



PSUADE Short Manual
(Version 1.6)

Charles Tong
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94551-0808
September, 2013

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_v1.6.x.tar.gz`
2. `untar xvf PSUADE_v1.6.x.tar`
3. `cd PSUADE_v1.6.x`
4. For adding Gaussian process capability, consult the developers on how to install it first.
5. Set the 'FC' environment variable to your preferred Fortran 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)
10. To verify correct installation, do: `cd Examples/Bungee`
11. `cc -o simulator simulator.c -lm`
12. `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 of statistics and all input and output data 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 density 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) 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 to 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 ('ANALYSYS' section).

PSUADE supports many types of analysis such as

- Response surface analysis (MARS, polynomial regression, Kriging, etc.)
- Parameter screening (several such methods)
- Hypothesis testing
- Correlation analysis
- Main effect (first order sensitivity) analysis
- Interaction (second order sensitivity) analysis without
- Group and total sensitivity analysis
- Bayesian calibration
- Deterministic 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 Fortran compiler>
[Linux] cmake ..
      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 many 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] cmake ..
  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 this my case cmake has picked up the wrong cc:
  CMAKE_C_COMPILER          /usr/bin/cc
  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 following 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 L 0.0 1.0
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 (joint PDF for normal and lognormal distributions are also supported). If the PDF is triangular (T), the mean and width are to be provided. Other available distributions are beta, exponential, gamma, Weibull, and user-provided samples.

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
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 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.

Some of the other sampling methods available are (refer to the theory manual 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. `opt_driver` points to the executable for numerical optimization. Again, the user code can just be a simple program or a complex super-script performing postprocessing, 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 output file (3 output variables) 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
limited_launch_only  - same as launch_only but only launch max_parallel_jobs jobs
gen_inputfile_only  - generate all sample files only (no code runs)
```

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 moments of the sample on output number 1).

```
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
RMSoSobol1     - response surface-based first order sensitivity analysis
RMSoSobol2     - response surface-based first second sensitivity analysis
RMSoSobolG     - response surface-based group main effect analysis
RMSoSobolTSI   - response surface-based total sensitivity analysis

```

When response surfaces are used together with the selected analysis method, the response surface type 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
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 **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. 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. Interactive mode is activated by just 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):

```
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 file name is `psData`). To create scatter plots for the data in the `interactive` mode, do:

```
[Linux] psuade
*** *****
*** Welcome to PSUADE (version 1.6.0) ***
*** *****
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.

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 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 Confidence Intervals

Let the function be given by:

$$Y = F(X_1, X_2) = X_1 + X_2 + X_1X_2 + X_1^2, \quad X_i \in [0, 1]$$

and we would like to compute the few moments of this function assuming the inputs are uniformly distributed. We select Latin hypercube sampling with a sample size of 1000. The jobs are to be run in sequential mode. The corresponding PSUADE input file `psuade.in` is:

```

PSUADE
INPUT
    dimension = 2
    variable 1 X1 = 0.0 1.0
    variable 2 X2 = 0.0 1.0
END
OUTPUT
    dimension = 1
    variable 1 Y1
END
METHOD
    sampling = LH
    num_samples = 1000
    randomize
END
APPLICATION
    driver = ./testmain.py
END
ANALYSIS
    analyzer method = Moment
    diagnostics 3
END
END

```

The driver program can be in any language provided that it is executable. Our example uses Python to simulate the above function (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])
lineIn = infile.readline()
ncols = string.split(lineIn)
X1 = eval(ncols[0])
lineIn = infile.readline()
ncols = string.split(lineIn)
X2 = eval(ncols[0])
infile.close()
Y = X1 + X2 + X1 * X2 + X1 * X1
outfile = open(sys.argv[2], "w")

```

```
outfile.write("%e \n" % Y)
outfile.close()
```

After these files have been prepared (These files can be found in the `Examples/UserExample` directory. Make sure the python link in the `testmain.py` file is correct, and that `testmain.py` has execute permission), run PSUADE with:

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

and, at the completion of the runs, the moment information will be displayed and the `psuadeData` file will also be created for use in further analysis.

4.2 Basic Regression Analysis

Suppose we use the above function again and we would like to perform a quadratic regression analysis. We can reuse the data in the `psuadeData` file from the last exercise (let's copy this file to `psData` so that it will not be overwritten by PSUADE). To do so, the `ANALYSIS` section in `psData` has to be modified to:

```
PSUADE
... data ...
INPUT
...
END
OUTPUT
...
END
METHOD
...
END
APPLICATION
...
END
ANALYSIS
analyzer method = RSFA
analyzer rstype = quadratic
END
END
```

This modified file is to be run with PSUADE via:

```
[Linux] psuade psData
```

and regression analysis results (the regression coefficients and R^2) will be displayed.

4.3 Screening for Important Inputs

A useful design and analysis tool in PSUADE is parameter screening using the Morris method, which is an effective variable selection method when the number of inputs is large (say, > 15). The corresponding PSUADE input file should be set up as (say, for a 20-dimension problem in the Examples/MOATTest 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 Y1
END
METHOD
  sampling = MOAT
  num_samples = 210
  randomize
END
APPLICATION
  driver = ./morris_simulator
END
ANALYSIS
  analyzer method = MOAT
  diagnostics 3
END
END
```

Here the sample size should be a multiple (usually 10) of $K + 1$ where K 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 copied to the `psData` file):

```
[Linux] psuade
*** ***** **
*** Welcome to PSUADE (version 1.6.0) **
*** ***** **
```

```

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.m
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).

Screening can also be done via other methods such as the Delta method and approximate modeling method (use MARS to do the ranking) which will not be covered in detail here.

4.4 Response Surface Analysis

Since main effect analysis requires many model evaluations to ensure sufficient accuracy, it may not be feasible for computationally expensive functions. However, when the input-output relationship is well-behaved (namely, smoothly varying), response surface methods can be used to create a cheap surrogate for the actual function. In the following, we describe how to use PSUADE to create response surfaces and how to use response surface for quantitative analysis. The first step in creating a response surface is to select a sampling method. Typically, a space-filling sample such as quasi-Monte Carlo or maxi-min Latin hypercube is adequate (PSUADE also has advanced capabilities to create response surfaces using adaptive sampling techniques). The input file (`psuadeRS.in` in the `Examples/UserExample` directory) for sampling is:

```

PSUADE
INPUT
  dimension = 2
  variable 1 X1 = 0.0 1.0
  variable 2 X2 = 0.0 1.0
END
OUTPUT
  dimension = 1
  variable 1 Y1
END

```

```

METHOD
    sampling = LPTAU
    num_samples = 300
END
APPLICATION
    driver = ./testmain.py
END
ANALYSIS
    diagnostics 1
END
END

```

Here we choose the LP- τ quasi-Monte Carlo method based on the Sobol' sequence with a sample size of 300. This sample is then evaluated via

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

At the conclusion of the runs, a file named `psuadeData` will be created which contains the data set (inputs and outputs). Next, rename this file to, for example, `psData`. In the next step, we check to see if this data set can be fit with some surface-fitting schemes. A popular scheme is the multivariate adaptive regression splines (MARS) developed by Jerome Friedman. To check the goodness of the fit, we launch the PSUADE command line mode and the details of a session is given below:

```

[Linux] psuade
*** ***** **
*** Welcome to PSUADE (version 1.6.0) ***
*** ***** **
PSUADE - A Problem Solving environment for
        Uncertainty Analysis and Design Exploration
psuade> load psData
load complete : nSamples = 300
                nInputs  = 2
                nOutputs = 1
psuade> rscheck
Which response surface tool to use ?
Available response surface tools:
0. MARS
1. Linear regression
.....
12. MARS with bootstrap aggregating (bagging)
Enter you choice ? 0
.....
RSFA: L 0: interpolation error on training set =

```

```

Lin error = 2.431e-04 (unscaled), 3.540e-04(scaled)
Avg error = -4.150e-08 (unscaled), 4.574e-05(scaled)
RMS error = 5.048e-04 (unscaled), 9.115e-04(scaled)
Max error = 5.338e-03 (unscaled, BASE=1.004e+00)
Max error = 6.774e-03 ( scaled, BASE=1.333e-01)
Based on 300 training points (total=300).
Perform cross validation ? (y or n) y
Enter the number of groups to validate : (2 - 300) 30
.....
.....
psuade> quit
[Linux]

```

The command `rscheck` first asks for which response surface tool to select and then it displays the interpolation error statistics (error checking on the original data set). If cross validation is selected, the number of groups is to be specified (that is, divide the sample into k groups, hold out one group at a time and compile prediction error statistics). Subsequently, cross validation will be performed, a summary of error statistics will be displayed and an error file (`RSFA_CV_err.m` or `RSFA_CV_err.sci`) will be created to be plotted with `Matlab` or `Scilab`.

If `rscheck` shows that the sample is not good enough to represent the model input-output relationship (the scaled RMS error or the Max error is large), more sample points should be added to the original sample until sufficient accuracy is attained. Or, a different response surface can be applied to use whether it gives a better fit.

Once the sample and the corresponding response surface scheme are deemed to be satisfactory, the data set (say, again in the `psData` file), it can be used for main effect analysis. The setup is the same as in the last section, except that the definition of `driver` is different in this case (this is the `psuadeRSME.in` file in `Examples/UserExample`):

```

PSUADE
INPUT
  dimension = 2
  variable 1 X1 = 0.0 1.0
  variable 2 X2 = 0.0 1.0
END
OUTPUT
  dimension = 1
  variable 1 Y1
END
METHOD
  sampling = LH
  num_samples = 1000
  num_replications = 50

```

```

    randomize
END
APPLICATION
    driver = psData
END
ANALYSIS
    analyzer method = MainEffect
END
END

```

Again, simply run the following command

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

and main effect results will be displayed.

More advanced features for main effect analysis include the handling of correlation inputs (governed by some inequality conditions) and recursive analysis, which are not described in this document.

4.5 Main Effect and Total Sensitivity Analysis

Main effect analysis studies the first order sensitivities of individual input parameter based on variance decomposition. The sensitivity indices can be computed using replicated Latin hypercube sampling with the main effect analysis, the FAST sampling and analysis, or direct numerical integration using response surfaces. In the following example, we describe the use of the replicated Latin hypercube approach. The input file is given as follow (this can be found in the `Examples/UserExample/psuadeME.in` file):

```

PSUADE
INPUT
    dimension = 2
    variable 1 X1 = 0.0 1.0
    variable 2 X2 = 0.0 1.0
END
OUTPUT
    dimension = 1
    variable 1 Y1
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 with $1000/50 = 20$ levels. To run this analysis, go to `Examples/UserExample` and issue the following (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.

There are two other alternatives to perform main effect analysis:

1. Use response surfaces in place of the simulator; and
2. Use direct numerical integration on the response surface.

The first alternative can be used simply by replacing ‘`driver = simulator`’ in the PSUADE input file with ‘`driver = psData`’ where ‘`psData`’ is a PSUADE data file containing a set of sample inputs and outputs.

The second alternative can be performed by first generating and running a sample (say the result has been put into a file called ‘`psData`’). The next step consists of launching PSUADE’s command line interpreter:

```

[Linux] psuade
*** ***** **
*** Welcome to PSUADE (version 1.6.0) ***
*** ***** **
PSUADE - A Problem Solving environment for
        Uncertainty Analysis and Design Exploration
psuade> load psData
load complete : nSamples = 100
                nInputs  = 2
                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.

Total sensitivity analysis can be performed in a similar fashion by using ‘rssoboltsib’ instead. Also, more sophisticated quantitative sensitivity analyses involving input correlations are available. There are two ways to include input correlations (input constraints) into a main effect analysis:

1. If you are using replicated Latin hypercube (on the simulator or on response surface), the steps are: (a) generate the sample; (b) apply input constraints to filter out infeasible sample points; (c) run the ‘reduced’ sample; and (d) launch PSUADE and run ‘me’.
2. You can also use numerical integration (‘rssobol1b’). You will also need to turn on the ‘rs_constraint’ command in your sample file. 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 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).

4.7 Bayesian Calibration

Let the function for numerical optimization be the function:

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

where X is the design parameter; and a and b are parameters in the function that are not precisely known except that they fall between 0 and 1. We do, however, have a set of data points $\{X_i Y_i\}_{i=1}^N$ in an effort to find the true values of a and b . One way to find these

values is to perform a deterministic numerical optimization. If the outputs Y_i 's are noisy, an alternative is to perform a Bayesian inference. A few things need to be set up for this inference:

1. A function that returns the errors between prediction and data given any possible a and b :

$$e_i = Y_i - F(X_i); i = 1, \dots, N; \quad \text{and}$$

2. Information that are needed to create a likelihood function in the Markov Chain Monte Carlo (MCMC) iterations.

A skeleton for the function is (which is compiled into an executable called 'simulator'):

```
#include <stdio.h>
main(int argc, char **argv)
{
    int i, m, N;
    double AB[2];
    /* set up Y[i]'s and define the function F */
    fopen(argv[1], "r");
    fscanf(fp, "%d", &m);
    for (i = 0; i < m; i++) fscanf(fIn, "%lg", &AB[i]);
    fclose(fp);
    fopen(argv[2], "w");
    for (i = 0; i < N; i++) fprintf(fp, " %e\n", Y[i] - F(X[i],AB));
    fclose(fp);
}
```

Let $N = 4$ is the number of data point and let the noise in the data has a standard deviation of 0.1. We will specify these information to facilitate the construction of the likelihood function by creating a file, say 'mcmcFile', that contains

```
PSUADE_BEGIN
1 0 4
1 0 0.1 0 0.1 0 0.1 0 0.1
PSUADE_END
```

The first and last lines are markers recognized by PSUADE. The second lines specifies that there is a set of data with 4 elements and the second number indicates that there is no design parameter (design parameters are useful when discrepancy modeling is included in the Bayesian inference).

Since Bayesian inference generally requires many simulations, often the expensive simulator is replaced by an inexpensive surrogate (or response surface). We follow the steps above in creating the response surfaces for the N error outputs from the simulator.

The PSUADE input file ('psuadeRS.in') to generate the Latin hypercube sample (of size 100) for response surfaces is:

```

PSUADE
INPUT
  dimension = 2
  variable  1 A  = 0.0  1.0
  variable  2 B  = 0.0  1.0
END
OUTPUT
  dimension = 4
  variable 1 Y1
  variable 2 Y2
  variable 3 Y3
  variable 4 Y4
END
METHOD
  sampling = LH
  num_samples = 100
END
APPLICATION
  driver = ./simulator
END
ANALYSIS
  diagnostics 1
END
END

```

Again, simple run the following command

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

and then move the result data file 'psuadeData' to, say, 'psData'. After the preparation steps have been completed, Bayesian inference can be launched by:

```

[Linux] psuade
*** ***** **
*** Welcome to PSUADE (version 1.6.0) **
*** ***** **
PSUADE - A Problem Solving environment for
          Uncertainty Analysis and Design Exploration
psuade> load psData
load complete : nSamples = 100
                nInputs  = 2
                nOutputs = 1
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
.....
MCMC PHASE 1: BURN In
 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
MCMC PHASE 1 completed
.....
MC PHASE 2 completed
*****
MCMC: final matlabmcmc.m file has been created.
=====
MCMC: matlabmcmc2.m file (2-input analysis) is ready.
*****
psuade> quit
[Linux]

```

If the inference is completed successfully, the two Matlab (or Scilab) files will contain the posterior distributions for visualization.

There are many advanced features in PSUADE's MCMC method such as modeling discrepancy, tuning MCMC parameters, generating a posterior sample, and including response surface errors. These features can be selected by turning on the 'ana_expert' mode (by entering 'ana_expert' on the command line before running 'rsmcmc'). Finally, there is a similar command called 'mcmc' which uses the actual simulator instead of response surfaces. However, this method may 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).

4.8 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_1 , 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:

```
[Linux] psuade
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 put the result in 'psData2'.
5. Apply MCMC to generate a posterior sample for X_2 (turn on 'ana_expert' mode).

```
[Linux] psuade
*** ***** **
*** Welcome to PSUADE (version 1.6.0) ***
*** ***** **
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
.....
Say 'no' to the first question.
==> 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 50000, 50000, and 20 to the next 3 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 to the next question.
.....
MCMC PHASE 1: BURN In
 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
MCMC PHASE 1 completed
.....
MC PHASE 2 completed
.....
*****
MCMC: check the MCMCPostSample file for a posterior sample.
psuade> quit
[Linux]
```

6. Convert the MCMC posterior sample to PSUADE data format by removing the first and last lines; and replacing the second line with '100000 1 0' (100000 points with 1 input and no output). Then read this file in using 'read_xls MCMCPostSample' and write it to 'sample2' using the 'write' command.
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. Concatenate the two 1-parameter samples 'sample1' and 'sample2' to be a two-parameter sample using the 'rand_draw2' command.

[Linux] psuade

*** ***** **

```

*** Welcome to PSUADE (version 1.6.0) ***
*** ***** **
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. Set the driver in 'newSample to be simulator' and run PSUADE on 'newSample'. After that rename 'psuadeData' to 'psData'. (Alternatively, if 'simulator' is expensive to run, replace it with a small sample and a response surface type.)
11. Launch PSUADE to compute uncertainties and sensitivities (turn on 'ana_expert' mode for Matlab graphics).

```

[Linux] psuade
*** ***** **
*** Welcome to PSUADE (version 1.6.0) ***
*** ***** **
PSUADE - A Problem Solving environment for
        Uncertainty Analysis and Design Exploration
psuade> load psData
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').