the right hand side) so that, for example, if $X_1 = 2$ and $X_2 = 1$, then the simulator should produce -2 $(1 - 2 - 1)$ when evaluating this constraint. This example can be found in the directory 'Optimiztion/CobylaTest'. Read the 'README' file for more information on how to set up and run these optimizations. Also, read the 'simulator.c' file for information on how to set up the constraints.

4.9.5 Other Optimization Algorithms

For unconstrained optimization (with BOBYQA), go to directory 'Examples/Optimization/Rosenbrock' and run 'psuadeBobyqa.in'. For unconstrained optimization (with NEWUOA), go to the same directory and run 'psuadeNewuoa.in'. Although NEWUOA solves unconstrained problems, constraints can be added by adding a penalty function to the objective function for violating them. In addition, the L-BFGS algorithm is also available in scenarios when derivatives (or approximate derivatives using finite difference) are available.

4.9.6 Response Surface-based Optimization

Suppose a training sample for response surfaces has been prepared, and the objective is to perform parameter optimization using response surfaces. If the training sample output is the optimization objective function, PSUADE can readily handle it by replacing the opt_driver with the training sample file (in PSUADE format). However, if the optimization problem is:

$$\min_{\omega} [\hat{F}(X, \omega) - D(X)]^2$$

where $\hat{F}(X, \omega)$ is the response surface representation of the true function $F(X, \omega)$ where X and ω are design and optimization parameters, respectively (that is, experimental data are collected at different design settings X, and the optimal values of ω are to be searched by the selected optimization method). Also, $D(X)$ represents experimental data to be supplied by users. This scenario is not unlike that of statistical inference, except that it is deterministic in nature (that is, searching for optimal parameter values rather than some posterior distributions).

There are two ways to handle this scenario using PSUADE:

1. Have PSUADE create a response surface interpolation code (in C++ or Python using the rs_codegen and rscheck commands) and incorporate experimental data into the code to compute the squared-error quantity); or

2. Use the response surface-based least-squares capability in PSUADE.

To set up option (2), the mechanism is to use the PSUADE_LOCAL keyword for opt_driver, which, during optimization, causes PSUADE to construct response surfaces and compute the sum of squared errors internally. In this context, during the initialization, PSUADE will request users to supply both the training sample and the data set, as well as which response surface type to use. An example is given in the Examples/Optimization/RSBasedOpt directory.

4.9.7 Two-Level Optimization

PSUADE also supports two-level optimization using two simulation models: a low-fidelity model and a computationally expensive high-fidelity model. The optimization methods for this type are the Space-Mapping (SM) and the Manifold-Mapping (MM) methods developed by D. Ciaurri. To set up these methods, opt_driver is to point to the high-fidelity model, while aux_opt_driver is to point to the low-fidelity model (which may also be a training sample for building a response surface). Also, the optimization method is to be set to sm or mm. In addition, an optional target file is to be prepared to provide the optimizers with the expected outputs (i.e. the experimental data from which the optimal parameter values are to be recovered). A user may incorporate the experimental data inside the simulation model so that this target file is not needed. Examples are given in Examples/Optimization/Poisson directory.

4.9.8 Optimization Under Uncertainty (OUU)

Let the simulation model be represented by

$$Y = F(X, U, \omega, \theta)$$

which is characterized by four types of variables:

1. Design/Decision variables X are the optimization variables that will be tuned to optimize some objective function,

2. Recourse/Operational variables U are scenario variables which can be tuned in a given system operating under different conditions (different values of ω and θ),

3. Discrete/Scenario variables ω have an enumerable set of states (called scenarios) such that each state is associated with a probability (and sum of probabilities for all possible states is equal to 1), and

4. Continuous uncertain variables θ are associated with a joint probability density function.

In the context of the formulation above, two different types of OUU are possible:

1. Single-stage OUU: this formulation requires no recourse variables U so that it is solving the following problem:

$$\min_X \Phi_{\omega,\theta}(F(X,\omega,\theta))$$

where $\Phi_{\omega,\theta}$, the objective function, is some statistical quantity (e.g. mean) from running $F(X,\omega,\theta)$ on a sample drawn from the probability distributions of ω and θ (either ω or θ can be an empty set but not both).

2. Two-stage OUU: this formulation requires recourse variables for inner optimization but the recourse variables may be defined implicitly (not needing to be declared). This type of OUU solves the following problem:

$$\min_X[\Phi_{\omega,\theta}(\min_U F(X,U,\omega,\theta))].$$

In essence, the first type is analogous to typical numerical optimization except that, in the presence of uncertainties, the objective function to be optimized is some average of the deterministic objective function. The second type, on the other hand, has the goal to obtain an optimized design (with respect to X) but this design can dynamically adapt to different operational conditions (U) to optimize its performance. Thus, the 2-stage method involves an inner-outer optimization loop. Users can optionally provide the function F in opt_driver for which PSUADE will wrap around it with two layers of the BOBYQA optimizer (to optimize with respect to X in the outer loop and U in the inner loop), or provide the inner optimization ($\min_U F(X,U,\omega,\theta)$) for which PSUADE will just wrap around it with one layer of the BOBYQA optimer (to optimize with respect to X).

An example is given in Examples/Optimization/OUU/Problem2. The PSUADE input file is 'ouu_opt_driver.in':

```
PSUADE
INPUT
   dimension = 12
   variable   1 D1  = -5     5.0
   variable   2 D2  = -5     5.0
   variable   3 D3  = -5     5.0
   variable   4 D4  = -5     5.0
   variable   5 X1  = -10    10
   variable   6 X2  = -10    10
   variable   7 X3  = -10    10
   variable   8 X4  = -10    10
   variable   9 W1  = -5    5
   variable  10 W2  = -5    5
   variable  11 W3  = -5    5
   variable  12 W4  = -5    5
END
OUTPUT
```

```
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = MC
   num_samples = 1
   randomize
   random_seed = 41491431
END
APPLICATION
   opt_driver = optdriver
END
ANALYSIS
   optimization method = ouu
   opt_expert
END
END
```

In this example, the first 4 variables are design variables (type 1), the next 4 are recourse variables (type 2), and the last 4 are continuous uncertain parameters (type 4). The optimization driver should be created by compiling the 'optdriver.c' file. To run OUU, simply do:

```
[Linux] psuade ouu_optdriver.in
```

and you will be prompted with a few questions asking for the number of each of the 4 types of variables $(4, 4, 0, 4)$, the type of each declared variables, the choice of objective function (enter '1' - the expected value), the choice of the sample source for the type 4 variables (PSUADE-generated), option to use response surface for estimating the expected value (enter 'n'), the sampling method used for type 4 (enter Latin hypercube), the sample size (enter 200), to choose 'your own inner optimizer', and 'n' to the rest of the questions. You will then see the OUU optimization in action giving the best design variable (X - variable $1 - 4$) settings at the end.

The newest additions of the OUU suite are:

- OUU with linear constraints (Powell's LINCOA)
- OUU with nonlinear inequality constraints (Powell's COBYLA)
- OUU with no bound constraints (Powell's NEWUOA)
- OUU with bound constraints and derivatives (LBFGS)
- OUU with both continuous and integer parameters (NOMAD)
- Different objective functions: output mean, linear combination of output mean and standard deviation, and value-at-risk (VaR).

4.10 Experimental Design

Although the UQ discipline has been around for over two decades, many application scientists are still skeptical of its usefulness. They often question the value added by expending large computational resources for UQ analysis. They generally agree that model predictions are plagued with uncertainties, but question how quantifying these uncertainties can lend much help in their causes.

The UQ community has realized that one important use of UQ is to guide design the next experimental campaign. Physical experiments are expensive, and it is in the interest of application scientists to design experiments that maximize the information needed to better understand the physical phenomenon at hand and thus to enable a better modeling strategy. In similar manner, when simulations are expensive, 'smart' design of computer experiments can be useful in reducing the overall computational cost. Experimental design (or design of experiment, DoE) is a class of techniques for intelligently proposing new physical or computer experiments. In the following we describe PSUADE's DoE capabilities separately for static/adaptive design of computer experiments and optimal design for physical experiments.

4.10.1 Optimal Design of Experiments for Simulation Campaigns

The purpose of optimal design of experiments for simulation is to improve the quality of response surfaces using as few simulations as possible. PSUADE currently has capabilities for proposing experimental designs that optimize certain useful criteria such as maximizing space-fillingness or minimizing prediction uncertainties. For the latter, capabilities are available for single-batch and sequential experimental designs:

A. Single-batch space-filling optimal design

PSUADE provides a capability to search a user-provided candidate set of possible sample points and extract a subset that maximizes the minimum distance between all selected points (thus called MMD - maxi-min distance method). Both weighted and unweighted MMD schemes are available.

To give a more concrete description, let C_N be the set of N design candidate '$\{X_i\}_{i=1}^N$ where $X_i \in \Re^m$ and m is the number of parameters in the model. Let n be the number of points to be selected from C_N, and let S_n denote any subset of C_N with n points. Finally, let the distance between any two points X_i and X_j be given by:

$$D(X_i, X_j) = \sqrt{\sum_{k=1}^{m} w_i w_j (X_{ik} - X_{jk})^2}.$$

where w_i and w_j are weights for the weighted version of MMD. Then, the utility function for the MMD method can be defined as:

$$S_n^* = \text{argmax}_{S_n \in C_N} \{\min_{i,j \in S_n, i \neq j} D(X_i, X_j)\}.$$

The MMD method is performed in the command line mode using the following steps:

1. Prepare a candidate set (or a sample) that can be loaded to PSUADE's local memory using, for example, load or read_std commands.
2. Run 'odoe_mmd'
   - Select unweighted or weighted option (if weighted option is selected, one of the sample outputs will be used as weights).
   - Select one of the 3 search methods:
     - Brute force search (it finds the optimal set, but can be very expensive).
     - Global optimization search (fast but it may not find the optimal set due to many local minima),
     - Hybrid search: use global optimization to find a good initial guess to speed up the subsequent brute force search. This is recommended because it finds the optimal set and it can be much faster than option 1.
   - Specify the size of the optimal set.
   - If hybrid search has been selected, you will have to specify how many multi-starts and how many iterations to use in the global optimization search. You may have to play around with these settings to get a good overall speedup. A suggestion is 50 multi-starts and 2000 iterations.
   - To see in more detail the search in action, create in another terminal screen in your run directory a file called 'psuade_print' (e.g. use touch psuade_print). This tells PSUADE to send all intermediate search results (instead of only the intermediate best search results) to the screen. You can turn this off during search by deleting this file.
   - Wait for the final result (To see the number of multi-starts and iterations are good enough, observe - in 'psuade_print' mode - the progression of the search combinations; e.g. for choosing 4 out of 100 candidates, you will observe on the screen that combinations will run from '1 2 3 4' to '97 98 99 100'. If this progression is too slow, you may be to re-run to get a better initial guess.)
   - At the end of the run, a file called 'matlabodoe_mmd.m' will have been created that has the final selection.

Examples (unweighted and weighted MMD) are given in the Examples/ODoE/Example1 directory.

B. Optimal design based on GP prediction uncertainties

This method - the mini-max variance or MMV method - first builds a GP-based response surface, then for each subset of k points from the candidate set, evaluates the maximum GP prediction uncertainties of the evaluation set (which is the same as the candidate set), and selects the final subset that minimizes the maximum GP prediction uncertainty. Note that this method can be used in a sequential fashion to add more sample points.

The steps for running this method is as follow:

1. Create an initial 'small' sample drawn uniformly from the parameter space (any space-filling design) and run simulations (this sample is used as a training sample for a GP response surface to be used to compute GP prediction uncertainties). Alternatively, if you can provide GP hyperparameters, there is no need to create the inital training sample.
2. Start PSUADE in command line mode:
   - Load the simulated sample;
   - Run the odoe_mmv command
     - Enter your candidate set or have PSUADE create one;
     - Enter the number of points to be selected (k);
     - Select search algorithm (e.g. enter '2' to perform global optimization);
     - Enter desired search algorithm options;
     - Enter training set or hyperparameters;
     - At the end, the selected set will be displayed.

PSUADE also provides a slightly different variance-based method - MAV, which denotes minimum average variance. That is, instead of finding the set that minimizes the maximum GP prediction variance, this method finds the set that minimize the average GP prediction over the evaluation set. An example for MAV is given in the Examples/ODoE/Example1 directory.

C. Sample Design based on Spatially Adaptive Refinement

This method is similar to odoe_mmv in that it selects candidates based on response surface prediction errors. The major difference is that, instead of finding a subset from a user-provided candidate set, it explores the parameter space and identifies 'good' points in that space. In addition, this method combines the ideas of space-fillingness and prediction errors via spatial decomposition and bisection. This method is best used with bootstrapped MARS response surface for detecting 'nonsmooth' behavior in the parameter space.

This method begins with a space-filling (METIS) sample with each sample point being placed into a sub-region or cell in the parameter space. This initial sample is evaluated using a stochastic response surface (e.g. bootstrapped MARS) to assess prediction uncertainties over different sub-regions in the parameter space. The sub-regions with the largest prediction uncertainties are to be selected for refinement (via bisection of the sub-regions) and new points are created for the new refined sub-regions. The enlarged sample is evaluated and further refined until the response surface prediction errors over the parameter space are acceptable or the number of new sample points have reached a desired size.

The steps for running this adaptive response surface using the command line mode is as follow:

1. Before running sampling refinement, first create an initial 'small' sample drawn uniformly from the parameter space (preferably using the METIS sampling method) and run simulations;
2. Start PSUADE in command line mode:
   - Load the simulated sample
   - Run the arefine_metis command (there are a few others similar commands available in PSUADE) during which you will be asked how many additional sample points to generate;
   - Store the resident sample (use write), in which the new sample points are appended at the end of the PSUADE_IO section.
3. Run simulations on the new points in the expanded sample.
4. Load the expanded simulated sample, start PSUADE in command line mode, and run the arefine_metis again for another iteration.

The procedure described above can also be run in batch mode. An example is given in Examples/Adaptive/AdaptiveRS. Specifically, running PSUADE with 'psuadeARSM1.in' launches 7 sequential refinements, beginning initially with 20 sample points and adding 10 additional points at each refinement to reach a total of 90 sample points at the end. The key features in activating adaptive refinements are in the METHOD and ANALYSIS sections:

```
METHOD
   sampling = METIS
   num_samples = 20
   num_refinements = 7
   refinement_size = 10
   refinement_type = adaptive
   random_seed = 541541191
END
APPLICATION
   driver = ./simulator
END
ANALYSIS
   analyzer method = ARSM
   analyzer rstype  = MARSBag
END
```

where 'ARSM' tells PSUADE to perform iterative refinements with specific instructions given by num_refinements, refinement_size, and refinement_type (which can be uniform or adaptive). Any response surface type that provides prediction uncertainties (e.g. radial basis function does not provide prediction uncertainties) may be used, but we recommend 'MARSBag' (bootstrapped MARS). driver needs to point to the simulation

model in order to expedite the automation of the entire process. To run this script, simply do:

```
[Linux] psuade psuadeARSM1.in
```

4.10.2 Optimal Design for Experimental Campaigns

PSUADE has another set of capabilities especially tailored for optimal (physical) experimental design. For this class of method, prediction errors are induced by the uncertain parameters in the simulation models, and not due to response surface uncertainties in the last section.

There are two objectives for using such designs: (1) for improving simulation models (reducing uncertainties of the uncertain parameters, and (2) for improving prediction uncertainties. The former produces D-, A-, and E-optimal designs; and the latter G- and I-optimal designs.

Two classes of methods are provided by PSUADE: one based on Bayesian inference and the other based on the Fisher information criteria, both of which are for general nonlinear models. The Bayesian approach is in general slower the Fisher approach. It is recommended if: (1) the size of the selected set is larger than the number of uncertain parameters, and (2) users are able to obtain better estimates (standard deviations) of the candidate design set than are predicted by the simulations. The Fisher approach is recommended otherwise.

To use the current optimal experimental design capbilities, users need to provide: (1) a simulation model with 2 sets of parameters - design (or control) and uncertain parameters; (2) initial ('prior') distributions for the uncertain parameters; and (3) a set of 'candidate' design points (in terms of the design or control parameters). Users also need to provide a training sample for constructing high-quality response surfaces. For G- and I-optimal designs, an evaluation set is also needed.

The steps to select 'optimal' designs from the candidate set are:

1. Create a training sample for response surface construction: Given the ranges for the design and uncertain parameters, create a space-filling design, run simulations, and store the sample and results into a file in PSUADE format.

2. Create a large sample drawn from the 'prior' distribution of the uncertain parameter and store it in a file (in PSUADE's iwrite format).

3. Create a candidate set and store it in a file (in PSUADE's iwrite format).

4. For the Bayesian approach, if users have good estimates for the experimental results of the candidate set (at this point no experiment has been conducted so normally no 'good' estimates are available except those predicted by the response surface using the prior sample, but users may have good estimates from other sources and these good estimates should be appended to the candidate set file.

5. For G- and I-optimal designs, create an evaluation set and store it in a file (in PSUADE's iwrite format). This set is used to evaluate how good a G- or I-optimal selected design

is, and it can be the same as the candidate set (but not recommended). This evaluation set should cover the entire design space but it should not be too large because it may impact computational cost significantly.

6. Start PSUADE in command line mode:

   - Run the odoeu_boptn or odoeu_foptn command and enter information (e.g. traninng sample file, response surface type, size of the selected set, prior sample) as requested;
   - Wait until the search is complete. At the end, PSUADE displays the final 'optimal' set along with the different optimality measures. (Note: due to the use of a global optimizer in the search, the final result may be sub-optimal). In case the search takes too long and you are satisfied with some intermediate results, you can create a 'psuade_stop' file (using, e.g. touch psuade_stop) in the run directory to force the search to be terminated in a graceful manner.

To gain experience with a concrete example, go to Examples/ODoE/Example3 (Examples/ODoE/Example4) and follow instructions.

4.11 Miscellaneous Capabilities

This section covers other PSUADE capabilities that users may find helpful in their UQ studies.

4.11.1 Sobol' Analysis for Correlated Inputs

Sometimes the model uncertain inputs may have constraints that cause the inputs to be 'correlated'. Correlations may be in the form of inequality constraints or multivariate normal distributions, or correlation may be embedded in a multivariate ensemble. This section describes how to handle these cases. Examples are given in the Examples/CorrelatedInps directory.

A. Correlation in Joint-Normal Distributions

In many occasions, correlations between input parameters may be in the form of multivariate normal distributions with non-diagonal covariance matrices. For example, the following INPUT section (in PSUADE format) specifies a multivariate normal distribution with a correlation of 0.7 between input 2 and 3.

```
INPUT
   dimension = 3
   variable   1 X1   = -5  5
   variable   2 X2   = -5  5
   variable   3 X3   = -5  5
   PDF 1 N 0 1
   PDF 2 N 0 1
```

```
   PDF 3 N 0 1
   COR 1 2 0.7
END
```

With this prescription, the response surface-based Sobol' first-order sensitivity method in PSUADE will draw samples from this prescription during the calculation of sensitivity indices.

B. Parameter Correlation Embedded in Large Ensembles

Oftentimes, parameter correlations may not be in any of the standard forms such as multivariate normal. Instead, the only way that correlation is expressed is in the form of a large multivariate sample (such as the posterior sample created during statistical inferences). PSUADE provides a method to compute first-order sensitivity indices from such a multivariate sample. The first step in this analysis is to evaluate this large sample using a response surface built from some training sample. Subsequently, this large sample and the corresponding sample output from response surface evaluation is to be directly fed into the main effect analyzer in PSUADE, which will output the parameter sensitivity indices.

C. Parameter Correlation in the Form of Inequality Constraints

Oftentimes, parameter correlations may be in the form of inequality constraints imposed on the parameters (e.g. $X1 + X2 < 1$). The mechanism provided in PSUADE to prescribe inequality constraint is to add a line in the ANALYSIS section of a psuade input file:

```
   analyzer rs_constraint = constrData indexFile Lbnd Ubnd
```

where 'constrData' is a training sample for constructing a response surface to compute constraint values. 'indexFile' is a file that specifies which variables are involved in the constraint (e.g. $X1$ and $X2$ for the constraint $X1 + X2 < 1$). During calculation of sensitivity indices, each sample point will be evaluated using the constraint response surface and tested to see if it lies inside the prescribed lower and upper bound ('Lbnd' and 'Ubnd' above). Multiple rs_constraint statements can be specified if there are multiple constraints.

D. A Mixture of the Above Three Scenarios

PSUADE also supports sensitivity studies for models that have all 3 types of constraints. The only limitation is that the parameters that have multivariate normal correlations should not overlap with those that use large ensemble to express the parameter correlations.

4.11.2 Search for Optimally Space-filling Designs

When selecting a sampling method for building response surfaces, the desirable characteristics of the sampling method is 'space-fillingness' (unless adaptive response surface analysis is to be employed). Most of the sampling methods are not strictly speaking optimally 'space-filling'. For example, the Monte Carlo and Latin hypercube samples may sometimes be rather 'non-space-filling' (although the optimal Latin hypercube sampling method has been proposed). PSUADE provides a function (the sopt_mmd command in interactive mode), which, when presented with a sample, attempts to optimize its 'space-fillingness' by performing a maxi-min distance optimization.

4.11.3 Sample Quality Check

To quantify the space-fillingness of a sample, PSUADE provides the sqc command, which computes a few metrics such as minimum distance between all pair of sample points (the larger the better), average minimum distance for all sample points (the larger the better) and minimum distance between the sample points and the corners (the smaller the better).

4.11.4 Use of Response Surface Index File

Suppose you have a training sample for 5 inputs that have been identified for your simulation model, but then for analysis (uncertainty/sensitivity analysis or numerical optimization) you would like to freeze the first input to a fixed value (e.g. 0.7). Can this set up within PSUADE?

The answer is yes: use a response surface index file to select fixed and varying inputs. The index file has the following format:

```
5
1   0    0.5
2   2
3   3
4   4
5   5
```

where the first line has the total number of inputs, and then next 5 lines specifying how each of the 5 inputs should be treated. If an input is not to be frozen, then the line should have 2 identical numbers. However, if an input is to be frozen, then the second field of the corresponding line should be set to 0, and the third value is the fixed value to set to.

An example is given in the Examples/RSTests/RSIndex directory. In the example, there are a total of 4 inputs for which a sample is created for building a 4-dimensional response surface, but the model is to be analyzed using main effect analysis with the 4-th input fixed at 2.0. To do so, an index file is to be created following the example given above. Then a new line is to be added to the ANALYSIS section of the training sample file to specify the index file.

4.11.5 Adaptive Sampling Refinement

While space-filling samples are useful for building response surfaces in general, some functions have local variations that require finer sampling resolution to ensure accurate characterization of the response surfaces. Adaptive sampling refinement for building response surfaces in batch mode has been described earlier in this document. This section covers how the similar refinement procedure can be performed manually via the command line model. An example is given in the Adaptive/AdaptiveSampling sub-directory of the Examples directory. In the following we illustrate how to perform individual refinement steps:

1. First of all, an initial training sample is needed to build the first response surface for the purpose of estimating prediction errors. The recommended sampling method is METIS, which is a space-filling method based on spatial decomposition.

2. Then, the generated sample is fed into the arefine_metis command to create new points. Store the appended sample.

3. Afterward, the new points are to be evaluated by running the appended sample (PSUADE recognizes the unevaluated new points and calls the simulator.

4. If further refinement is needed, repeat Step (2) and (3). (Important: do not change or remove the psuadeMetisInfo file during the iterative procedure, since it contains pertinent information for refinement.)

5. The sample distribution of the final appended sample can be viewed using the iplot2 command.

4.11.6 Reliability Analysis

Reliability analysis (RA) concerns the identification and quantification of the region in the parameter space where the simulation output falls outside a prescribed 'safe' region. It can be considered a special case of adaptive sampling refinement since the identification of the threshold boundary is analogous to locating the 'interesting' regions in adaptive sampling..

RA methods in PSUADE fall into two classes: simulation-based and response surface-based methods. Both of them use adaptive sampling to 'zoom' into the threshold boundary, which is a $(m-1)$-dimensional surface for a m-dimensional parameter space. As such, the methods are not suitable for large m (in general, $m > 5$ may be considered high-dimensional since the computational cost will be too high to quantify a 5-dimensional surface, no matter whether it is simulation-based or response surface-based). In the following we discuss a few ways to run the RA method.

A. Simulation-based RA Method: To perform simulation-based RA, the first thing is to prepare a PSUADE input file with the driver field in the APPLICATION section pointing to the simulation executable, and the analyzer method field in the ANALYSIS section set to REL (meaning reliability analysis). In addition, in the METHOD section, a few things may need to be specified:

- Sampling method: which is defaulted to be METIS (this method divides the parameter space into regions or agglomerates from which local refinements are performed) and should not be changed by users.
- Initial sample size: the num_samples field in the METHOD section.
- Number of refinements: the num_refinements field in the METHOD section that controls how many iterations to take.
- Reliability tolerance: the analyzer threshold field in the ANALYSIS section that controls when (the precision of the reliability value) the iterative procedure should be terminated (so there are two termination criteria: number of refinements and tolerance).

B. RS-based RA Method: The setup for RS-based is similar to that of the simulation-based approach with one exception: that the driver field in the APPLICATION section, instead of pointing to a simulation executable, now points to a PSUADE data file that contains an pre-evaluated training sample (Note that the response surface type is to be specified in the training sample).

C. Command Line Mode Simulation-based RA Method: While the two examples given above use batch mode to compute reliability, reliability quantification using adaptive refinement is also available in the command line mode. Specifically, the command is called arel_metis, which can be called multiple times simulating the iterative adaptive sampling refinement procedure. An example is given in Examples/ReliabilityTests/Test2.

4.11.7 Python Plotting Tools

PSUADE creates plotting file mostly in Matlab and Scilab. As some users may not have access to both, a collection of Python plotting tools are also made available to users (although they are less well-developed). For example, the posterior parameter distributions from statistical inference can be displayed by the 'view_mcmcposteriors.py' script. Others are for displaying one- and two-dimensional response surface plots, two-dimensional scatter plots, histogramming, etc. The Python scripts are located in the Python directory.

4.11.8 Stand-alone Response Surface Interpolators

Previously we describe how to build and validate response surface. Once a suitable response surface has been constructed, many users express interest in getting the actual stand-alone interpolation code to be inserted into their own code. Users can access this option via turning on issuing rs_codegen in command line mode *before* running rscheck. When the operation is complete, a file called psuade_rs.info containing the 'C' or 'C++' interpolation code will have been created (for most response surface types) in which there are instructions on how to extract the interpolation code. In addition, another file called 'psuade_rs.py' containing the Python interpolation code will also have been created for Python users.

4.11.9 The Use of Integer Variables

PSUADE in general does not support integer variables. Nevertheless, user can generate integer samples in the following way:

1. Create the sample as a continuous variable, and then use command line interpreter commands to turn the floating point numbers into integers (e.g. use iceil, ifloor, or iround commands).

2. Set the variable distribution type to be 'S' (sample) and provide a random sample of all integers so that when PSUADE draws a sample, the variable values will be integers.

3. Some optimizers (e.g. SCE and NOMAD) provides a mechanism to set some variables to be integer variables. Examples are given in the Optimization/MixedInteger directory.

4.11.10 Sample Editing

Sometimes a user may want to change some sample inputs or outputs, so PSUADE provides a suite of commands in the command line interpreter for sample editing. Issue a help edit to see the list of commands. Specifically, users can:

- Create a new sample input or output (iadd, iadd1, oadd, oadd1).
- Delete sample inputs or outputs (idelete, odelete).
- Filter out sample points not in some prescribed input or output range (ifilter, ofilter).
- Change the orderings of the sample inputs or outputs (ishuffle or oshuffle).
- Sort the sample based on values of a selected input or output (isort, osort).
- Change an individual sample input or output (imodify, omodify).
- Rescale an input or output (iscale, oscale, inormalize, onormalize).
- Turning an input into integers by rounding (iround), truncation (ifloor), or ceiling (iceil).
- Transform (e.g. logarithmic) an input or output (iop, oop).
- Create a new input or output from combination of 2 selected inputs (iop, oop), or create a new output from combination of 2 outputs (ioop.
- Combine duplicate sample points by averaging or other operations.

4.11.11 Use of Configuration Table

PSUADE tries to provide flexibility and many options for users to choose in a UQ analysis. This flexibility may sometimes cause nuisances in performing certain analysis. For example, in Sobol' sensitivity analysis, PSUADE allows users to select sample sizes in the analysis interactively. For some methods such as the bootstrapped Sobol' analysis, PSUADE prompts the users for sample sizes for every bootstrapped instantiation, which is cumbersome. The use of configuration file alleviates this nuisance by declaring the special options implicitly thus freeing users from having to repeatedly enter the required information. This configuration feature is usually not needed but it may be useful sometimes. You can learn about this by trying the 'config' commands.

4.11.12 High-Dimensional Input Reduction

PSUADE has a statistical inference capability that couples with an input dimension reduction method. Since MCMC performs poorly for high-dimensional input space, it may be necessary to reduce the input dimension to a manageable size. The dimension reduction method used in this context is the kernel principal component analysis (KPCA) technique that works on a high-dimensional input distribution represented by an ensemble of input samples. An example is given in the DimReduction/KPCA directory.

5 UQ Methodology

So far we have described many tools and techniques for performing uncertainty assessment, sensitivity analysis, optimization, and the like, in practice developing and following a systematic UQ process comprising a well-thought-out plan of actions to achieve a UQ objective is as important. Without a good UQ plan, subsequent UQ results may not hold up under scrutiny upon reviews by subject matter experts (for example, some prescribed input uncertainties are later found to be not physically realistic and thus not defensible), and thus the computational budget expended on the study may be wasted. In the following section we describe a general UQ process as an example for users to consider in developing their own process.

1. Define goals and objectives of the UQ study as well as provide detailed specification of the simulation models under UQ study,
2. Identify 'all' relevant sources of uncertainties (in reality, not all sources can be identified and thus some assumptions have to be made),
3. Characterize, quantitatively if possible, the selected sources of uncertainties,
4. Propagate the uncertainties through ensembles of computer simulations,
5. Analyze the impact of the uncertainties,

6. Review and document the findings, and

7. Optionally, reduce the uncertainties as guided by the findings.

In the next few sub-sections, we discuss these steps in more detail and highlight issues about UQ method selections.

5.1 UQ Objectives and Problem Specification

This step entails a full description of the objectives of the UQ study as well as assumptions about the models. In addition, it includes a full specification of all essential components of the computer model under study so that sufficient information are available to enable an objective validation of the UQ results (that is, reproducibility of the results). This is a crucial step in a UQ process, since the conclusions drawn from the study are valid only under the given assumptions. As such, the execution of the study should not move to the next step until this step has been completed to the satisfaction of all project team members.

The first task consists of defining as clearly as possible the objectives of the study. Some of the UQ objectives are:

- to quantify the prediction uncertainties of a model,
- to identify and quantify the contribution from the major sources of uncertainty in a simulation model,
- to validate a simulation model given data from physical experiments given uncertainties in model and data,
- to calibrate model parameters to match data from physical experiments,
- to provide information on designing new physical experiments for reducing prediction uncertainties,

and others.

A second task consists of the full specification of the simulation models as well as model assumptions. The outcome is an initial document giving details of

- the simulation model (source code/input decks and their version numbers, run-time environment),
- a list of all tunable uncertain parameters in the simulation models,
- a list of model outputs of interest,
- all relevant data from physical experiments and their associated uncertainties,
- a comprehensive list of model assumptions (for example, mesh resolution, full/simplified or phenomenological physics models),

- the objectives of the current UQ study, and
- computational resources available for the study.

A practical issue with a lengthy UQ analysis is that continuous simulation code updates may inadvertently alter the outcome of a UQ study. Thus, it is very important that a code version be frozen for a given study.

A complex computer model may contain hundreds of internal parameters that are tunable or uncertain. Due diligence should be exercised in compiling this full list of tunable parameters. A possible next step consists of a careful examination and identification of a representative subset that are deemed important (or sensitive in inducing model output uncertainties) through expert judgment. One reason for needing this expert judgment step is that it may not be possible to characterize the uncertainties of some of the identified parameters. Another reason is that the parameter space may be too large for a comprehensive analysis with limited computing resources and difficult choices have to be made based on expert judgment. Choices made in this first phase of parameter selection need to be documented in detail for later references, as leaving out some of these parameters may alter the UQ results if model simulation outputs are sensitive to these parameters.

Similarly, a comprehensive list of model outputs is to be compiled. Some of the model outputs have physical meanings and are useful for comparison with observational data. Others may be useful as diagnostics to gain confidence in the analysis results. Examples of model outputs are scalar values such as stress or displacement, function data such as a time sequence of pressure or temperature, 2D and/or 3D images, and/or time sequence of images.

Next, efforts should be made to gather experimental data that may be helpful in reducing model output uncertainties. For example, if a certain material fracture model (for example, the Johnson-Cook model is used in the current simulation model and additional focused experimental data are available to help refine the uncertainties of the model parameters, these data should be considered in the current study.

Compiling a full list of model assumptions is critical in defending the UQ results during reviews, as some of these assumptions may not be realistic and they will call the UQ results into questions. Below we give examples of model assumptions.

- A two-dimensional model is used, and we assume three-dimensional effects are negligible.
- A Reynold-averaged Navier Stokes model is used, and it is assumed to provide sufficient accuracy for this study.
- The geometry of some components in the system is simplified for expediency, and we expect it will have little effect on the quality of the solution.
- Some regions of the computational domain assume average particle properties, which should have negligible effect on the accuracy of model predictions.
- Attrition is not modeled explicitly, and we assume attrition is insignificant for this study.

- The mesh size used in the computational domain is assumed to be adequate (based on previous verification results.)

The objectives of the planned UQ study may affect which UQ methods are appropriate. For example, if the objective is to help prioritize research by assessing and comparing the impact of the various uncertain sources on the design, then global sensitivity analysis will be valuable.

Finally, an assessment of the quality of the computer model should be performed. The degree of rigor devoted to this effort should depend on the maturity of the models as well as the judgment of model developers. In any case, the models may need to go through verification and validation (not the main subject of manual) to ensure that numerical errors will not pollute the UQ results and excessive failures (e.g. code crash due to 'time step too small' or 'negative mass detected') will not happen during the course of ensemble simulations.

5.2 Identification of Uncertainty Sources

Suppose we have the following UQ scenario: before building a system, we would like to assess its performance using modeling and simulation. Efforts are expended in developing a mathematical model for the system and implementing the model using suitable numerical algorithms. Furthermore, suppose we are able to perform limited physical experiments to validate our model. Once the modeling effort is completed, we would ask ourselves the question: what confidence do we have that, when we actually build the system, its performance will be consistent with the prediction from the model?

In answering this question, it is important to realize that, in the process of translating from the physical process to a set of mathematical equations and then to the final simulation model, many errors and uncertainties are inadvertently introduced. These errors and uncertainties do negatively affect our confidence in the predictive power of the simulation model. Therefore, quantifying uncertainties of model predictions is a critical task in modeling and simulation. In the following we list some of the possible sources of uncertainty introduced in the modeling and simulation process:

- Uncertainties in the system geometry due to manufacturing variability;
- Uncertainties in the initial and boundary conditions imposed on the simulation models;
- Uncertainties in the surrounding environment (for example, temperature and pressure fluctuations),
- Uncertainties in the physics models (in physics parameters and in model form);
- Uncertainties/errors introduced by missing physics;
- Uncertainties in the experimental data used to calibrate the simulation model;
- Uncertainties introduced by interactions between physics uncertainties and computational errors such as round-off errors and discretization errors;

and others (uncertainties/errors induced by numerical algorithms, for example, convergence tolerance, are not included as they are considered code verification issues). Failure to account for these uncertain sources (and make assumptions about some of them if they are to be ignored in the study) may diminish the usefulness of the UQ results. In summary, the execution of this step should produce a comprehensive list of all relevant uncertain sources (including those that will be ignored in subsequent UQ analysis and justification for the decisions to ignore them).

5.3 Characterization of Sources of Uncertainty

This step corresponds to compiling credible initial ranges and/or probability distributions for the uncertain parameters, and is one of the most important and time-consuming tasks. Extreme caution should be taken in this step because the credibility of the final UQ results depends critically on the choice of ranges and distributions. Therefore, every choice of ranges and distributions should be carefully made and justified.

Before carrying out uncertainty characterization, it may be useful to classify the identified sources of uncertainty. One such classification is based on the nature of uncertainty for which uncertainties are classified as either *aleatory* or *epistemic*. In *A Complete Framework for Verification, Validation, and Uncertainty Quantification in Scientific Computing,* (AIAA 2010-124, 48th AIAA Aerospace Sciences Meeting, 2010), Roy and Oberkampf presented a useful description of these two types. In brief, aleatory uncertainties (also called variability or irreducible uncertainty) are uncertainties due to inherent randomness, the probability distribution of which can be accurately characterized given sufficient measurement data. For example, if a measurement instrument is susceptible to fluctuations in its readings, variations will be observed from one reading to the other. When many measurements are made under the same experimental conditions, a probability distribution can be constructed to characterize this source of uncertainty. This uncertainty can be better characterized with more experiments but it cannot be reduced, unless a more accurate measurement instrument is used. On the other hand, epistemic uncertainties (also called reducible uncertainties) are uncertainties due to limited data and knowledge. An example is a physics constant that should have a fixed value but its exact value is not known by model developers, albeit a range of possible values. This range can be refined with more experiments and thus this uncertainty is reducible. Another example is epistemic uncertainty due to important physics not captured well by the model (missing physics that causes model discrepancies) and this uncertainty can be assessed by comparing model outputs against measurement data.

The distinction between aleatory and epistemic uncertainty is not always clear. For example, an uncertain source may be aleatory but experimental data are insufficient to determine its true probability distribution. In this case, the uncertain source has a mixture of aleatory and epistemic uncertainties and needs to be managed accordingly.

For all practical purposes we confine the definition of aleatory uncertainty to be uncertainty that can be characterized by a known probability distribution; and the definition of epistemic uncertainty to be uncertainty that can be characterized by intervals (or ranges) with unknown probability distributions. Furthermore, epistemic uncertainties include sys-

tematic bias of model output caused by missing or imprecise physics. We prefer this distinction because it clarifies the different approaches to quantifying uncertainties for the two types (to be discussed later).

For uncertain variables (will be used interchangeably with uncertain parameters), another useful classification is: (1) continuous variables; (2) ordinal variables; and (3) categorical variables. Continuous variables can be set to any value within a given interval (or a union of disjoint intervals) and thus they have infinite number of distinct values. Discrete variables, on the other hand, can only take on a limited number of values within their ranges. For example, if the number of particles in a model is an uncertain variable that may be somewhere between 100 and 200, then this is a discrete variable that can only take on a finite number of integer values within the range. Finally, categorical variables do not have meaningful numerical values. They are used for the purpose of distinguishing different options such as different material fracture models that may have little functional relationship with each other.

After the uncertain sources have been classified, their ranges or probability distributions are to be prescribed. Information about 'reasonable' ranges and distributions can be obtained from theory, literature, or local subject matter experts. Complex parameter correlations and/or parameter constraints may arise in many multiphysics applications, and they should be carefully prescribed. Once again, this step has to be performed with great care since choices made in this step may affect the outcome of, for example, sensitivity analysis. In summary, the product of this step is a comprehensive list of relevant uncertain sources appended with their classifications and probability distributions (or intervals).

5.4 Propagation of Uncertainties

Uncertainty propagation concerns studying the effect on model input uncertainties on the model output. Uncertainties may be propagated via intrusive or non-intrusive approaches. Due to its practicality, we only consider non-intrusive approaches.

This step involves generating samples and running a possibly large number of simulations. For complex multi-physics models, however, an additional task may be needed to manage the workflow before running the simulations–the construction of an uncertainty flow diagram. In the following we provide the rationale for this task and give an example to illustrate the process.

After the previous step (characterization), we may find that the initial parameter ranges and probability distributions (called priors) may be too conservative (too wide) to be useful. These distributions can be refined (narrowed) via additional physical experiments on a smaller scale (for example, on a sub-system or on a single-physics module within the full multi-physics model), which we call focused experiments. The refinement process can be achieved via a Bayes-like inversion applied to the sub-system or single-physics module (if the sub-system or module itself has a large number of uncertain parameters, it may require a mini-UQ analyses). For example, if a certain turbulence model is used in the model but some of its parameters are uncertain, then additional focused experiments can be performed to better characterize these parameter uncertainties (intermediate posterior distributions that

will be used as priors for the full system). Since there may be many such focused experiments, their individual contributions will have to be merged in the final stage of uncertainty propagation through the full system model. The construction of an uncertainty flow diagram (UFD) can help facilitate data fusion (which aims at accurate re-analysis, estimation and prediction by fusing observed data from physical experiments into a model) at component, sub-system, and system levels. A simple example is given in the Examples/HierarchicalUQ directory.

To create an UFD, we first need to identify the available sub-system (experimental) data and their corresponding simulation models. Each focused experiment together with its experimental data will be assigned one node in the UFD. Each node thus corresponds to a Bayes-like inversion process (that is, use Bayes-like methods to compute the posterior bounds and distributions given the priors and the likelihood function) for refining the parameter ranges in view of the available data (The reason for using the term "Bayes-like" is that in many cases the posteriors sought after are bounds and not distributions, in which case the 'ABC' or approximate Bayesian computation may be more appropriate.) The input for each node is a subset of uncertain parameters in the full system with their associated prior distributions. The outputs of the node are the posterior distributions of the same input parameters. These posterior distributions may in turn be inputs be another node in the network. Careful construction of the UFD gives an overall picture of uncertainty flow and thus help guide the UQ process.

An Example: Suppose 5 groups of uncertain parameters (e.g. each corresponds to parameters of individual physics in a multiphysics model) have been identified for a simulation model with their corresponding initial (prior) distribution $p(X_1)$ to $p(X_5)$. In addition, data for a few focused experiments (CE) are available to help constrain some of the parameter uncertainties:

- CE1 constrains X_1 using single-physics model and the corresponding experimental data,
- CE2 constrains X_2 using single-physics model and the corresponding experimental data,
- CE3a constrains X_1, X_2 and X_3 together using the subsystem model and the corresponding experimental data,
- CE3b is analogous to CE3a but it corresponds to a different focused experiment to constrain X_1, X_2 and X_3,
- CE4 constrains X_4 using single-physics model and the corresponding experimental data,
- CE5 constrains X_5 using single-physics model and the corresponding experimental data, and
- CE6 constrains X_4 and X_5 together.

These initial distributions and focused experiments can be integrated into a uncertainty flow diagram as depicted in Figure 1.

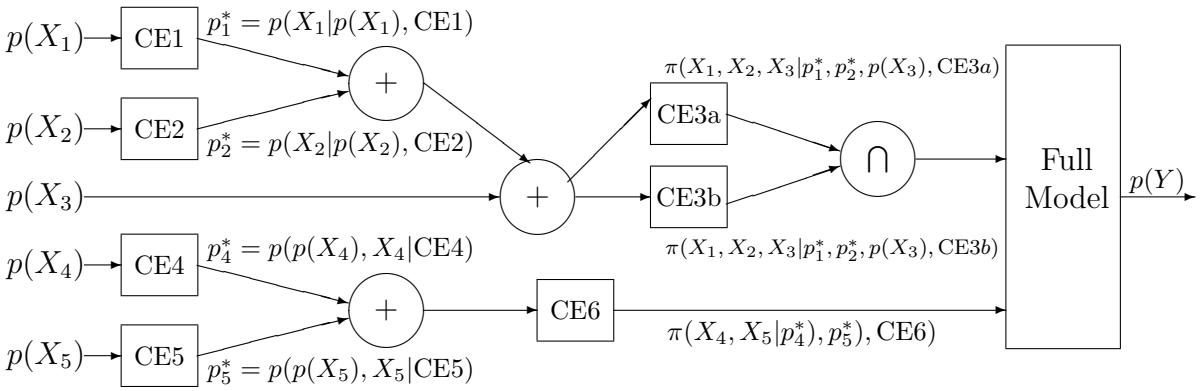

Figure 1: Uncertainty Flow Diagram for this example

Here + denotes concatenation of parameter distributions, and $\bigcap$ denotes 'intersection' of the two distributions (project team members need to agree on whether it should be $\bigcap$ or $\bigcup$ based on sound reasoning). The posterior distributions are denoted by, for example, $\pi(X_4, X_5|p_4^*, p_5^*, CE6)$ describing the posterior distributions of X_4 and X_5 conditioned on their intermediate priors p_4^* and p_5^*), as well as the focused experimental data in CE6. Finally, all the 'intermediate' posteriors will be used as priors on the full model to quantify uncertainties and sensitivities.

A key element to propagating uncertainty is the sampling strategy (also called design of computer experiments). The proper selection of sampling strategies depends on the UQ objectives, uncertain parameter distributions, as well as certain properties of the simulation models. In summary, the execution of this step produces sampling designs at various stages of the uncertainty workflow and runs ensemble simulations to obtain sample outputs.

5.5 Analysis of Uncertainties

UQ objectives guide the selection of relevant sampling strategies and analysis methods. In the following we list several analyses commonly performed during a UQ study:

- Quantify the statistical moments or some other statistical metrics of selected model outputs,
- Select a subset of a large number of uncertain parameters for more detailed study based on an objective analysis (via screening or down-select),

- Construct response surfaces relating model input parameters to the model outputs to be used as computationally inexpensive surrogates in place of the actual simulators for faster subsequent analysis,
- Quantify the global sensitivities of uncertain parameters,
- Use observational data for model parameter estimation/calibration,
- Optimize system performance in the presence of uncertainties,
- Analyze system reliability in the presence of uncertainties,

and others. In addition to quantitative analysis, qualitative analysis methods such as visual inspection of trends and variations due to uncertainty are other viable ways of assessing model uncertainties.

5.6 Documentation/Reporting

The first three steps of a UQ study (definition, identification, characterization) normally consumes more than half of the overall project effort. Depending on the availability of computing resources and the simulation cost, the propagation and analysis steps may take up most of the remaining effort. Other important tasks in a UQ study are the rigorous internal and external reviews as well as detailed documentation of the study. Internal reviews (among project team members) should be conducted frequently, and especially after each step has been completed. Insufficient attention given to rigorous reviews may result in wasted efforts and erroneous conclusions. Detailed documentation of the entire process will increase transparency and also enable the replication of the UQ study and confirmation of UQ results by a third party.

5.7 Reduction of Uncertainties

Quantifying the uncertainty of a model output is often not the final objective. Specifically, when the model output uncertainty is unacceptably large, UQ results may help guide new efforts to reduce or manage uncertainties. For example, sensitivity analysis results may be useful in pinpointing the major uncertain sources (some physics modules or observation data sets) where improvements are needed. Tracing uncertainty flow through the UFD may also suggest where additional experimental data should be collected for guiding uncertainty reduction. Optimal experimental design, which has been an active area of research and development in the clinical community, can be a valuable tool for driving experimental campaigns.

Uncertainty reduction can be tackled from two different perspectives: uncertainty reduction in model uncertain parameters and uncertainty reduction in model predictions. For each perspective a relevant metric (or utility function) should be prescribed to measure the effectiveness of new experiments. For model parameters, volume of the hyperellipsoid or

the length of the major axis of the hyperellipsoid enclosing the probable parameter values are popular metrics. For model prediction, prediction variance is a natural choice. Some of these uncertainty reduction methods are captured in PSUADE's optimal experimental design module and examples can be found in the Examples/ODoE directory.

6 Summary

PSUADE has been designed to be a general-purpose toolkit for uncertainty quantification. Many enhanced features have been incorporated based on our experiences gained from practical application to complex multi-physics models; and not all of these features have been comprehensively described in this document. Users are encouraged to go through all examples included in the software releases and give us feedback and suggestions on improving the manuals and also the software itself.