PSUADE Reference Manual (Version 1.7)

Charles Tong
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94551-0808
August, 2015

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, Bayesian calibration, etc. It comprises three major components: a suite of sampling methods, a job execution environment, and a collection of analysis (including numerical optimization) tools. In addition, it provides a command line interpreter (in fact there are two command line interpreters - one serial and one parallel) to allow users to interactively perform some of the analyses. 'Command interpreter' mode is activated by calling PSUADE without argument (or with '-mp' for parallel mode); that is,

```
[Linux] psuade [-mp]
```

Morever, PSUADE also can be used as a library with functions that can be called directly by user programs.

This document first describes some of the available commands for the command line (or interactive) mode (most of the commands can be found by issuing the 'help' command). Afterward, a detailed list of PSUADE functions that can be called by user program will be given.

2 PSUADE Command Line Interpreter

The 'interactive mode' commands can be divided roughly into the following sections.

2.1 Commands for Read/Write from/to Files

To provide enhanced capabilities and flexibility, PSUADE interacts with users at different levels, often via file input/output. For some of the file formats used, enter the command showformat.

load <file> : where <file> is a PSUADE data file.

This command loads a sample file in PSUADE data format to the resident memory. When loaded successfully, the sample data are ready to be analyzed or manipulated. You can, for example, use sinfo or listall to examine what you have loaded.

loadmore <file> : where <file> is a PSUADE data file.

This command should be used to add another sample (in PSUADE data format) on top of the sample previously loaded by the load command (issuing a 'load' again will erase the current sample in PSUADE's local memory.) In order for this command to be successful, the added sample must have the same number of inputs and outputs as the loaded sample.

write <file> : where <file> is a PSUADE data file.

This command writes a sample which has been loaded and manipulated previously to another file in the standard PSUADE data format. You will be asked to save the sample with just one output or all outputs.

nwrite <file> : where <file> is a data file.

This command writes only the unevaluated sample points (those with sample state = 0 or if any of the outputs is undefined) to the specified file.

iread <file> : where <file> is a data file.

This command reads a set of sample inputs from a file, which should have the first line having the keyword 'PSUADE_BEGIN' (this is now optional in the latest version), the second line specifying the sample size and number of inputs, subsequent lines specifying the input values of each samples, and finally the optimal last line having the keyword 'PSUADE_END' (use showformat to see details of the data format).

iwrite <file> : where <file> is a data file.

This command writes ONLY the sample inputs to a file which has the data format that can be read by the iread command. Use showformat to see details of the data format.

owrite <file> : where <file> is a data file.

This command writes ONLY the sample outputs to a file which will have the first line having the keyword 'PSUADE_BEGIN', the second line specifying the sample size and number of outputs, subsequent lines specifying the sample numbers and the output values, and finally the last line having the keyword 'PSUADE_END'.

read_std <file> : where <file> is a data file.

This command is similar to the load command except that this command reads the sample data in a different format - the standard format - which has the first line specifying the sample size, the number of inputs, and the number of outputs; and subsequent lines specifying each sample input and output values (one sample point per line). An example is:

```
4 2 1
0.0 0.0 0.0
1.0 0.0 1.0
0.0 1.0 1.0
1.0 1.0 2.0
```

write_std <file> : where <file> is a data file.

This command writes the sample in PSUADE's local memory to the specified file in standard format (see read_std for the format).

write_matlab <**file**> : where <file> is a data file.

This command writes to a file the sample previously loaded into the PSUADE memory in Matlab format. The sample input matrix will be stored as X and the sample output matrix will be stored as Y.

read_xls <**file**> : where <file> is a data file.

This command reads in a sample which has been specified in a format that can be created from an Excel spreadsheet. The data format has the first line specifying the sample size, the number of inputs, and the number of outputs; and subsequent lines specifying the sample number and sample input and output values (one sample point per line). An example is:

```
4 2 1
1 0.0 0.0 0.0
2 1.0 0.0 1.0
3 0.0 1.0 1.0
4 1.0 1.0 2.0
```

read_csv <**file**> : where <file> is a special CSV file.

This command reads in a sample which has been specified in a format that can be created from an Excel spreadsheet. The data has the following format (where the first line may optionally be used to specify which columns are inputs and which are outputs; the second line may optionally be used to specify the input and output names; and subsequent lines are the sample input and output values separated by commas. If the first line is not specified, users will be asked for the number of inputs and outputs):

```
input,input,input,output,output (optional line)
I1, I2, I3, O1, O2 (optional line)
0.0,0.0,0.0,0.0,0.0
1.0.0.0.1.0.1.0,2.0
0.0.1.0.1.0,2.0,3.0
1.0.1.0.2.0,3.0,4.0
```

write_xls <**file**> : where <file> is a data file.

This command writes a sample previously loaded (into PSUADE local memory) to a file in Excel format.

write_ultra <**file**> : where <file> is a data file.

This command writes a sample previously loaded (into PSUADE local memory) to a file in the 'ultra' format. It can only handle samples which have two inputs. If you do not know what 'ultra' format is, most likely you do not need to use this command.

update <file> : where <file> is a standard PSUADE data file.

This command updates the sample previously loaded with the sample from the file <file>. The sample points that have been evaluated in <file> will be used to update the corresponding (same) unevaluated sample points in the local memory.

iadd <file> : where <file> is a standard PSUADE data file.

Suppose you have a PSUADE data file, and you would like to add a few more inputs to this file. You will have to prepare another PSUADE data file (<file>) which has the same sample size but different sample inputs. For example, if you have a sample with 2 inputs loaded in PSUADE local memory and another sample with 3 inputs in some data file, then after issuing this command, the total number of inputs in PSUADE local memory will be 5 (the new inputs will be appended to the end of the input list). You can write this 'appended' sample using write. This command is useful if you would like to, for example, add a few dummy input variables in your sample data file. Note that no sample outputs will be modified.

oadd <file> : where <file> is a standard PSUADE data file.

This command is analogous to iadd. This command only checks for the same sample size in both samples and it does not check that the two sets of sample inputs are exactly the same (so beware).

iadd1 : create a new sample input.

This command should be used after a sample has been loaded into PSUADE local memory. It creates a new sample input to be appended to the current input set. The values of this new input in the sample are drawn randomly from $[0, 1]$. This command is useful in adding a nugget input variable to the existing sample.

oadd1 : create a new sample output.

This command should be used after a sample has been loaded into local memory. It creates a new sample output to be appended to the current output set. The values of this new output in the sample are drawn randomly from $[0, 1]$. This command is useful in preparing for the 'oop' command.

ireplace <file> : where <file> is a data file.

This command is similar to iadd except that instead of adding more inputs, it replaces ONE of the current inputs by another input in <file>. The format of the data file should be:

```
PSUADE_BEGIN
4
0.0
1.0
```

```
2.0
3.0
PSUADE_END
```

oreplace <file> : where <file> is a data file.

This command is similar to ireplace except that it operates on ALL sample outputs. The format of the data file should be:

```
PSUADE_BEGIN
4 2
0.0 1.0
1.0 4.0
2.0 3.0
3.0 2.0
PSUADE_END
```

This command may be useful if you run the sample yourself and you would like to update your PSUADE sample file with the actual simulation outputs.

splitsample : split the currently loaded sample into two subsets.

This command splits the sample in local memory into two subsets. It will ask for the sample size of the first subset. The two subsets are stored in psuadeSplit1 and psuadeSplit2, respectively. Note: the sample in PSUADE's local memory will be destroyed after this command, so you need to re-load for more analysis.

2.2 Commands for Basic Statistics

ua : uncertainty analysis on the sample in local memory.

This command should be used after a sample has been loaded into local memory. It displays basic statistics information such as mean, standard deviation, skewness and kurtosis. This command will also create Matlab distribution plots.

ca : correlation analysis on the sample in local memory.

This command should be used after a sample has been loaded into local memory. It displays correlation information such as Pearson, Spearman and Kendall coefficients. Pearson coefficients are useful sensitivity indicators when the input-output relationship is more or less linear. For nonlinear but monotonic functions, the Spearman coefficients are better indicators.

me : perform variance-based main effect analysis.

This command should be used after a sample has been loaded into local memory. It displays first-order Sobol' indices for the sample. Ideally, this command works best

with replicated Latin hypercube samples. It will still work otherwise, but the results are much less accurate when the sample size is small. This command is intended for computationally inexpensive functions that you can evaluate many times (hundreds of thousands) quickly. Otherwise, response surface-based methods are better.

ie : perform variance-based two-way interaction effect analysis.

This command should be used after a sample has been loaded into local memory. It displays second-order Sobol' indices for the sample. Ideally, this command works best with replicated orthogonal array samples. It will still work otherwise, but the results will be much less accurate when the sample size is small. This command is intended for computationally inexpensive functions that you can evaluate many times (hundreds of thousands) quickly. Otherwise, response surface-based methods are better.

tsi : perform variance-based total sensitivity analysis on the raw sample which has loaded in the local memory.

This command should be used after a sample has been loaded into local memory. It requires a very large sample to be accurate and currently it only works with low input dimensions (less than 20). This command is intended for computationally inexpensive functions that you can evaluate many times (hundreds of thousands) quickly. Otherwise, response surface-based methods are better.

sobol : Sobol' first and total order sensitivity analysis on the sample in local memory.

This command should be used after a sample has been loaded into local memory. It works only on the Sobol' sampling design.

fast : total order sensitivity analysis on the sample in local memory.

This command should be used after a sample has been loaded into local memory. It works only on the FAST (Fourier Amplitude Sampling Test) designs.

anova : analysis of variation on the sample in local memory.

This command should be used after a sample has been loaded into local memory. It performs and displays the results of an analysis of variation on the loaded sample.

1stest : perform one-sample tests.

This command performs one-sample tests on a sample (not the loaded sample) such as the Chi-squared test or the distribution test (search for mean and standard deviation for the sample). The sample data in the requested format should be given when prompted.

2stest : perform two-sample tests.

This command performs two-sample tests on a sample (not the loaded sample) such as the Student T-test (do two normally distributed populations differ?), the Mann-Whitney test (do the two sample have the same probability distribution?), and Kolgomorov Smirnov test (does the sample come from a population with a specific distribution with a given mean and standard deviation?). You will have to enter 2 data files in the format instructed by this command.

gendist : create a data set using the specified PDFs.

This command creates a ONE-INPUT sample based on the selected probability distribution. The result sample is written to a file 'sample1D' which has the first line specifying the sample size and subsequent lines specifying the sample input values.

pdfconvert : convert a sample based on its probability distribution.

This command should be used after a sample has been loaded into local memory. It converts the sample inputs (assumed uniform to begin with) based on their probability distributions (defined in the INPUT section of the data file itself) and replaces the sample inputs with the converted values. The results can be written to a PSUADE data file using write.

rand_draw : draw a sample from the loaded sample with replacement.

This command should be used after a sample has been loaded into local memory. It draws from the loaded sample a bootstrapped sample with replacement. The results will be written to a PSUADE data file with a user-given file name.

rand_draw2 : draw a sample from two samples with two different sets of inputs.

When initiated, this command will request two sample files in PSUADE data format and then randomly draws a new sample of a given size having the set of inputs from both files. The results will be written to another PSUADE data file with a user-given file name.

gensample : generate a sample based on parameter information in the INPUT block of the loaded sample.

This command is useful when working with the rsuab command, which requires a user to enter a sample file for propagation through the response surface.

cdf_lookup : look up cumulative probability given a value. This command is useful in computing the probability mass within an interval. For example, given $X1$ and $X2 > X1$, one can compute the probability between $X1$ and $X2$ by looking up probabilities for both (say, $P1$ and $P2$) and computes the difference $(P2 - P1)$.

2.3 Commands for Parameter Screening (Input Dimension Reduction)

lsa : perform local sensitivity screening analysis.

This command should be used after a sample has been loaded into local memory. It displays screening information based on a gradient analysis. This command works only with LSA samples.

moat : perform Morris screening analysis.

This command should be used after a sample has been loaded into local memory. It displays screening information such as modified mean and standard deviations of the Morris gradients. Analysis results will also be saved in a matlab/scilab file for graphical display. This command works only with MOAT samples.

moatmo : similar to the 'moat' command but it analyzes all outputs and displays the results together.

ff : perform fractional factorial screening analysis.

This command should be used after a sample has been loaded into local memory. It displays screening information based on a fractional factorial analysis. This command works only with fractional factorial samples.

mars_sa : perform MARS screening analysis.

This command should be used after a sample has been loaded into local memory. It display screening information based on the MARS importance ranking. This information will NOT be generated on some platform where MARS outputs are suppressed or when there is an error in MARS analysis.

gp_sa : perform Gaussian process screening analysis.

This command should be used after a sample has been loaded into local memory. It displays screening information based on importance ranking with Gaussian process (if the Gaussian process package has been installed) or Kriging.

delta_test : perform Delta test for screening input parameters.

This command should be used after a (random or quasi-random) sample has been loaded into local memory. This test involves numerical optimization and so it can be computationally intensive for large sample size and large number of inputs (so this test is good for moderate number of inputs with moderate sample size). During the optimization, the configurations and their corresponding objective values are displayed. Then the ranking results will be displayed. Finally, there is a final test that incrementally adds the identified important parameters to the list and computes the objective values starting from the most important parameter. You can plot the objective values. If you observe a 'tub'-like curve, the recommended parameter partitioning (into important and unimportant ones) is around the bottom of the curve.

sot_sa : perform sum-of-trees screening analysis.

This command should be used after a sample has been loaded into local memory. It displays screening information based on a sum-of-trees analysis. This method is good for high input dimension and large sample sizes.

pca : perform principal component analysis on the sample outputs.

This command should be used after a sample has been loaded into local memory. PSUADE performs singular value decomposition on the sample outputs (normalized by mean and standard deviation of each output). Specifically, given the output matrix Y, the following decomposition is performed:

$$Y = USV^T$$

where Y is a matrix with N rows (sample size) and M columns (number of outputs), U is a matrix of size $M \times N$ (left singular vectors), S is a diagonal matrix of size $N \times N$, and V is a matrix of size $N \times N$ (right singular vectors). Upon exit from this command, the file 'psPCA.out' will contain the projection of the sample outputs onto the principal components V (that is, $Y \times V$ or $U \times S$). If you decide to use these principal components, you can use 'oreplace' to insert the file 'psPCA.out' to your PSUADE sample file.

2.4 Commands for Response Surface Analysis

svmfind : find the best parameter setting for SVM RBF response surface.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. The support vector machine (SVM) has two parameters if the RBF kernel is selected. This command tunes these parameters to obtain the best training errors.

rscheck : check the quality of the response surface.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a response surface and then checks the training errors (to appear in the file 'psuade_rsfa_err.m' if print level has been set to 4). Optionally, a cross validation can also be performed. To do that, you will need to select how many groups (that is, the k in k-fold cross validation). At the end of cross validation, the file 'RSFA_CV_err.m' will be created for visualizing the distribution of cross validation errors. When this file is run in Matlab, two plots will be displayed. Ideally, the error distribution should center around zero with very small spread. Also, on the right plot, ideally all sample points lie on the diagonal line.

rstest : this corresponds to a suite of tests for the goodness of a response surface created from the loaded sample.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It builds a response surface for the loaded sample and may ask for a test set (in PSUADE data format) to compute the interpolation errors. There are four different types of response surface test:

- rstest_ts: use the training test
- rstest_hs: hold out test set
- rstest_cv: k-fold cross validation
- rstest_gt: generalization test

rscreate : create a response surface from the loaded sample.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It builds a response surface for the loaded sample. This command should be used with 'rseval'.

ivec_create : create a single sample point in local memory.

You can create a local sample point in the local memory for use with 'rscreate'. Use 'ivec_modify' (or 'vmodify') and 'ivec_show' (or 'vshow') to manipulate and read this internal sample point. The initial values are the mid points in each input.

ivec_modify : modify the content of the sample vector (an internal vector for storing a single sample point) in local memory.

This command is to be used after an internal sample vector has been created using 'ivec_create' to modify the content of the sample register.

ivec_show : display the content of the sample vector in local memory.

This command is to be used after an internal sample vector has been created using 'ivec_create' to show the current content of the register.

rseval : evaluate a data point from the response surface built from the loaded sample.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory and after recreate has been called. You can evaluate a single sample point from the local sample vector (created by 'ivec_create' or from a file. The file should have PSUADE_BEGIN as its first line, the number of inputs as the second line, the input values in subsequent lines (each line begins with the input number followed by the input value), and PSUADE_END as the last line.

rseval_m : this command uses PSUADE as a response surface server.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It should be used after 'rscreate' has been called. This command waits for users to put the new sample point in a file (called 'rsevalDataIn.x'), evaluates the sample point, writes the result to a file (called 'rsevalDataOut.x'), and wait for

another point or a termination signal (the presence of the 'psComplete' file in the current directory).

rs_splot : create scatter plots from a response surface.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It is similar to splot except that the 'large' sample is taken from sampling the response surface instead of from the original sample. The scatter plots are created in the Matlab file 'matlabrssp1.m'.

rstgen : create a sample for response surface test.

Response surfaces usually give poor interpolations at the corners or edges or faces of the parameter space. This command creates a sample at the corners of the parameter space, and is to be evaluated and then used with 'rstest' for validating the response model at the corners. Depending on the number of inputs, we may choose full factorial or fractional factorial sample design.

rsvol : calculate volume of the constrained space.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. This command builds a response surface for the loaded sample, creates a large sample, and tabulates the number of sample points that satisfy a given constraint. The percentage of 'feasible' sample points is the percentage of the feasible space in the entire parameter space.

rsint : perform numerical integration.

This command should be used after a PSUADE sample has been loaded into local memory. It uses the sample data to build a response surface and then compute the integral under the response curve within the parameter range.

2.5 Commands for Quantitative Sensitivity Analysis

rsua : perform uncertainty analysis on a fuzzy response surface.

This command should be used after a PSUADE sample has been loaded into local memory. It performs uncertainty analysis on the response surface (built from the loaded sample).

rsuab : perform uncertainty analysis on a response surface.

This command should be used after a PSUADE sample has been loaded into local memory. It performs uncertainty analysis on the response surface (built from the loaded sample) with bootstrapped aggregation and optional discrepancy function. A sample will be requested from users to be used to propagate uncertainties through the response surface. This sample can be generated using 'gensample'.

rsmeb : perform main effect analysis on the loaded sample.

This command should be used after a PSUADE sample has been loaded into local memory. This analysis is performed on a fuzzy response surface with bootstrapped aggregation. This method performs the same analysis as 'rssobol1b' but uses a different algorithm.

rsieb : perform second-order analysis on the loaded sample.

This command should be used after a PSUADE sample has been loaded into local memory. This analysis is performed on a fuzzy response surface with bootstrapped aggregation. This method performs the same analysis as 'rssobol2b' but uses a different algorithm.

rssobol1b : perform variance-based main effect analysis on a response surface.

This command should be used after a PSUADE sample has been loaded into local memory. It displays first-order Sobol' indices for the response surface built from bootstrapped samples in the local memory (instead of from the raw sample in 'me'). Constraints can be imposed by using 'rs_constraint' in the ANALYSIS section of the sample file. Make sure to validate your response surfaces before performing this analysis. This command does not request response surface type but it uses the 'rstype' in the loaded sample.

rssobol2b : perform variance-based second-order analysis on a response surface.

This command should be used after a PSUADE sample has been loaded into local memory. It displays second-order Sobol' indices for the response surface built bootstrapped samples from the loaded sample (instead of from the raw sample in 'ie'). Constraints can be imposed by using 'rs_constraint' in the ANALYSIS section of the PSUADE sample file. This command does not request response surface type but it uses the 'rstype' in the loaded sample.

rssoboltsib : perform variance-based total sensitivity analysis on a response surface.

This command should be used after a PSUADE sample has been loaded into local memory. It displays the total order sensitivity indices for the response surface built from the loaded sample. Total sensitivity indices are computed by performing total-order Sobol' analysis multiple times with different bootstrapped samples. As such, the analysis results will include standard deviations of the computed Sobol' indices. This command does not request response surface type but it uses the 'rstype' in the loaded sample.

rssobolg : perform variance-based group sensitivity analysis on a response surface.

This command should be used after a PSUADE sample has been loaded into local memory. It displays group-order Sobol' indices for the response surface built from the loaded sample. You need to provide a file to specify the groupings. Constraints can

be imposed by using 'rs_constraint' in the ANALYSIS section in the sample file. This command does not request response surface type but it uses the 'rstype' in the loaded sample.

ae_ua : response surface-based mixed aleatory-epistemic uncertainty analysis. This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. This command performs outer-inner iterations to create cumulative distribution functions for mixed aleatoric-epistemic uncertainties. A plot file will be created ('matlabaeua.m') at the end.

so_ua : response surface-based second order uncertainty analysis.

This command should be used after a PSUADE sample has been loaded into local memory. This command performs second level uncertainty analysis given uncertainties about the parameter lower and upper bounds.

2.6 Commands for Optimization/calibration

rsmcmc : perform MCMC on a response surface.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It performs a Bayesian inference using MCMC on the response surface built from the sample in the local memory (make sure to validate the response surface first). You will need to answer a few questions (including options to add a discrepancy model and to generate a posterior sample) to set up the inference problem. Upon completion, the file 'matlabmcmc2.m' will have the posterior plots.

mo_opt : response surface-based multi-objective optimization.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. This function is useful if you have more than one sample output and the objective function (for optimization) is formed by weighted combination of these outputs where the weights are unknown. So given the ranges of these weights, this command generates optimal solution at the lattice points in the weight space.

2.7 Commands for Creating Visualization Plots

splot : generate scatter plots of an output against each input.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates Matlab/Scilab scatter plots (in 'matlabsp.m' or 'scilabsp.m') with each input on the X-axis and an output on the Y-axis. This command is useful for visualizing how the sample outputs vary with respect to each individual input.

splot2 : generate scatter plots of an output against 2 inputs.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a Matlab/Scilab scatter plot with 2 selected inputs on the

X- and Y-axes, and an output on the Z-axis (in 'matlabsp2.m' or 'scilabsp2.m'). This command is useful for investigating problems with fitting a 2-parameter model with response surfaces via visualizing the actual simulation output behavior. It is not very useful when the simulation model under consideration has more than 2 inputs (for 3 inputs, use 'splot3').

splot3 : generate scatter plots for 3 inputs.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a Matlab scatter plot ('matlabsp3.m') with 3 selected inputs on the X-, Y-, and Z-axes; and a selected output represented by the sizes of dots. This command is useful for investigating problems with fitting a 3-parameter model with response surfaces via visualizing the actual simulation output behavior. It is not useful when the simulation model under consideration has more than 3 inputs.

splot3m : generate scatter plots for 3 inputs (movie mode).

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a sequence of Matlab scatter plots (in 'matlabsp3m.m') with 2 selected inputs on the X-, Y-axes, and a third input on the time axes; Z-axes; and a selected output on the Z-axis. This command works only for 3-input factorial designs. It is useful for visually investigating output behaviors on the raw data. It is not useful when the simulation model under consideration has more than 3 inputs, or when the sample points are not on a lattice.

rs1 : create an one-input response surface plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates an one-input response surface plot against a sample output. You can create a Matlab ('matlabrs1.m') or Scilab file ('scilabrs1.m') by toggling the 'scilab' command.

rs1s : create an one-input response surface plot with uncertainty.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates an one-input response surface plot against a sample output (with interpolation uncertainties). You can create a Matlab ('matlabrs1s.m') or Scilab file ('scilabrs1.m') by toggling the 'scilab' command.

rs2 : create a two-input response surface plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a two-input response surface plot against a sample output. You can create a Matlab ('matlabrs2.m') or Scilab file ('scilabrs2.m') by toggling the 'scilab' command.

rs3 : create a three-input response surface plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a three-input response surface plot against a sample output. The magnitudes of the output values are represented by a color chart. The plot file is currently only available in Matlab format ('matlabrs3.m').

rs3m : create a three-input response surface plot (movie).

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a three-input response surface plot against a sample output. The third input is represented by the time axis. The plot file is currently only available in Matlab format ('matlabrs3m').

rs4 : create a four-input response surface plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a four-input response surface plot against a sample output. The fourth input is represented by the time axis and the magnitudes of the output values are represented by a color chart. The plot file is currently only available in Matlab format ('matlabrs4.m').

rssd : create a response surface error standard deviation plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. The response surface methods used here for estimating the interpolation error must be one of Gaussian process, MARS with bootstrapped aggregation, Kriging, or polynomial regression (other response surfaces do not provide error estimates). You can create a Matlab ('matlabrssd.m') or Scilab plot file ('scilabrssd.m') by toggling the 'scilab' command.

rssd_ua : perform uncertainty analysis of error standard deviation from response surface fit.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It first uses the loaded sample to create a response surface (Gaussian process, MARS with bagging, Kriging, or polynomial regression). It then estimates the errors (standard deviations) of the response surface at a parameter space lattice and generate basic statistics such as mean and standard deviation about the response surface errors. This command is useful to estimate errors and distributions of errors induced by the use of a selected response surface. The command will create a plot file in Matlab ('matlabrssdua.m') or Scilab ('scilabrssdua.m') by toggling the 'scilab' command.

rsi2 : create a two-input response surface intersection plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. In order to perform the intersection operation, the sample must have 2 or more outputs. Imposing constraints on two selected outputs will result in a smaller feasible space which can be visualized in Matlab ('matlabrsi2.m').

rsi3 : create a three-input response surface intersection plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. This command is similar to rsi2 except that it creates a three-input intersection plot. This plot capability is currently only available in Matlab ('matlabrsi3.m').

rsi3m : create a three-output response surface intersection plot (movie).

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. This command is similar to rsi2 except that it creates a three-input response surface plot against a sample output. The third input is represented by the time axis. This plot capability is currently only available in Matlab ('matlabrsi3m.m').

rawi2 : create a two-input intersection plot using raw data.

This command is similar to rsi2 except that it assumes the sample is already a factorial design and thus no response surface interpolation is required. This plot is currently available in Matlab format only ('matlabrawi2.m').

rawi3 : create a three-input intersection plot using raw data.

This command is similar to rawi2 except that it creates a three-input intersection plot. It assumes the sample is already a factorial design and thus no response surface interpolation is required. This plot is currently available in Matlab format only ('matlabrawi3.m').

rspairs : create response surfaces for all 2-input pairs.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a number of sub-plots each is a 2-input response surface plot (by freezing all other inputs at their nominal values) for all pairs of the selected inputs. The plots are currently available in Matlab format only ('matlabrspairs.m').

rsipairs : create intersection plots for all 2-input pairs.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a number of sub-plots each is a 2-input response surface plot for all pairs of the selected inputs while holding all other inputs fixed. All the outputs will be used as constraints. The plots are currently available in Matlab format only ('matlabrsipairs.m').

iplot1 : create a one-input sample input plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a sample input scatter plot for a selected input (sample number on the X-axis and input values on the Y-axis). You can create a Matlab ('matlabiplt1.m') or Scilab ('scilabiplt1.m') file by toggling the 'scilab' command.

iplot2 : create a two-dimensional sample input plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a sample input scatter plot for two selected inputs (input 1 on the X-axis and input 2 on the Y-axis). You can create a Matlab ('matlabiplt2.m') or Scilab ('scilabiplt2.m') file by toggling the 'scilab' command.

iplot3 : create a three-dimensional sample input plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a sample input scatter plot for three selected inputs (on X-, Y-, and Z-axes). You can create a Matlab ('matlabiplt3.m') or Scilab ('scilabiplt3.m') file by toggling the 'scilab' command.

iplot4m : create a three-input response surface plot (movie).

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a sample input scatter plot for four selected inputs (on X-, Y-, Z-, and t-axes; that is, the fourth input is represented by the time axis). The output file is currently only available in Matlab format ('matlabiplt4m.m').

iplot2_all : create two-dimensional sample input plots for all input pairs.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It is similar to iplot2 except that the same operation is performed on all input pairs. You can create a Matlab ('matlabiplt2_all.m') or Scilab ('scilabiplt2_all.m') file by toggling the 'scilab' command.

iplot_pdf : plot the sample PDF for individual inputs.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates histogram plots for each selected input of the loaded sample. You can create a Matlab ('matlabiplt_pdf.m') or Scilab ('scilabiplt_pdf.m') file by toggling the 'scilab' command.

iplot2_pdf : plot sample PDF for inputs (1- and 2-inputs).

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates histogram plots for each selected input of the loaded sample and heat maps for input pairs. A Matlab file called 'matlabiplt2_pdf.m' will be created at the end.

oplot2 : create a two-dimensional sample output plot.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates a sample output scatter plot for two selected outputs (output 1 on the X-axis and output 2 on the Y-axis). You can create a Matlab ('matlaboplt2.m') or Scilab ('scilaboplt2.m') file by toggling the 'scilab' command.

oplot_pdf : plot the sample PDF for individual outputs.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates histogram plots for each selected output of the loaded sample. You can create a Matlab ('matlaboplt_pdf.m') or Scilab ('scilaboplt_pdf.m') file by toggling the 'scilab' command.

oplot2_pdf : plot sample PDF for outputs (1- and 2-outputs).

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It creates histogram plots for each selected output of the loaded sample and heat maps for output pairs. A Matlab file called 'matlabiplt2_pdf.m' will be created at the end.

ihist : create an input histogram Matlab file.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It takes any single input and creates a histogram that can be displayed by Matlab ('matlabihist.m').

ihist2 : create an 2-input histogram Matlab file.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It takes any two inputs and creates 3D plot showing the frequency of each bin.

ohist : create an output histogram Matlab file.

This command is similar to 'ihist' except that it works on the sample output.

ohist2 : create an 2-output histogram Matlab file.

This command is similar to 'ihist2' except that it works on the sample outputs.

iotrace : plot inputs/outputs of each run.

This command should be used after a PSUADE sample has been loaded into PSUADE's local memory. It generates a plot with the number of lines equal to the sample size and with one line connecting all inputs and outputs of one sample point.

2.8 Commands for Setting Up PSUADE Runs

setupguide : display a short guide to help set up an application.

geninputfile : create a PSUADE input file.

genbatchfile : create a batch file.

Since many applications at LLNL use our Livermore Computing machines where jobs are submitted in batch mode, this command helps to set up the batch jobs.

gendriver : create a PSUADE driver program.

A driver program is needed in a PSUADE input file. This driver program takes the input parameter values from a file (argument 1), substitutes the data into the application input decks, runs the application, post-processes the application outputs, and writes the output data to another specified file (argument 2). This command helps to create this driver program.

genexample : create a PSUADE example.

This command creates a simple PSUADE example. After the example has been created, simply run psuade psuade.in to see the results.

chkjobs : monitor the status of job runs.

2.9 Command for Editing the Sample in PSUADE's Local Memory

validate : change the states of selected sample points to be 'evaluated'.

This command should be used after a sample has been loaded into local memory. The selected sample points will be considered 'evaluated' in local memory and can be saved using 'write'. The 'evaluated' sample points will not be evaluated when PSUADE is run on this saved sample.

invalidate : change the state of selected sample points to be 'unevaluated'.

This command should be used after a sample has been loaded into local memory. The selected sample points will be considered 'unevaluated' in local memory and can be saved using 'write'. The 'unevaluated' sample points will be evaluated (even if it has been evaluated before) when PSUADE is run on this saved sample.

srandomize : randomly change the order of the sample points.

imodify : modify an input of a selected sample point.

This command should be used after a sample is loaded into local memory.

omodify : modify an output of a selected sample point.

This command should be used after a sample is loaded into local memory.

ifilter : take out sample points that do not satisfy certain input conditions (bounds).

This command should be used after a sample is loaded into local memory. It removes all sample points having a selected input falling outside some lower and upper bounds.

ofilter : take out sample points that do not satisfy certain output conditions.

This command should be used after a sample is loaded into local memory. It removes all sample points having a selected output falling outside some lower and upper bounds.

idelete : remove an input from all sample points.

This command should be used after a sample is loaded into local memory.

odelete : remove an output from all sample points.

This command should be used after a sample is loaded into local memory.

sdelete : remove a sample point.

This command should be used after a sample is loaded into local memory. It is useful if a sample point has been determined to be an outlier and you want to remove it from further analysis.

purge : take out failed sample points.

This command should be used after a sample has been loaded into local memory. It removes all sample points that have at least one output indicated as undefined ($1.0e35$) or the status flag turned off.

ishuffle : re-arrange the orders of sample inputs.

This command should be used after a sample is loaded into local memory. You can use this command to re-arrange the sample inputs in a different ordering.

iselect_index : select and re-arrange the sample inputs based on user-given ordering information.

This command should be used after a sample has been loaded into local memory. You can use this command to align the inputs in the loaded sample with the desired ordering. This command is useful in a multi-stage UQ analysis when the Bayesian posterior sample file has a different input ordering than the one in the full system input list.

iselect_name : select and re-arrange the sample inputs based on user-given input name information.

This command should be used after a sample has been loaded into local memory. You can use this command to align the inputs in the loaded sample with the desired ordering. This command is useful in a multi-stage UQ analysis when the Bayesian posterior sample file has a different input ordering than the one in the full system input list.

sshow : display one sample point.

This command should be used after a sample has been loaded into local memory. It shows the sample input values and the corresponding output values for the selected sample point.

sinfo : display information about the current sample.

This command displays information about the loaded sample such as sample size, number of inputs, number of outputs, input names, and output names.

list1 : list one output along with one input of the loaded sample.

This command should be used after a sample has been loaded into local memory. Options are provided for sorting the inputs or the outputs before displaying.

list2 : list one sample output along with two sample inputs.

This command should be used after a sample has been loaded into local memory. Options are provided for sorting the inputs or the outputs before displaying.

listall : list one sample output along with all sample inputs.

This command should be used after a sample has been loaded into local memory.

max : display the sample point with the maximum output value.

This command should be used after a sample has been loaded into local memory.

min : display the sample point with the minimum output value.

This command should be used after a sample has been loaded into local memory.

onorm : display the 2-norm of a sample output.

This command should be used after a sample has been loaded into local memory.

osum : display the sample sum of a sample output.

This command should be used after a sample has been loaded into local memory.

irerange : re-generate a sample input using a different range.

This command should be used after a sample has been loaded into local memory. It is intended for local sample refinement whereby it is desired to create another sample with the same coordinates for most input parameters except a selected input which is shrunk to a smaller range.

ireset : map the range of an input to a distinct value.

This command should be used after a sample has been loaded into local memory. It is intended for converting continuous variables into discrete (from ranges to discrete values).

oreset : map the range of an output to a distinct value.

This command should be used after a sample has been loaded into local memory. It is intended for converting continuous output variables into discrete (from ranges to discrete values).

ifloor : map an input to the nearest smaller integer.

This command should be used after a sample has been loaded into local memory. It is intended for converting continuous variables into discrete values.

iceil : map an input to the nearest larger integer.

This command should be used after a sample has been loaded into local memory. It is intended for converting continuous variables into discrete values.

iround : map an input to the nearest integer.

This command should be used after a sample has been loaded into local memory. It is intended for converting continuous variables into discrete values.

itran : transform selected inputs to their logarithmic or exponential values.

otran : transform selected outputs to their logarithmic or exponential values.

itag : tag sample points with a selected input falling outside some bounds.

This command should be used after a sample has been loaded into local memory. Sample points with selected input values that falls outside the specified bounds will be tagged and the untagged sample indices will be displayed. Calling this command or 'otag' again will accumulate the tagged sample points. This command is useful if you want to display the list of sample points with certain input properties. The tagging can be reset (the list will be emptied) after a 'load'.

otag : tag sample points with a selected output failling outside some bounds.

This command should be used after a sample has been loaded into local memory. Sample points with selected output values that falls outside the specified bounds will be tagged and the untagged sample indices will be displayed. Calling this command or 'itag' again will accumulate the tagged sample points. This command is useful if you want to display the list of sample points with certain output properties. The tagging can be reset (the list will be emptied) after a 'load'.

rm_dup : take out duplicate sample points.

This command should be used after a sample has been loaded into local memory. It removes all duplicate sample points.

oop : perform a certain linear combination operation on the outputs.

This command should be used after a sample has been loaded into local memory. Three outputs should be selected: O1, O2 and O3. Also, two scalar numbers are to be entered: a1 and a2. This command performs the following operations on all sample points: O1 = a1 * O2 + a2 * O3.

setdriver : set the driver fields of the current PSUADE sample.

This command helps setting the driver fields in the loaded sample. The driver fileds are: simulation driver ('driver'), optimization driver ('opt_driver'), auxiliary optimization driver ('aux_opt_driver'), ensemble simulation driver ('ensemble_driver') and ensemble optimization driver ('ensemble_opt_driver'), the last of which is especially created for optimization under uncertainty.

2.10 Other Miscellaneous Commands

rename : rename a file.

This command is similar to the Unix 'mv' command. It is provided here for convenience since the load command cannot take psuadeData as a valid file name so one may want to change the name of this file before loading.

run : run a PSUADE input file.

This command ('run psuade.in') is equivalent to running the batch mode 'psuade psuade.in'.

quit : exit the current PSUADE session.

rsmax : change the maximum sample size for response surface.

Response surface construction in generally is computationally intensive. Thus, for large sample sizes, the processing time can be prohibitive. As such, PSUADE internally sets a maximum sample size for response surface construction. The default is 10000. This command is provided for users to override this restriction.

sys : initiate Linux shell command.

This command calls the external shell command with a C system call. The string attached to the end of this command (separated by a space) will be passed to the shell for execution. This general features replace the specific feature such as the previously support 'ls' command which is recognized only on Linux.

printlevel : change the diagnostics print level.

Depending on the diagnostics level, different types of information are displayed on the console. For example, setting print level to 4 will activate the cross validation in rscheck.

sqc : check sample quality.

This command should be used after a sample has been loaded into local memory. It checks the quality of the sample by computing the max-min distances between all pairs of sample points. The minimum distance between any 2 sample points should be as large as possible.

nna : perform nearest neighbor analysis for outlier detection.

This command should be used after a sample has been loaded into local memory. It finds the nearest neighbor for each sample point and then computes its distance to the nearest neighbor. The distance statistics are stored in a 'Matlab/Scilab' file for display. If the gradient of a sample point with respect to its nearest neighbor is large, it may indicate an 'outlier'.

showformat : display different PSUADE file formats.

User-provided information are often needed in design and analysis. These information are provided by users to PSUADE at various stages. This command lists several such file formats.

scilab : a toggle between Matlab or Scilab graphics outputs. The Siclab graphics capability is currently less well-developed than the Matlab graphics.

start_matlab : start Matlab without leaving the PSUADE command line interpreter.

checkformat : check a user file for correct format.

There are currently several different types of data files used in various analysis. This commands helps users check whether a data file has valid format.

2.11 Advanced Commands

script <file> : where <file> is a PSUADE command file.

This command is equivalent to running PSUADE with <file> as an argument (which allows you to execute a batch script without leaving the PSUADE interactive session).

io_expert : toggle the I/O expert mode.

This expert mode affects I/O operations internal to PSUADE via prompting users for more questions.

rs_expert : toggle the response surface expert mode.

This expert mode affects response surface methods internal to PSUADE via prompting users for more detailed information.

rs_codegen : toggle the response surface code generation mode. This mode turns on the capability to produce a stand-alone C/C++ and Python code for response surface interpolation.

ana_expert : toggle the analysis expert mode.

This expert mode affects analysis methods internal to PSUADE via prompting users for more detailed information.

sam_expert : toggle the sampling expert mode.

This expert mode affects the internal sample generation methods by prompting users for more detailed information.

opt_expert : toggle the optimization expert mode.

This expert mode affects optimization methods internal to PSUADE via prompting users for more detailed information.

genhistogram : create scenarios based on a sample

This command takes sample, which has been loaded to PSUADE's local memory, and converts it to a smaller sample set where each sample point is associated with a probability. This command is useful for optimization under uncertainty: reduce a large sample from, for example, a posterior sample from MCMC to a smaller set of scenarios.

genhistogram2 : create scenarios based on a sample and target size

This command is similar to 'genhistogram' except that users are prompted for the target number of distinct sample points.

genconfigfile : create a PSUADE configure file.

Configuration files are useful for setting more detailed options (as in expert modes) non-interactively. This command spits out an example configuration file. Features can be activated by uncommenting the lines.

use_configfile : tell PSUADE to use a configure file.

Configuration files are useful for setting more detailed options (as in expert modes) non-interactively. This command tells PSUADE to use the given configure file for setting more detailed internal options.

refine : refine a sample uniformly.

This command should be used after a sample has been loaded into local memory. The loaded sample will be *uniformly* refined (uniformly subdivide all subdomains) and the enlarged sample can be written to another PSUADE file for further processing. This command operates only on a few sampling designs (LH, MC, LPTAU, METIS).

a_refine : refine a sample adaptively.

This command should be used after a sample has been loaded into local memory. The loaded sample will be adaptively refined by subdividing the subdomains that give the largest prediction uncertainties from the response surface (as such, stochastic response surface methods such as Kriging are needed). The enlarged sample can be written to another PSUADE file for further processing. This command operates on any initial sample, but prefers the initial sample to be somewhat space-filling. Cubic splines

(MARS) with bootstrapped aggregation will be the default method used to estimate where refinement should be performed. Other methods are available when the response surface expert mode is turned on.

a_refine_metis : refine a METIS sample adaptively.

This command is similar to 'a_refine' except that it works only with an initial METIS sample with the METIS information having been stored in the 'psuadeMetisInfo' file ('a_refine' works with any sampling designs).

a_refine_cv : refine a sample adaptively.

This command should be used after a sample has been loaded into local memory. The loaded sample will be adaptively refined by subdividing the subdomains containing sample points that give the largest cross validation errors from the response surface (stochastic response surface methods is not needed). The enlarged sample can be written to another PSUADE file for further processing. This command operates on any initial sample, but prefers the initial sample to be somewhat space-filling.

interface_track : create a response surface for the interface defined by the output values (0 or 1).

This command may be useful to characterize a discontinuity in the parameter space based on the values of the given sample output. If the neighbors of a sample point with output '0' has a value of '1', this sample point is a candidate to be considered as on the interface.

moat_adjust <file> : where <file> is the file that has the Morris adjustment data.

The Morris design may require some inputs to be adjusted due to the presence of constraints that cause the sample space to be non-hyper-rectangular. To create a Morris sample that stays inside the feasible parameter space, the moatgen command should be used first. The outputs from moatgen are to be used to adjust a Morris sample. So the steps to create a Morris sample given some constraints are:

1. create a standard Morris sample (in the hyper-rectangular space)
2. use moatgen to create the adjust file (say, adjustData)
3. change the second line of the adjust file to match up the input numbers
4. start PSUADE command line mode and load the Morris sample
5. use this command (moat_adjust adjustData)
6. write the adjusted sample to a file (use write)

gmoat_adjust <file> : where <file> is to store the Morris adjust data.

This command is similar to moat_adjust except that the original sample should be a generalized Morris (GMOAT) sample.

moatgen : generate a Morris adjustment design.

This command creates a Morris adjust file. More details is found in the description of moat_adjust above. A PSUADE data file needs to be loaded first before moatgen can be used. The loaded sample is used to provide the constraints for creating the adjust file. This command will ask for the desired resolution, which determines the width of the base where gradients will be computed (default is 4).

moatgen2 : generate a Morris adjustment design for multiple constraints.

This command is similar to moatgen except that this command can handle multiple constraints. This command does not require a PSUADE data file be pre-loaded (but at least a PSUADE input file needs to be pre-loaded). Constraint files will be requested. The format for the constraint file can be found in the 'showformat' command.

moat_concat <file> : concatenate two Morris samples.

This command concatenates two Morris samples with two different sets of inputs. If the first sample has n_1 inputs and the second sample has n_2 inputs, then the final sample will have $n_1 + n_2$ inputs. The two samples should have the same number of replications. For example, if the sample in the PSUADE local memory has 10 inputs and 110 sample points and <file> has 5 inputs and 60 sample points (number of replications = 10), then the combined sample will have $10 \times (10 + 5 + 1) = 160$ sample points. The combined sample will be in local memory and needs to be stored.

3 PSUADE Function Calls

PSUADE functions can be called directly from a user program. There are a few requirements that must be met in order to do this:

- PSUADE has to be compiled in shared library mode (meaning that when 'ccmake' is launched, the 'BUILD_SHARED' flag should be on). After compilation is completed, a number of PSUADE library files ('*.so') will be present in the 'build/lib' directory.
- In the user program that intends to use PSUADE functions, make sure to include the relevant include files. During compilation, make sure to set the proper include directory (that is, the '-I' option in C/C++ compilers). It will be preferable to copy all PSUADE include files in a single directory (e.g. go to '¡PSUADE.dir¿/build', issue 'cp ../Src/*/*.h lib' where ¡PSUADE.dir¿ is the PSUADE top level directory). so that you only need to specify a single include directory.
- When linking your user program to the PSUADE libraries, do:

```
g++ -o main main.o <psuade.lib>/libpsuade.so <psuade.lib/libcobyla.so \
    -llapack -lblas -Wl,-rpath,<PSUADE.dir>/build/lib:
```

In the following we give examples on how to call a few PSUADE functions.

3.1 Calling Optimizers

PSUADE has a few numerical optimization methods that can be linked into user programs. This sub-section describes the steps to do this linking on a Linux platform for a few of these optimizers.

3.2 Calling the Bobyqa Optimizer

Bobyqa, which was developed by Michael Powell, can solve bound-constrained optimization problems. To call this optimization function, do the following in the user program:

```
....
#include "BobyqaOptimizer.h"
....
/* define the user program here */
void userEvaluator(int nInputs, double *inputs, int nOuts, double *outputs)
{
   /* subroutine body - read from inputs and write to outputs */
   /* outputs[0] will store the objective function values */
   /* outputs[1:nOuts-1] will store the constraint values */
   /*         constraint is feasible if it is <= 0)
}
...
main()
{
   int    ii, nInps=2, nOuts=3, maxEvals=1000;
   double *inputs, *lbnds, *ubnds, tol=1e-6;
   BobyqaOptimizer *opt = BobyqaOptimizer();
   opt->setObjectiveFunction(userEvaluator);
   inputs = new double[nInps];
   lbnds = new double[nInps];
   ubnds = new double[nInps];
   /* put initial guess into inputs */
   /* put lower and upper bounds into lbnds and ubnds */
   opt->optimize(nInps, inputs, lbnds, ubnds, nOuts, maxEvals, tol);
   for (ii = 0; ii < nInps; ii++)
      printf("Cobyla final   X %d = %12.4e\n",ii+1,X[ii]);
   printf("Optimal Y = %e\n", opt->getOptimalY());
   printf("Total number of function evaluations = %d\n",
          opt->getNumFuncEvals());
   delete opt;
   delete [] inputs;
   delete [] lbnds;
   delete [] ubnds;
```

```
}
```

3.3 Calling the Cobyla Optimizer

Cobyla, which was developed by Michael Powell, can solve nonlinearly-constrained optimization problems. To call this optimization function, do the following in the user program:

```
....
#include "CobylaOptimizer.h"
....
/* define the user program here */
void userEvaluator(int nInputs, double *inputs, int nOuts, double *outputs)
{
   /* subroutine body - read from inputs and write to outputs */
   /* outputs[0] will store the objective function values */
   /* outputs[1:nOuts-1] will store the constraint values */
   /*         constraint is feasible if it is <= 0)
}
...
main()
{
   int    ii, nInps=2, nOuts=3, maxEvals=1000;
   double *inputs, *lbnds, *ubnds, tol=1e-6;
   CobylaOptimizer *opt = CobylaOptimizer();
   opt->setObjectiveFunction(userEvaluator);
   inputs = new double[nInps];
   lbnds = new double[nInps];
   ubnds = new double[nInps];
   /* put initial guess into inputs */
   /* put lower and upper bounds into lbnds and ubnds */
   opt->optimize(nInps, inputs, lbnds, ubnds, nOuts, maxEvals, tol);
   for (ii = 0; ii < nInps; ii++)
      printf("Cobyla final   X %d = %12.4e\n",ii+1,X[ii]);
   printf("Optimal Y = %e\n", opt->getOptimalY());
   printf("Total number of function evaluations = %d\n",
          opt->getNumFuncEvals());
   delete opt;
   delete [] inputs;
   delete [] lbnds;
   delete [] ubnds;
}
```

3.4 Calling the Nomad Optimizer

NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct search), which was developed by Abramson, *et al*, can solve mixed integer and continuous variable optimization problems. To call this optimization function, you will first need to download the NOMAD library, compile it into PSUADE, and do the following in the user program:

```
....
#include "NomadOptimizer.h"
....
/* define the user program here */
void userEvaluator(int nInputs, double *inputs, int nOuts, double *outputs)
{
   /* subroutine body - read from inputs and write to outputs */
   /* outputs[0] will store the objective function values */
   /* outputs[1:nOuts-1] will store the constraint values */
   /*         constraint is feasible if it is <= 0)
}
...
main()
{
   int    ii, nInps=2, nOuts=3, maxEvals=1000;
   double *inputs, *lbnds, *ubnds, tol=1e-6;
   NomadOptimizer *opt = NomadOptimizer();
   opt->setObjectiveFunction(userEvaluator);
   inputs = new double[nInps];
   lbnds = new double[nInps];
   ubnds = new double[nInps];
   /* put initial guess into inputs */
   /* put lower and upper bounds into lbnds and ubnds */
   opt->optimize(nInps, inputs, lbnds, ubnds, nOuts, maxEvals, tol);
   for (ii = 0; ii < nInps; ii++)
      printf("Cobyla final   X %d = %12.4e\n",ii+1,X[ii]);
   printf("Optimal Y = %e\n", opt->getOptimalY());
   printf("Total number of function evaluations = %d\n",
          opt->getNumFuncEvals());
   delete opt;
   delete [] inputs;
   delete [] lbnds;
   delete [] ubnds;
}
```