PSUADE Short Manual
(Version 1.7)

Charles Tong

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory

Livermore, CA 94551-0808

September, 2014

1   Introduction

PSUADE (Problem Solving environment for Uncertainty Analysis and Design Exploration) is a software package for various uncertainty quantification (UQ) activities such as uncertainty assessment (UA), sensitivity analysis (SA), parameter study, numerical optimization, etc. It consists of three major components: a suite of sampling methods, a job execution environment, and a collection of analysis tools. This document describes how to set up and use these UQ tools. The mathematics of the sampling and analysis methods can be found in the theory manual.

1.1   A Quick Start

Follow the instruction in this section and you should be able to build and run PSUADE (on a simple example) in less than 5 minutes (depending on the speed of your hardware) on a Linux-based system (Please consult PSUADE developers for a Windows version of the PSUADE executable and the associated installation guide). For building PSUADE executables on other platforms (MAC, Windows), refer to detailed instruction in a later section.

1. gzip -d PSUADE_v1.7.x.tar.gz

2. tar xvf PSUADE_v1.7.x.tar

3. cd PSUADE_v1.7.x

4. For adding the external Gaussian process capability, consult the PSUADE developers on how to install it.

5. Set the 'FC' environment variable to your preferred Fortan compiler, if you have one (cmake will select one automatically if not set).

6. mkdir build

7. cd build

8. ccmake .. (Select packages by typing 'c' and then using the arrow keys to move up and down the list and type 'enter' to select. When you are finished with package selection, type 'c' (may be twice) and then 'g' to save and exit. If you would like to install the executable somewhere else, set the install directory)

9. make (to create the 'psuade' executable and libraries in the buiid/bin directory), or

10. make install (to install 'psuade' in your install directory if you would like to make it available to your project team).

11. To verify correct installation, do: cd Examples/Bungee

12. cc -o simulator simulator.c -lm

13. **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 the summary statistics; and all sample input and output data will have been stored in the **psuadeData** file. 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). Some of the available sampling methods are:

- Monte Carlo and quasi-Monte Carlo

- Latin hypercube and orthogonal arrays

- Morris one-at-a-time, its variants, and other screening designs

- Central composite, factorial and fractional factorial

- Fourier Amplitude Sampling Test (FAST)

- Other space-filling designs

- Support several popular input probability distribution functions

These sample points are then evaluated by running the user simulation codes. PSUADE provides a mechanism to accomplish this via a runtime environment which performs the following tasks when invoked:

- Write the values of a sample point to a parameter file.

- Call the user code (provided by users in the PSUADE input file - 'driver') with the parameter file as its first argument.

- The user code is expected to read in the parameter values from the parameter file, run the application, produce some output quantities and write them to an output file which is specified as the second argument when calling the user code. Thus, the user code can be a simple program such as **simulator.c** in some of our examples, or a complex super-script performing preprocessing, actual model evaluation and postprocessing.

- PSUADE detects the presence of the output file and reads in the outputs.

- PSUADE moves on to the next sample point and continues until all sample points have been processed (PSUADE can process multiple sample points at the same time using asynchronous mode).

- Finally, PSUADE reads in all sample data and analyzes them based on user requests given in the PSUADE input file (the 'ANALYSYS' section).

PSUADE supports many types of analysis such as

- Parameter screening (several such methods)

- Response surface construction and validation (including adaptive methods)

- Basic uncertainty and correlation analysis (for raw sample or response surfaces)

- Main effect (first order sensitivity) analysis

- Two-way interaction (second order sensitivity) analysis

- Group and total sensitivity analysis

- Bayesian calibration

- Hypothesis testing

- Deterministric numerical optimization

- Mixed aleatory-epistemic uncertainty analysis

- Graphical analysis (e.g. scatter plots via Matlab/Scilab)

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'
    Select BUILD_SHARED, MARS, BOBYQA, and METIS (Note: Consult PSUADE
        developers for instructions on how to install other packages).
```

```
     If you need to change the compiler, hit 't' and find the
        CMAKE_C_COMPILER and CXX fields and fix 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 ..
     * and then open the CMakeCache.txt file and turn on the packages
     *    MARS, BOBYQA, and METIS.
  [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' has been selected, 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. If it is desirable to have the executable and libraries accessible to multiple users, you can 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 requires 'cmake', 'macports', 'gcc', 'g++', and 'gfortran' with version 4.4 or higher. A session to check the compiler versions is given below:

2.2.1   Step 1: Check Compilers

This step consists of checking to make sure you have 'gcc' installed from 'macports'. A session to check the compiler versions is given below (which shows that all except the 'cc' compiler are from macports):

```
[macos] cc --version
Apple LLVM version 4.2 (clang-425.0.27) (based on LLVM 3.2svn)
Target: x86_64-apple-darwin12.4.0
Thread model: posix

[macos] gcc --version
gcc (MacPorts gcc45 4.5.4_6) 4.5.4
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[macos] c++ --version
c++ (MacPorts gcc45 4.5.4_6) 4.5.4
```

```
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


[macos] gfortran --version
GNU Fortran (MacPorts gcc45 4.5.4_6) 4.5.4
Copyright (C) 2010 Free Software Foundation, Inc.
```

2.2.2   Step 2: Run Cmake

```
[Linux] mkdir build
[Linux] cd build
[Linux] ccmake ..
    hit 'c'
    Select BUILD_SHARED, BOBYQA, and METIS (Note: Consult PSUADE
        developers for instructions on how to install other packages).
    hit 'c'
    hit 't' to go to advanced options.
    I can see that in my case cmake has picked up the wrong cc:
    CMAKE_C_COMPILER                    /usr/bin/cc
    So I need to change it to:
    CMAKE_C_COMPILER                    /opt/local/bin/gcc

    Aside from this it seems OK, so I hit 'c' until I can hit 'g', then
    hit 'g' to generate and exit.
```

2.2.3   Step 3: Build Executable

To build the PSUADE executable, do the following:

```
[Linux] make
```

At the end of this installation, the PSUADE executable will have been created in the build/bin directory.

2.3   Windows

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.

2.3.1   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.
Select BUILD_SHARED, BOBYQA, and METIS.
Click 'Generate'.
```

2.3.2   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)
```

2.3.3   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 in one of the two modes: batch and command line modes.

3.1   Batch Mode

In batch mode, PSUADE interacts with users via a few files. At the first level, an PSUADE input file (called psuade.in here) has to be created and run via

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

This psuade.in file should begin with the keyword PSUADE as the first line and should have 5 subsections followed by the last line having the keyword END. The formats of the subsections are described next (for an example, read the psuade.in file in the Examples/Bungee directory).

3.1.1 The Input Section

The INPUT section allows the users to specify the number of inputs, their names, their range, and their 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
   PDF 4 N  0.0    1.0
   COR 2 4  0.5
END
```

In this example the number of inputs is 4, their names are X1, X2, X3, and X4 (notice that the variable indices are 1-based), and their lower and upper bounds are all 0 and 1, respectively. The probability density distributions (PDF) for the inputs are optional (the default is uniform U). If the PDF is either normal (N) or lognormal (L), the mean and standard deviation also have to be provided. If the PDF is triangular (T), the mean and width are to be provided. Other available distributions are beta (B), exponential (E), gamma (G), Weibull (W), and user-provided samples (S). The last option (S) allows probabilities to be represented by a pre-generated sample file ('sample3' above) such as the posterior samples generated from Bayesian inference. An example of the 'S' type sample file can be found in the Examples/PDFTest/sample file. The 'S' option also facilitates the use of discrete variables even though PSUADE assumes all variables to be continuous. 'COR' specifies the correlation between two inputs and it is currently available only for the normal distribution.

3.1.2 The Output Section

The OUTPUT section is similar to but simpler than the INPUT section. Here only the output dimension and the names of the output variables are needed. For example, given as follows:

```
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. An example is given below.

```
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 in the theory manual). 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 $600/60 = 10$. 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 repeatable (having the same random seed every time the experiment is repeated.)

Some of the other sampling methods available 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 parameters. An example is given below.

```
APPLICATION
   driver = ./testmain
   opt_driver = NONE
   max_parallel_jobs = 1
   max_job_wait_time = 1000000
END
```

Here driver points to the executable to be used for function evaluations, and opt_driver points to the executable for numerical optimization, if it is used. Again, the user code can just be a simple program or a complex super-script performing preprocessing, actual model evaluation and postprocessing. The user code can also be a PSUADE data file itself, as will be shown later.

After the creation of a sample based on information from the INPUT, OUTPUT and METHOD sections, PSUADE proceeds with launching the jobs. If the max_parallel_jobs is set to 1, the sequential mode is turned on. In this mode, PSUADE schedules the evaluation of the user-provided function by sequencing from sample point 1 onward. 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 by the values of the input parameters for the i-th sample point. PSUADE then calls driver with two parameters (for example, for the sample point 9)

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

The driver program is expected to take the input parameters from the psuadeApps.in.9 file, do whatever is needed, and write the outputs to the psuadeApps.out.9 file. An example of the content of an input file (2 variables) created by PSUADE is:

```
2
0.345
1.429
```

An example of the content of an output file (3 variables) to be created by user programs is:

```
3.12
15.9
100.4
```

If max_parallel_jobs is set to a number larger than 1, then the asynchronous job scheduling mode is turned on. In this mode, multiple psuadeApps.in.i files are created simultaneously, and driver is called max_parallel_jobs times simultaneously. max_job_wait_time is used for fault detection and recovery.

Some of the other options in this section are:

```
launch_only         - launch all jobs without waiting for results
gen_inputfile_only  - generate all sample files only (no code runs)
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. In this case PSUADE is terminated after all jobs have been launched, and when all the jobs are completed, PSUADE should be launched again to harvest the results.

gen_inputfile_only is useful when job launching management is handled outside PSUADE. In this case PSUADE is launched initially to create the input files for all sample points (all psuadeApps.in.i). At this point your preferred job scheduler can take over to run all sample points. After the runs have been completed and all psuadeApps.out.i have been created, PSUADE should be called again to harvest the results.

Finally, limited_launch_only is useful in a semi-automated job launching environment (e.g. on the LC machines at LLNL). In this case PSUADE is called initially to launch a small number of jobs (e.g. 10 out of 100 in total). The job execution scripts for these 10 jobs are equipped with capabilities to launch more jobs when it is completed. This 'domino effect' job launching (also called 'chain mode') will continue until all 100 jobs have been completed. After that, PSUADE should be called again to harvest the results.

3.1.5   The Analysis Section

The ANALYSIS section specifies what type of analysis is desired and the parameters used in the analysis. An example is given below (which computes the different statistical moments on output number 1 based on the sample selected above). Comment lines begin with '#' (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:

```
MainEffect      - sensitivity indices
TwoParamEffect - second order sensitivity indices
RSFA            - response surface 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       - response surface-based first order sensitivity analysis
RSMSobol2       - response surface-based first second sensitivity analysis
RSMSobolG       - response surface-based group main effect analysis
RSMSobolTSI     - response surface-based total sensitivity analysis
```

When response surfaces are used together with the selected analysis method, the response surface type (rstype above) has to be specified. The available response surface types are (some of these are not included in the release):

```
MARS              - multi-variate adaptive regression splines (by Friedman)
MARSBag           - MARS with bootstrapped aggregation
linear            - linear regression
quadratic         - second order polynomial
cubic             - third order polynomial
quartic           - fourth order polynomial
user_regression   - user-specified polynomial
GP1               - Gaussian process (Tpros)
SVM               - support vector machine
Kriging           - an universal Kriging method
SOT               - a sum-of-trees method
KNN               - K nearest-neighbors
RBF               - Radial Basis functions
sparse_grid_regression
Splines
```

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 crude (a simple examination of all sample outputs and select the minimum one), minpack (an external optimization package), cobyla (an external optimization package), and sm/mm (space-mapping and manifold-mapping methods by David E. Ciaurri). num_local_minima tells PSUADE how many minima to identify (from the initial sample) 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 the minima. num_fmin tells PSUADE the number of optimal points to be expected so that PSUADE can decide to stop when this has been achieved (num_fmin needs to be larger or equal to num_local_minima. Also, users can also tell PSUADE what the optimal value to look for via fmin.

3.2 Command Line Mode

PSUADE allows users to interactively perform some of the analyses. The idea is to run all the simulations with the batch mode (and delete or comment out all analysis methods in the input file). Once all simulations have been completed, the psuadeData file will contains all sample inputs and outputs enclosed in the PSUADE_IO section. This file (which needs to be renamed) is to be loaded in the command line mode for analysis. Command line mode is activated by calling

```
[Linux] psuade
```

without any argument. Some of the available commands in the command line mode are (most of the commands can be found by the help command and also in the reference manual):

```
load <filename>   (load a data file, e.g. psuadeData)
splot             (generate scatter plot in matlab)
me                (main effect study + matlab plot)
rs2               (2-input response surface in Matlab)
rs3               (3-input response surface in Matlab)
rs3m              (3-input response surface in Matlab (movie)
rsi2              (2-input response surface intersections in Matlab)
rsi3              (3-input response surface intersections in Matlab)
rscheck           (Check quality of response surface with MARS)
quit
help
```

For example, after you have completed a set of runs, a PSUADE data file will be created (say, the renamed file is psData). To create scatter plots for the data in the command line mode, do:

```
[Linux] psuade
*** ********************************************************** ***
*** Welcome to PSUADE (version 1.7.2)                         ***
*** ********************************************************** ***
PSUADE - A Problem Solving environment for
Uncertainty Analysis and Design Exploration
psuade> load psData
psuade> splot
matlabsp.m is now available for scatter plots.
psuade> quit
[Linux]
```

You can now use Matlab to display the scatter plot. You can also generate Scilab files by toggle the 'scilab' command (scilab capability is less well-developed as matlab capability inside PSUADE).

4   Examples

PSUADE provides many tools for answering many questions with uncertainty quantification. For example, given a computational model simulating some physical processes,

1. How to assess the impact of parameter uncertainties on the variabilities of some output of interest? (uncertainty analysis)

2. How do available experimental data alter the outcome of an uncertainty analysis? (conditional or Bayesian analysis)

3. How to identify a small subset of parameters accounting for most of the output variabilities ? (parameter screening)

4. How to quantify the impact of a particular subset of parameters on the output uncertainties ? (global sensitivity analysis)

5. How to construct a relationship between some input parameters to the model and the output of interest? (response surface modeling)

6. How to find the parameter values that best fit the available experimental data ? (calibration, parameter estimation)

7. How to search for the parameter values that give the best model performance? (optimization)

8. How to process, manipulate and visualize the uncertainty data?

9. How to formulate and perform hypothesis testing?

In the following we provide a few examples to show in more details 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 Rosenbrock function: let the function be given by

$$Y = \sum_{i=1}^{m-1} (1 - X_i)^2 + 100(X_{i+1} - X_i^2)^2 \quad X_i \in [0, 2]$$

where m is the number of input parameters and can be any integer larger than 1 (we use $m = 6$ in this example). To compute the basic statistical moments of this function assuming all inputs are uniformly distributed in $[0, 2]$, we select the Latin hypercube design with a sample size of 100 (an arbitrary pick). The simulations are to be run in sequential mode. The corresponding PSUADE input file psuadeRawUA.in is (this example is in the Examples/UserExample directory):

```
PSUADE
INPUT
   dimension = 6
   variable 1 X1  =   0.0   2.0
   variable 2 X2  =   0.0   2.0
   variable 3 X3  =   0.0   2.0
   variable 4 X4  =   0.0   2.0
   variable 5 X5  =   0.0   2.0
   variable 6 X6  =   0.0   2.0
END
OUTPUT
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = LH
   num_samples = 100
   randomize
END
APPLICATION
   driver = ./testmain.py
END
ANALYSIS
   analyzer method = Moment
   diagnostics 3
END
END
```

Here 'randomize' in the 'METHOD' section specifies that random perturbations are to be added to the sample. The '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). The driver program can be in any language provided that it can be executed (by the Linux 'system' command). Our example uses Python to represent the above function (Examples/UserExample/testmain.py:

```
#!/usr/local/bin/python
import string
import sys
infile  = open(sys.argv[1], "r")
lineIn  = infile.readline()
ncols   = string.split(lineIn)
n       = eval(ncols[0])
X       = range(n)
```

```
for ii in range(n):
   lineIn  = infile.readline()
   ncols   = string.split(lineIn)
   X[ii]   = eval(ncols[0])
infile.close()
Y = 0
for ii in range(n-1):
   Y = Y + pow(1 - X[ii], 2) + 100 * pow(X[ii+1] - X[ii] * X[ii], 2)
outfile = open(sys.argv[2], "w")
outfile.write("%e \n" % Y)
outfile.close()
```

After these files have been prepared (make sure the Python link in testmain.py is correct, and that testmain.py has execute permission turned on), run PSUADE with:

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

and, at the completion of the runs, the moment information will be displayed and the psuadeData file (which contains the 100 sets of sample inputs/outputs) will also be available for use in further analysis.

The above shows how to perform uncertainty analysis in batch mode. This analysis can also be conducted in two steps: (1) run the simulation and ignore the analysis (by commenting out the 'analyzer method' line in psuadeRawUA.in); (2) renaming psuadeData to, for example, psData); and (3) load psData in command line mode and issuing the ua command. This two-step approach is recommended because the command line interpreter provides many other functions to analyze the same data set (e.g. ca for correlation analysis).

In this example, we observe that even for computing simple statistical moments, a sample size of 100 may be too small. You may increase the sample size and re-run to see if the result changes much. You can also turn on the refinement mode (by uncommenting the line 'num_refinements = 5' and re-run), which will iteratively increase the sample size and re-anaylze.

4.2   Screening for Important Inputs

A useful design and analysis tool in PSUADE is the suite of parameter screening methods. There are several parameter screening methods that may be useful under different scenarios. In this section we provide a survey of these methods and their applicability; and then we will show how to use one of them.

If you already have a sample available for analysis (that is, simulations are complete), the following screening methods may be useful:

1. Correlation analysis (ca). This method, however, assumes that the model input-output relationship is more or less linear. If the model input-input relationship is not linear but exhibits monotonic behavior (non-decreasing or non-increasing), the Spearman

coefficients are more informative. For general nonlinear relationships, this analysis may give erroneous results.

2. Delta test (delta_test). This method works only with relatively large (hundreds to a thousand) random or quasi-random samples. Since this method involves minimization of certain noise function, it may be computationally intensive. Screening results may be confirmed by using different sample sizes.

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

4. Approximate response surface methods. If your already-run sample is small, you can perform a response surface analysis on your sample and use the response surface to rank parameters. 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 right trend (error distribution centers around zero).

If you do not have a sample ready for analysis, and the simulations are computationally expensive, a more careful sample design (called design of experiment techniques) is needed. Again, design selection depends on your knowledge about the simulation model and how much computational resources are available. PSUADE currently provides the following screening methods:

1. If the model input-output is known to have a near-linear relationship, then the Plackett-Burman or local sensitivity analysis design (and the corresponding lsa analysis command) may suffice. These method requires only $m + 1$ simulations (m is the number of uncertain parameters).

2. If, in addition to linear relationship between each input and the model output, there are also interaction terms (e.g. the model equation consists also of terms involving two or more inputs, then the fractional factorial designs and the corresponding analysis (ff) may be useful.

3. 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).

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/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
      diagnostics 3
   END
   END
```

Here the sample size should be a multiple (usually 10) 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 analysis can be performed interactively by (again psuadeData has been created and has been renamed to psData):

```
[Linux] psuade
*** ********************************************************* ***
*** Welcome to PSUADE (version 1.7.2)                        ***
*** ********************************************************* ***
PSUADE - A Problem Solving environment for
Uncertainty Analysis and Design Exploration
psuade> load psData
load complete : nSamples = 210
nInputs  = 20
nOutputs = 1
psuade> moat
... (MOAT results) ...
```

```
...
Create screening diagram ? (y or n) y
matlab/scilab screening diagram file name (no extension): screen
MOAT screening diagram matlab file = screen.m
psuade> quit
[Linux]
```

Thereafter, you can launch Matlab and run **screen** to view the Morris screening diagram (scatter and bootstrap plots can also be generated).

4.3 Response Surface Analysis

As we can see from the previous exercise, even a relatively simple test problem may require large samples to compute the statistics accurately. 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 by a simple curve fitting method. In this example we show how to use the curve fitting tools in PSUADE to find the best response surface model. To do this, we run PSUADE on Examples/UserExample/psuadeRS.in to evaluate a sample of size 100. We then rename the sample file **psuadeData** to **psData** so that it will not be overwritten by PSUADE (since 'psuadeData' is the PSUADE default output file). The following gives a snapshot of how to perform response surface analysis in the PSUADE command line mode:

```
[Linux] psuade
**********************************************************************
*       Welcome to PSUADE (version 1.7.2)
**********************************************************************
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration
(for help, enter <help>)
======================================================================
psuade> load psData
load complete : nSamples = 100
nInputs  = 6
nOutputs = 1
psuade> rscheck (NOTE: this command performs response surface analysis)
... list response surface types ....
Enter you choice? 2 ('2' is for quadratic regression)
... (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)
```

19

```
...
RSA: final CV error = -3.694e+02 (avg unscaled)
RSA: final CV error = -5.043e-01 (avg   scaled)
RSA: final CV error =  4.157e+02 (rms unscaled)
RSA: final CV error =  6.512e-01 (rms   scaled)
RSA: final CV error =  1.061e+03 (max unscaled, BASE=2.432e+03)
RSF: final CV error =  2.879e+00 (max   scaled, BASE=2.479e+02)
...
CV error file is RSFA_CV_err.m
***********************************************************************
psuade> quit
[Linux]
```

Following the procedure and upon exit from PSUADE, there should be a file called 'RSFA_CV_err.m' in your working directory. This file can be run in Matlab giving two plots: the left showing the distribution of the cross validation errors and the right showing qualitatively how good the selected curve fitting method is. In addition, when 'printlevel 4' has been issued before running 'rscheck', another file called 'psuade_rsfa_error.m' will be created. To assess, the 'goodness' of the response surface, we recommend the following:

1. The 'final CV errors' similar to those shown above give important information. For example, if there is a large departure of the scaled average error from zero, a significant systematic bias may be present. If the scaled max error is large (> 1) with non-negligible base (e.g. if BASE=0.53 and max scale 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. Launching Matlab and running 'psuade_rsfa_error.m' will display a few plots. The Matlab Figure 2 will show the distribution of errors 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' and no further analysis is needed. The left plot in Matlab Figure 1 shows the 'training errors' for individual sample points and the right plot compares the actual sample data against the predicted values from the response surface (perfect predictions put every point on the diagonal line). The Matlab Figure 3 displays 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 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.

20

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

After some preliminary analysis, it should be clear that fitting with quadratic regression does not lead to a satisfactory response surface. Since this is a fourth order polynomial, analyzing the data set with cubic regression will not improve the quality of the response surface either. However, the data set of size 100 is not large enough to analyze with quartic regression (needs a sample of 210). To see that quartic regression is the ideal candidate, you can change the 'num_samples' field in psuadeRS.in to 220 (210 plus a few more to allow room for cross validation) and perform the simulation runs. Now run 'rscheck' with quartic regression and cross validation with 22 groups and observe an almost perfect fit.

You may experiment with other response surfaces and compare their error properties. 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 until the response surface can be 'validated' (using the refine or a_refine functions).

Once you have identified a suitable response surface, it will be ready for subsequent analysis. Note that there are two types of response surfaces provided by PSUADE - the ones that predict the output given the input values (splines, MARS, sparse grid, SVM), and the ones that also provide errors associated with the predictions (polynomial and user-defined regressions, MARS with bootstrapping, Kriging, nearest-neighbors, sum-of-trees), so if you desire to include response surface uncertainties into subsequent analysis, select the ones which provide errors.

Once a suitable response surface has been constructed, many users express interest in getting the actual stand-alone code for future interpolation using this response surface. Users can access this option in the new PSUADE version (1.7.2 and after) via turning on the

'rs_codegen' mode 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. In addition, a file called 'psuade_rs.py' containing the Python interpolation code will also have been created.

4.4   Response Surface-Based Uncertainty Analysis

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

1. Generate a sample to run through the validated response surface.

   (a) Replicate the sample file (psData from previous section), e.g. to ps.in.

   (b) Take out the 'PSUADE_IO', 'APPLICATION', and 'ANALYSIS' sections in ps.in.

   (c) Modify the 'INPUT' section as needed (e.g. change ranges or add distributions to reflect desired input distributions).

   (d) Modify the 'METHOD' section to change the sample size (to say, 50000) large enough to perform reasonable uncertainty analysis.

2. Run the large sample generated in the last step by calling psuade ps.in, which gives psuadeData at completion.

3. Rename psuadeData, e.g., to ua_sample.

4. Launch PSUADE in command line mode to run uncertainty analysis:

   (a) Load psData,

   (b) Run the rsuab command,

   (c) Enter ua_sample when prompted for a sample file, and

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

5. Launch Matlab and run 'matlabrsuab'.

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

   (b) The bottom left plot gives the model output probability distributions for each bootstrap,

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

   (d) If the response surface uncertainty is significant, the bootstrapped probability distribution curves will be easily distinguished.

Another command for response surface-based uncertainty analysis is the 'rsua' command, which performs an average case or worst case analysis. Use 'rsua -help' to see what these functions do.

4.5 Quantitative Sensitivity Analysis

Quantitative sensitivity analysis include main effect, two-way interaction effect, group main effect, and total sensitivity analysis. Main effect analysis studies the first order sensitivities of individual input parameter based on variance decomposition. Sensitivity indices can be computed using one of the following three methods:

1. Use replicated LH directly on the simulator or its response surface;

2. Use FAST sampling directly on the simulator or its response surface; or

3. Use direct numerical integration on the response surface.

In the following example, we describe the use of the replicated Latin hypercube approach directly on the simulator. The input file is given as follow (this can be found in the Examples/UserExample/psuadeME.in file):

```
PSUADE
INPUT
   dimension = 6
   variable  1 X1  =   0.0   2.0
   variable  2 X2  =   0.0   2.0
   variable  3 X3  =   0.0   2.0
   variable  4 X4  =   0.0   2.0
   variable  5 X5  =   0.0   2.0
   variable  6 X6  =   0.0   2.0
END
OUTPUT
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = LH
   num_samples = 1000
   num_replications = 50
   randomize
END
APPLICATION
   driver = ./testmain.py
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 Examples/UserExample and run the following (first make sure to change the file permission to allow Python script to be execute-ready):

```
Linux] psuade psuadeME.in
```

At the conclusion of the analysis, main effect statistics will be displayed. More information will be displayed if the **diagnostics** level is increased. In addition, a Matlab plot of the main effects (**matlabme.m**) will have been created.

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 Examples/UserExample/psuadeRS.in to create a sample of size 100 for use in constructing a response surface. Again, rename the file **psuadeData** to **psData** and then run PSUADE again with Examples/UserExample/psuadeRSME.in, which is given below (It is important that you set the response surface type in **psData** before you launch this run).

```
PSUADE
INPUT
   dimension = 6
   variable 1 X1  =   0.0   2.0
   variable 2 X2  =   0.0   2.0
   variable 3 X3  =   0.0   2.0
   variable 4 X4  =   0.0   2.0
   variable 5 X5  =   0.0   2.0
   variable 6 X6  =   0.0   2.0
END
OUTPUT
   dimension = 1
   variable 1 Y1
END
METHOD
   sampling = LH
   num_samples = 50000
   num_replications = 50
   randomize
END
APPLICATION
   driver = psData
END
ANALYSIS
   analyzer method = MainEffect
END
END
```

Again, simple run the following command

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

and main effect results will be displayed, along with a few Matlab plot files (if selected). 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.

Again, one can use command line interpreter to perform the main effect analysis (by running the above script without setting the analysis method, loading the result file, and use the me command). Note that me will work for any sampling design (not just replicated LH) although the result will be less informative.

PSUADE also provides the functionality to perform 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 will 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, you can run PSUADE with Examples/UserExample/psuadeRSIE.in. After completing this run, you can also load the sample data and have additional analysis.

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, run PSUADE on Examples/UserExample/psuadeRSIE.in to obtain the psuadeData file and rename it. Launch the PSUADE command line mode, load the data file, and run 'tsi'.

The final example in this section shows how to compute sensitivity information in an alternative method - using direct numerical integration. Once the response surface has been validated and deemed satisfactory, it psData can be loaded into PSUADE's command line interpreter (make sure you indicate which response surface to use inside psData before loading the sample file):

```
[Linux] psuade
*** ********************************************************** ***
*** Welcome to PSUADE (version 1.7.2)                         ***
*** ********************************************************** ***
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration
psuade> load psData
load complete : nSamples = 100
               nInputs  = 6
               nOutputs = 1
psuade> rssobol1b
...
Choose which output
Choose how many bootstrap samples to use
```

```
...
...
rssobol1 Statistics (based on 100 replications):
Input    1: mean =   1.1243143e+00, std =   0.0123132e+00
Input    2: mean =   2.5523545e+00, std =   0.1232000e+00
Matlab plot for first order sensitivities is in matlabrssobol1b.m.
psuade> quit
[Linux]
```

At the conclusion of the session, the main effects together with their standard deviations will be displayed. In addition, a Matlab file is also available for visualizing the main effects. This analysis is different from 'me' on response surfaces in that it provides error bars to each sensitivity index based on bootstrapping. You can mimic this function with 'me' by generating multiple bootstraps yourself, running each with 'me', and computing the means and standard deviations of the sensitivity indices. Similarly, PSUADE provides 'rssobol2b' and 'rssoboltsib' to compute second and total order sensitivities with error bars from bootstrapping.

For response surface-based sensitivity analysis, you can experiment with different input distributions by modifying the input section of your psuadeRSME.in (or psuadeRSIE.in, etc.) file or, if you use 'rssobol1' (or similar commands), modifying your psData file to be loaded in command line mode. In the former case, if you use any non-uniform distributions, you need to change the sampling method to MC.

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 correlations (input constraints) into main effect analysis:

1. If you are using replicated Latin hypercube, say, in psuadeRSME.in, the steps are:

   (a) Generate a sample (for example, run PSUADE o psuadeRSME.in with analysis turned off;

   (b) Apply input constraints to filter out infeasible sample points in the sample file from the last step;

   (c) launch PSUADE and run 'me'.

2. Alternatively, you can also use numerical integration ('rssobol1b'). In this case, you will need to turn on the 'rs_constraint' line in your sample data file (psData). 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 $0 < X_2 + X_3 < 1$, then 'constrSample' should contain a sample for the function $Y = X_2 + X_3$; 'indexFile' should contain 2 and 3; 'Lbound'$= 0$; and 'Ubound'$= 1$.

4.6   Mixed Aleatory-Epistemic Uncertainty Analysis

When some of the inputs are epistemic parameters, it is not sufficient just to display the output probability distribution. Rather it should be an ensemble of probability distributions with each corresponding to the distributions due to variations in the aleatory parameters at some fixed epistemic parameter values drawn from their respective ranges. PSUADE provides the functionality to perform this analysis.

Since this analysis is computationally intensive, it is currently implemented on response surfaces only. So the first step in this analysis is to create a response surface (say you have done it and the sample for constructing the response surface is in psData). Next, you need to prescribe the probability distributions for the aleatoric parameters by modifying the 'INPUT' section in psData (you can keep the epistemic variables with uniform distributions). Then, launch PSUADE in command line mode, load psData, and run 'ae_ua'. You will be asked to select which input parameters should be considered as epistemic. At the completion of this analysis, PSUADE will output a 'matlabaeua.m' file for viewing 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 an uncertainty about the mean; then this analysis 'so_ua' draws samples from this second level parameter uncertainties and generates distributions for each. At completion, this command will create a 'matlabsoua' file for visualizing the uncertainties.

4.7   Numerical Optimization

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 = 2
   variable  1 X1  =   -2.0   2.0
   variable  2 X2  =   -2.0   2.0
END
OUTPUT
```

```
        dimension = 1
        variable 1 Y1
    END
    METHOD
        sampling = FACT
        num_samples = 9
    END
    APPLICATION
        driver = ./simulator
        opt_driver = ./simulator
    END
    ANALYSIS
        optimization method = bobyqa
        optimization num_local_minima = 3
        optimization max_feval = 10000
        optimization tolerance = 1.0e-4
        optimization print_level = 2
    END
END
```

This analysis first creates a 3×3 factorial sample. The 9 sample points are evaluated and the 3 (since **num_local_minima = 3** points with the lowest output values are selected as the starting points for a multi-start optimization. The maximum number of function evaluation is set to be 10000 and the termination tolerance is set to be $1e-4$. The **driver** points to an executable called **simulator**. Again, a PSUADE data file such as **psData** can be used instead.

Users can also specify their own initial points which have the same format as in the PSUADE_IO section in the **psData** file.

The above example is located in the **Examples/OptRosenbrock** 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).

4.8   Bayesian Calibration

Let the function for numerical optimization be the function:

$$Y = F(X; a, b)$$

where X is some model design parameter and Y is the model output (assume the output is scalar simplify discussion); and a and b are parameters in the function that are not precisely known except that they fall between 0 and 1. Suppose we also have a set of observation data $D = \{X_i \tilde{Y}_i\}_{i=1}^{M}$ that may help guide the search for the true values of a and b. Suppose further that we have decided to set a to some fixed value a^* and search only for the best b

28

that fits the data set D. One way to find these values is to perform a deterministic numerical optimization. If the outputs $\tilde{Y}_i$'s are noisy, an alternative is to perform a Bayesian inference.

To perform Bayesian inference, we need to have the following:

1. a set of calibration parameters and their distributions (priors),

2. an observation data set (D in our case) together with the corresponding observation errors (let D^* be D appended with its standard deviations, i.e. $D^* = \{D\Sigma\}$), and

3. a sample (e.g. a Latin hypercube of size N) for generating a response surface to approximate F (needed because a typical Bayesian inference requires many function evaluations).

A few other decisions need to be made:

1. whether discrepancy modeling is to be included in the inference (see below), and

2. whether response surface errors are to be included in the inference.

To include discrepancy modeling, a surrogate (or response surface) is to be created that describes the differences between the observation data and the corresponding function values (model outputs). This surrogate is created from the sample set $\{X_i e_i\}_{i=1}^{M}$ where

$$e_i = Y_i - \tilde{F}(X_i; a = a^*, b = b^*); i = 1, \cdots, M;$$

and b^* is some carefully selected value of b (the best choice is the posterior mean of b but since its posterior mean is not known, it can be set to its prior mean or mode). Note that the construction of this discrepancy function is performed automatically PSUADE once this mechanism is turned on.

Let $M = 4$ is the number of observations at the design points $\{X_i\}_{i=1}^{4}$, and let the observation noise be 0.1 (standard deviation). We will specify these information to facilitate the construction of the likelihood function by creating a file, say 'mcmcFile', that contains

```
PSUADE_BEGIN
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
PSUADE_END
```

The first and last lines are markers recognized by PSUADE. The second lines specifies that there are four observations, one output, and 1 design paramter, which is parameter 1. The next four lines each consists of the data set number, the design parameter value, and the observation and error.

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 generate the Latin hypercube sample is:

```
PSUADE
INPUT
   dimension = 3
   variable   1 X   = 0.0  1.0
   variable   2 A   = 0.0  1.0
   variable   2 B   = 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, simple run the following command

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

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

```
[Linux] psuade
*** *********************************************************** ***
*** Welcome to PSUADE (version 1.7.2)                          ***
*** *********************************************************** ***
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration
psuade> load psData
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
Output 2
Enter you choice (for response surface type) ? 0
Output 3
Enter you choice (for response surface type) ? 0
Output 4
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 ('matlabmcmc2.m') will contain the posterior distributions for visualization.

There are other advanced features in PSUADE's MCMC method such as tuning MCMC parameters, generating a posterior sample, etc. when 'ana_expert' mode is on. Finally, there is a similar command called 'mcmc' which uses the actual simulator instead of response surfaces. However, this method may still be computationally expensive even though the simulator is fast because of the I/O requirement in running the simulator. A remedy for this is to compile this function into PSUADE and use 'driver = PSUADE_LOCAL' to activate that function (this will speed up evaluation by reducing the I/O overhead).

4.9   A More Comprehensive Example

Suppose we are given a simulation model with 2 uncertain parameters X_1 and X_2 such that $Y = F(X_1, X_2)$ and with some given default values for X_1 and X_2. Suppose we do not know the uncertain range for X_1 and we arbitrarily impose its uncertain range to be +/-20% of its default value. For X_2, we also impose an initial range of +/-20%, but we have another experiment that will help refine its uncertainty range. Our overall objective is to quantify the uncertainty and parameter sensitivity of this model.

The steps to achieve the objective are (in Examples/CompositeTest):

1. Compile the available experimental data for refining the uncertainty distribution of X_2 (in file 'expdata2').

2. Acquire the model ('simulator2.c') to apply Bayesian inference to refine X_2 and compile it (to become 'simulator2').

3. Put together a PSUADE input file ('psuade2.in') for Bayesian inference:

```
PSUADE
INPUT
   dimension = 1
   variable 1 X2  =  0.4   0.6
END
OUTPUT
   dimension = 1
   variable 1 Y
END
METHOD
   sampling = FACT
   num_samples = 10
   randomize
END
APPLICATION
   driver = ./simulator2
END
ANALYSIS
   diagnostics 1
END
END
```

4. Run PSUADE with 'psuade2.in' and rename psuadeData to psData2.

5. Launch PSUADE, load psData2 and apply 'rsmcmc' in command line mode to generate a posterior sample for $X2$ (turn on 'ana_expert' mode).

```
[Linux] psuade
*** ********************************************************** ***
*** Welcome to PSUADE (version 1.7.2)                         ***
*** ********************************************************** ***
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration
psuade> load psData2
load complete : nSamples = 100
               nInputs  = 2
               nOutputs = 1
psuade> ana_expert
psuade> rsmcmc
```

```
.....
===> 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 10000 and 20 to the next 2 questions.
Enter 1 and 0 next to select input 1 and terminate.
Say 'no' to discrepancy modeling.
Say 'yes' to create posterior sample.
Enter 100, 3 and 1.05 to the next 3 questions.
.....
MCMC: input   1 value at peak of likelihood = 5.600000e-01
MCMC: input   1 mean     = 7.689363e-01
MCMC: input   1 std dev = 1.300829e-01
MCMC iterations completed
MCMC: matlabmcmc2.m file has been created.
*************************************************************************
MCMC: check the MCMCPostSample file for a posterior sample.
psuade> quit
[Linux]
```

6. Convert the MCMC posterior sample to PSUADE data format by using the 'iread' in command line mode and the 'write' to a PSUADE file (say, 'sample2').

7. Generate a large sample for X_1 by running psuade psuade1.in (generate sample only and no simulation) and rename 'psuadeData' to 'sample1'.

8. Now we need to combine the two different distributions from two different inputs - one drawn from some standard distribution ('sample1') and the other from the Bayesian posterior distribution ('sample2'). Concatenation of the two 1-parameter samples 'sample1' and 'sample2' via 'rand_draw2' will give a 2-parameter sample to be propagated through the simulation model (or its response surface).

```
[Linux] psuade
*** ********************************************************** ***
*** Welcome to PSUADE (version 1.7.2)                         ***
*** ********************************************************** ***
PSUADE - A Problem Solving environment for
         Uncertainty Analysis and Design Exploration
psuade> rand_draw2
Enter name of the first file : sample1
Enter name of the second file : sample2
Size of the sample to be drawn : (1-2000000) 100000
Store random sample to : (filename) newSample
```

```
        psuade> quit
        [Linux]
```

9. Prepare the original 2-parameter simulation model ('simulator.c') by compiling it (to become 'simulator').

10. Propagate the 2-parameter sample through the simulator by setting the driver field in 'newSample' to be 'simulator' and running PSUADE on 'newSample' (Alternatively, if 'simulator' is expensive to run, replace it with a small sample and a response surface type.) After the runs have been completed, rename 'psuadeData' (e.g. to 'pdata').

11. Launch PSUADE to compute uncertainties and sensitivities (e.g. 'ua'). Turn on 'ana_expert' mode for Matlab graphics. You can also try 'me', 'tsi', and/or 'ca'.

```
        [Linux] psuade
        *** ********************************************************** ***
        *** Welcome to PSUADE (version 1.7.2)                         ***
        *** ********************************************************** ***
        PSUADE - A Problem Solving environment for
                 Uncertainty Analysis and Design Exploration
        psuade> load pdata
        psuade> ua
        .....
        psuade> me
        .....
        psuade> quit
        [Linux]
```

12. If desired, compare the uncertainty distribution with the sample set without using experimental data (by running PSUADE on 'psuade.in') to assess how the use of experimental data affects the output uncertainty.

5   Summary

PSUADE is intended to be a general-purpose toolkit for uncertainty quantification. Many enhanced features have been incorporated based on our experiences with its 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.