# Forward UQ propagation using a “black-box” approach

UQTk Example

# Description of input parameters

Command-line usage:

```
./prob3.py --nom nomvals -s stdfac -d dim -l lev -o ord -q sp --npdf npdf
           --npces npces
```

Parameters:

- *nomvals*: List of nominal parameter values, separated by comma if more than one value, and no spaces. Default is one value, 20.75
- *stdfac*: Ratio of standard deviation/nominal parameter values. Default value: 0.1
- *dim*: number of uncertain input parameters. Currently this example can only handle $dim = 1$
- *lev*: No. of quadrature points per dimension (for full quadrature) or sparsity level (for sparse quadrature). Default value: 21.
- *ord*: PCE order. Default value: 20
- *sp*: Quadrature type “full” or “sparse”. Default value: “full”
- *npdf*: No. of grid points for Kernel Density Estimate evaluations of output model PDF’s. Default value 100
- *npces*: No. of PCE evaluations to estimate output densitie. Default value $10^5$

  **Note: This example assumes Hermite-Gauss chaos for the model input parameters.**

# UQTk utilities used by this example

The path to UQTk's "src_cpp" directory needs to be set in the environment variable "UQTK_SRC". For example

```
setenv UQTK_SRC $HOME/uqtk_dist/working
```

or

```
export UQTK_SRC=$HOME/uqtk_dist/working
```

The executables for the utilities below reside under `$UQTK_SRC/src_cpp/bin` :

- *generate_quad*: Generate quadrature points for full/sparse quadrature and several types of rules.
- *pce_rv*: Generate samples from a random variable defined by a Polynomial Chaos expansion (PCE)
- *pce_eval*: Evaluates PCE for germ samples saved in input file "xdata.dat".
- *pce_resp*: Constructs PCE by Galerkin projection

# Sequence of computations:

1. *prob3.py*
   saves the input parameters' nominal values and standard deviations in a diagonal matrix format in file "pcfile". First it saves the matrix of nominal values, then the matrix of standard deviations. This information is sufficient to define a PCE for a normal random variable in terms of a standard normal germ. For a one parameter problem, this file has two lines.
2. *generate_quad*:
   Generate quadrature points for full/sparse quadrature and several types of rules. The usage with default script arguments

   ```
   generate_quad -d1 -g'HG' -xfull -p21 > logQuad.dat
   ```

   This generates Hermite-Gauss quadrature points for a 21-point rule in one dimension. Quadrature points locations are saved in "qdpts.dat" and weights in "wghts.dat" and indices of points in the 1D space in "indices.dat". At the end of "generate_quad" execution file "qdpts.dat" is copied over "xdata.dat"

3. *pce_eval*:
   Evaluates PCE of input parameters at quadrature points, saved previously in "xdata.dat". The evaluation is dimension by dimension, and for each dimension the corresponding column from "pcfile" is saved in "pccf.dat". See command-line arguments below.

```
pce_eval -x'PC' -p1 -q1 -f'pccf.dat' -sHG >> logEvalInPC.dat
```

   At the end of this computation, file "input.dat" contains a matrix of PCE evaluations. The number of lines is equal to the number of quadrature points and the number of columns to the dimensionality of input parameter space.

4. *Model evaluations*:

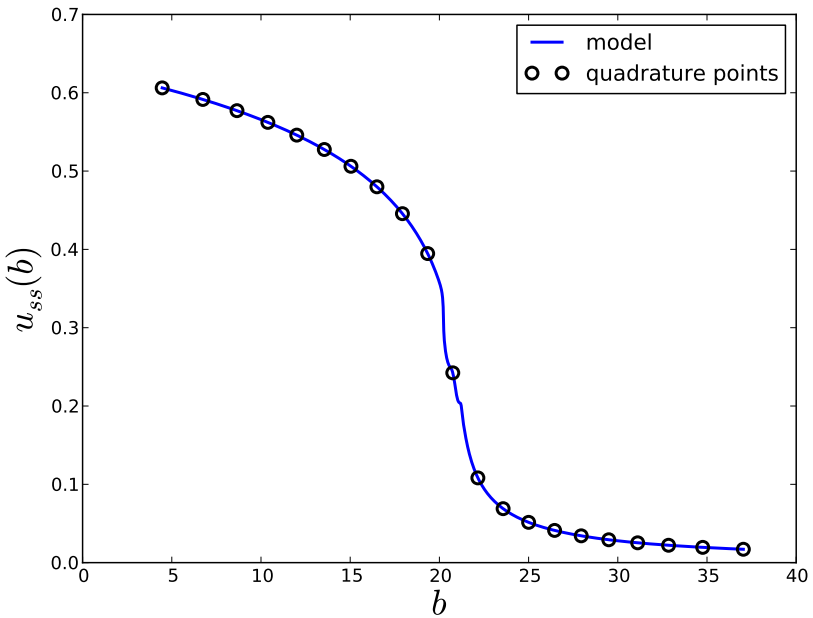
```
funcBB("input.dat","output.dat",xmltpl="surf_rxn.in.xml.tp3",
       xmlin="surf_rxn.in.xml")
```

   The python function *"funcBB"* is defined in file "prob3_utils.py". This evaluates the forward model at sets of input parameters in file "input.dat" and saves the model output in "output.dat". For each model evaluation, specific parameters are inserted in the xml file "surf_rxn.in.xml" which is a copy of the template in "surf_rxn.in.xml.tp3". At the end "output.dat" is copied over "ydata.dat"

5. *pce_resp*:

```
pce_resp -xHG -o20 -d1 -e > logPCEresp.dat
```

Computes a Hermite-Gauss PCE of the model output via Galerkin projection. The model evaluations are taken from “ydata.dat”, and the quadrature point locations from “xdata.dat”. PCE coefficients are saved in “PCcoeff_quad.dat”, the multi-index list in “mindex.dat” and these files are pasted together in “mipc.dat”




(average *u* as a function of parameter b values. Location of quadrature points is shown with circles.)

6. *pce_rv*:

   ```
   pce_rv -w'PCvar' -xHG -d1 -n100 -p1 -q0 > logPCrv.dat
   ```

   Draw a 100 samples from the germ of the HG PCE. Samples are saved in file "rvar.dat" and also copied to file "xdata.dat"

7. *pce_eval*:

   ```
   pce_eval -x'PC' -p1 -q1 -f'pccf.dat' -sHG >> logEvalInPCrnd.dat
   ```

   See item **??** for details. Results are saved "input_val.dat".

8. Evaluate both the forward model (through the black-box script "funcBB", see item **??**) and its PCE surrogate (see item **??**) and save results to files "output_val.dat" and "output_val_pc.dat". Compute $L_2$ error between the two sets of values using function "compute_err" defined in "utils.py"

9. Sample output PCE and plot the PDF of these samples computed using either a Kernel Density Estimate approach with several kernel bandwidths or by binning: