An Example application of the Quadrature-Based Uncertainty Quantification Package for MFiX

Xiaofei Hu
xhu@iastate.edu
Department of Chemical and Biological Engineering
Iowa State University
Ames, IA, USA

Alberto Passalacqua
albertop@iastate.edu
Department of Mechanical Engineering
Iowa State University
Ames, IA, USA

Rodney O. Fox
rofox@iastate.edu
Department of Chemical and Biological Engineering
Iowa State University
Ames, IA, USA

December 3rd, 2013

1 Introduction

This document provides an example of using the non-intrusive quadrature-based uncertainty quantification (QBUQ) package implemented in MFiX . The package is made of two parts: pre-processing and post-processing modules. In the pre-processing module, the space of the uncertain input parameters is sampled first, quadrature weights and nodes are obtained, and the corresponding MFiX input file mfix.dat for each sample is generated. A set of MFiX simulations can be performed to obtain quadrature abscissas. In the post-processing module, moments of the system response are estimated. Low order statistics, including mean, variance, skewness, and kurtosis are calculated, and the probability distribution function (PDF) of the system response at specific locations specified by the user can be reconstructed. In the following parts, the MFiX tutorial case fluidBed1, with uncertain particle, size is used as an example case to illustrate the QBUQ procedure.

2 An example to run QBUQ package

The QBUQ modules are written in python 3. The example shown here is using python 3 (version 3.2.3) [3] with the following packages:

- NumPy – Version 1.7.1 [1]
- SciPy – Version 0.12.0 [4]
- SymPy – Version 0.7.2 [5]

Simulations are run using MFiX -2013-1, and bash shell is used for shell scripts.

Notes:

- We assume python3 is the command running python 3, since many systems still default to python 2.x. If your operating systems uses python 3 by default, replace each instance of the "python3" command with "python".
- The QBUQ package was tested on Linux systems based on openSUSE 12.3 and openSUSE 13.1 [2].

The MFiX tutorial case fluidBed1 is used as an example case in this section. It is a 2-D bubbling fluidized bed reactor with a central jet. The uncertain parameter in this example is the particle diameter, uniformly distributed

from 300 μm to 500 μm. In order to use the QBUQ package, create a new run directory, and put the following directory and files in your run directory: directory "qbuq_package", python 3 files "pre_qbuq.py" and "post_qbuq.py", and bash script files "run_serial" and "extract_data". You also need to provide a basic mfix.dat file in the run directory to use for generating mfix.dat file for each sample.

2.1 Pre-processing module

The pre-processing module samples the space of uncertain parameters (particle diameter in the example case), generates quadrature weights and nodes, and creates the corresponding MFiX input files. The python 3 file "pre_qbuq.py" is the main script, and the corresponding functions are in the sub-package "pre_processing" under the package "qbuq_package". To use the module, follow the following procedure.

1. Go to your *run directory* in a terminal, then type the command below to run the module:

python3 pre_qbuq.py

2. Type **1** to select "1 – Univariate problem".

The program asks you to choose the number of uncertain parameters. Only univariate and bivariate problems are available at the moment. For our case, the only uncertain parameter is the particle diameter.

3. Type **y** for known distribution.

The program asks if the distribution of the variable is known. Since the particle diameter in our example is uniformly distributed, we choose "yes".

4. Type **1** to choose "Uniform distribution".

5. Type **10** to generate ten samples in the example.

6. Enter **0.03** as the minimum value and **0.05** as the maximum value.

This is the range of your uniformly distributed uncertain parameter. Remember, in the basic MFiX input file mfix.dat, the units system we use is "*cgs*". Therefore, our range of particle diameter is 0.03 cm to 0.05 cm.

So far the first step (sample the space of uncertain parameter) is completed. The program says a text file containing quadrature weights and nodes is generated successfully. The name of this file is "quadrature_weights_nodes.txt",

in which the first row shows quadrature weights, and the second row shows quadrature nodes. Then the program starts to generate the MFiX input files for each sample.

7. Type **Test** as the head of the run_name.

The head of the run_name is case sensitive. The run_name for each mfix.dat file will be this head followed by a number. In our case, with increasing particle size, the run_name will be Test0 to Test9.

8. Enter **1** as the number of keywords in mfix.dat file use the value of the uncertain parameter.

Please count carefully to include all the keywords related to the uncertain parameter. In our case, only one keyword "*D_p0*" is related to particle diameter.

9. Type **D_p0** as the name of the keyword.

Note: The keywords typed here should be exactly the same as those in the basic mfix.dat file because these keywords will be searched and their values will be replaced by the quadrature nodes. If the keywords do not match, the values will not be replaced.

The program then searches the keywords "*run_name*" and "*D_p0*" in the basic mfix.dat file, and replaces their values with the new run_name and quadrature node value, respectively. The MFiX input file mfix.dat generated for each sample is stored in separate directories named by their run_name. A text file named "*list_of_cases.txt*" is also generated, which contains run_name of each sample in the order of increasing quadrature nodes so that in post-processing part quadrature weights and nodes match.

The pre-processing part of QBUQ procedure is now completed. The program says the MFiX input files are generated successfully. In the run directory, you should have two more text files, "quadrature_weights_nodes.txt" and "list_of_cases.txt", and ten more folders named from Test0 to Test9, each one containing a MFiX input file mfix.dat. The only difference among these input files is the value of *D_p0*. In the next section, we will see how to use the shell script "run_serial" to run the simulations in serial.

2.2 Running simulations in serial

A bash script "run_serial" is written to run generated MFiX simulations in serial. In this example, when we compile MFiX , the serial mode of execution

for each sample is selected. To run MFiX in DMP or SMP mode, please change the commands in this file accordingly. If you want to submit your jobs on a cluster through a batch queue system, please consult with your system administrator.

To use the script "run_serial", in the run directory, type:

bash run_serial MFIX_model_directory

Here the "MFIX_model_directory" is the path of your MFiX model directory, such as "/home/mfix/model".

The program prompts the initial compilation of MFIX. The options used in this initial compilation will be used for all the simulations generated in 2.1. The options used in this example are following:

- Mode of execution: [1] Serial
- Level of optimization: [3] Level 3 (most aggressive)
- Option to re-compile source files in run directory:

  [1] Do not force re-compilation
- Compiler selection: [1] GNU (gfortran) version 4.3 and above

After the initial compilation, the same options are used repeatedly for all the simulations. From Test0 to Test9, simulations are compiled and run in the order listed in text file "list_of_cases.txt". The screen outputs of compilation and run for each simulation are redirected to log files "compile.log" and "run.log", respectively. Once all the simulations are completed, time-averaged quantities can be extracted, and the post-processing module of QBUQ can be used.

2.3 Post-processing module

The post-processing module first extracts time-averaged quantities with a bash shell script "extract_data" by calling MFiX post-processing program "post_mfix". Time-averaged results can then be used with the QBUQ post-processing module "post_qbuq.py". The module can perform two tasks: estimate moments and low order statistics (mean, variance, skewness, and kurtosis), and reconstruct the PDF of system response. The functions used in this module are in the sub-package "post_processing" under the package "qbuq_package".

2.3.1 Extracting time-averaged data

To extract time-averaged quantities, a basic text file needs to be provided in the run directory to use as the input file for MFiX post-processing program "post_mfix". Then the bash shell script "extract_data" will replace the run_name automatically and extract the data for each simulation. In this example, two basic text files are provided in the run directory: "post_all.txt" extracts time-averaged quantities on the whole computational domain which will be used for estimation of moments and low order statistics, and "post_point.txt" extracts data at one specific location which will be used for reconstruction of the PDF. The following sections show what settings are used in these two post_mfix input files for MFiX post-processing program.

Note: When chosing the file name where the extracted data are stored, a .txt file extension must be added. Only .txt files can be read by the post_qbuq module.

post_all.txt

- RUN_NAME to post_procss > Test (*Note:* This name will be replaced automatically with the run_name of each simulation.)
- Enter menu selection > 1 - Examine/print data
- Write output using user-supplied precision? (T/F) > f
- Time:(0.000, 0.000) > 0, 2
- Time average? (N) > y
- Variable: (EP_g) > EP_g
- I range: (1, 1) > 1, 9
- Average or sum over I? (N) > n
- J range: (1, 1) > 1, 102
- Average or sum over J? (N) > n
- K range: (1, 1) > 1, 1
- File: (*) > EP_g_all.txt (*Note:* Must use .txt file here. The post_qbuq module only reads .txt files.)
- Time:(0.000, 0.000) > -1

- Enter menu selection > 0 - Exit POST_MFIX

Only the gas volume fraction "*EP_g*" is extracted for the whole computational domain as an example. Other time-averaged quantities can be extracted by changing or adding conditions in the file accordingly.

post_point.txt

- RUN_NAME to post_procss > Test
- Enter menu selection > 1 - Examine/print data
- Write output using user-supplied precision? (T/F) > f
- Time:(0.000, 0.000) > 0, 2
- Time average? (N) > y
- Variable: (EP_g) > EP_g
- I range: (1, 1) > 7, 7
- J range: (1, 1) > 51, 51
- K range: (1, 1) > 1, 1
- File: (*) > EP_g.txt (*Note:* Must use .txt file here. The post_qbuq module only reads .txt files.)
- Time:(0.000, 0.000) > 0, 2
- Time average? (N) > y
- Variable: (EP_g) > P_g
- I range: (1, 1) > 7, 7
- J range: (1, 1) > 51, 51
- K range: (1, 1) > 1, 1
- File: (*) > P_g.txt
- Time:(0.000, 0.000) > 0, 2
- Time average? (N) > y
- Variable: (EP_g) > V_s
- I range: (1, 1) > 7, 7

- J range: (1, 1) > 51, 51
- K range: (1, 1) > 1, 1
- File: (*) > V_s.txt
- Time:(0.000, 0.000) > -1
- Enter menu selection > 0 - Exit POST_MFIX

Gas volume fraction "*EP_g*", gas pressure "*P_g*" and vertical solid velocity "*V_s*" are extracted at a specific location. Conditions in the file can be changed or added accordingly.

To use the bash shell script "extract_data" to extract time-averaged data for all the simulations in the run directory, type:

bash extract_data post_file post_mfix_directory

Here the "post_file" is the file used as input for the MFiX post-processing program "post_mfix". In this example, it can be either "post_all.txt" or "post_point.txt". The "post_mfix_directory" is the path of your MFiX post_mfix directory, such as "/home/mfix/post_mfix". We run the script twice using both "post_all.txt" and "post_point.txt" to extract gas volume fraction "*EP_g*" on the whole computational domain, and gas volume fraction "*EP_g*", gas pressure "*P_g*" and vertical solid velocity "*V_s*" at a specific location for all ten simulations. Once the process is completed, these data can be used with the post_qbuq module.

2.3.2 Estimation of moments and low order statistics

Two tasks can be performed by the QBUQ post-processing module, estimation of moments and low order statistics, and reconstruction of the PDF of the system response. We first show how to estimate the moments and low order statistics of the PDF of the system response.

1. In the run directory, type the following command to run the module:

python3 post_qbuq.py

2. Type **1** to choose "1 – Calculate low order statistics".

3. Type **EP_g_all.txt** to continue.

Here the program asks for the file name of the quantity of interest. In this example, it is the file name used in file "post_all.txt", "EP_g_all.txt".

4. Type **10** for ten samples in this example.

5. Type **4** as the highest order of moments.

In order to estimate low order statistics including mean, variance, skewness, and kurtosis, at least the fourth-order moment is needed.

The program uses the time-averaged data of each simulation extracted in 2.3.1 as quadrature abscissas and quadrature weights generated in 2.1 to estimate moments of the gas volume fraction "*EP_g*" up to the fourth order, on the whole computational domain. Mean, variance, skewness, and kurtosis are calculated as well. A runtime warning may be displayed at this point because, when calculating skewness and kurtosis, the standard deviation as the denominator may become 0 at some location. These warnings do not affect the results at other locations.

Once the program has run, five more text files are generated in the run directory: "moments.txt", "mean.txt", "variance.txt", "skewness.txt", and "kurtosis.txt".

Note: Make sure these files are stored and renamed correctly before using the post_qbuq module again. These files will be overwritten when the module is used for other quantities of interest.

2.3.3 Reconstruction of the PDF

We now illustrate an example of reconstructing the PDF of the system response. The PDF of three quantities, gas volume fraction "*EP_g*", gas pressure "*P_g*" and vertical solid velocity "*V_s*", at a specific location are reconstructed using extended quadrature method of moments (EQMOM)[6, 8].

2.3.3.1 Gas volume fraction "*EP_g*"

1. Type the following command in the run directory to run the QBUQ post-processing module:

python3 post_qbuq.py

2. Type **2** to select "2 – Reconstruction the probability distribution function".

3. Enter **1** to choose β-EQMOM for *EP_g*.

4. Type **4** as number of nodes.

Maximum 5 nodes can be used in β-EQMOM and γ-EQMOM.

5. Type **0**, **1e − 12**, **1e − 8**, and **1e − 4** as $rmin(0)$, $rmin(1)$, $rmin(2)$, and $rmin(3)$, respectively.

These are the four ratios of minimum to maximum weight (rmin), which are parameters used in adaptive Wheeler algorithm [7].

6. Enter **1e − 10** as eabs.

This is the minimum distance between distinct nodes, which is also a parameter used in the adaptive Wheeler algorithm [7].

7. Type **10** for a total of ten simulations in this example.

8. Type **EP_g.txt** as the file name of quantity of interest.

9. Enter **1** to use minimum to maximum value of the data as the interval.

In β-EQMOM, a bounded interval is needed. In the post_qbuq module, the bounded interval can either use the minimum to maximum value of the data or be set up manually.

The program then uses β-EQMOM to reconstruct the PDF of the gas volume fraction. Once this is completed, σ, weights, nodes on $[0, 1]$, and nodes on bounded interval are shown on the screen. Three more text files are generated in the run directory. The "betaEQMOM_sigma.txt" file keeps the value of σ. In the "betaEQMOM_weights_nodes.txt" file, the first row shows the weights of each EQMOM node, the second row gives the value of each EQMOM node on $[0, 1]$, and the third row shows the value of each EQMOM node on the bounded interval. The "data_set_for_betaEQMOM.txt" file contains ten values used to reconstruct the gas volume fraction, each of which is the value of gas volume fraction of that simulation at the specific location.

For your reference, Table 1 gives the results shown on the screen for the example case.

2.3.3.2 Gas pressure "*P_g*"

The procedure of using post_qbuq module to reconstruct the PDF of gas pressure P_g at the location is similar to the one used in 2.3.3.1 except that different options are selected. Conditions used in the module to reconstruct the PDF of gas pressure are as follows.

sigma	0.00523497997459	
weights	0.46426887803	0.284369847782
	0.217899463915	0.033461810273
nodes	0.00956478087672	0.0894838372351
on $[0, 1]$	0.629186940922	0.999750406862
nodes	0.415734483248	0.425965720843
on $[a, b]$	0.495058512177	0.542498047086

Table 1: Reconstruction results for the gas volume fraction using β-EQMOM.

- Type **python3 post_qbuq.py** to run the module.
- Type **2** to select reconstruction of the PDF.
- Type **2** to use γ-EQMOM to reconstruct the PDF of gas pressure.
- Enter **5** as the number of EQMOM nodes.
- For the parameters rmin in adaptive Wheeler algorithm, the first four are the same with 2.3.3.1. Type $\mathbf{1e-4}$ for the fifth one $rmin[4]$.
- Use $\mathbf{1e-10}$ again as eabs in adaptive Wheeler algorithm.
- Enter **10** for 10 samples.
- The file name of gas pressure in this example is **P_g.txt**.
- Select **2** to set the lower bound for γ-EQMOM manually.
- Type **200** as the lower bound.

Once the reconstruction is completed, three more files are generated in the run directory: "gammaEQMOM_sigma.txt", "gammaEQMOM_weights_nodes.txt", and "data_set_for_gammaEQMOM.txt". Table 2 gives results shown on screen for your reference.

sigma	19.606992775		
weights	0.177257589525	0.27693370017	0.181902294417
	0.150325247342	0.213581168547	
nodes	283.874714748	1536.18411219	3175.10056231
on $[0, \infty)$	4271.74179277	6774.60334977	
nodes	483.874714748	1736.18411219	3375.10056231
on $[a, \infty)$	4471.74179277	6974.60334977	

Table 2: Reconstruction results of gas pressure using γ-EQMOM.

2.3.3.3 Vertical solid velocity "*V_s*"

The procedure of using post_qbuq module to reconstruct the PDF of vertical solid velocity *V_s* at the location is slightly different from the procedure used in 2.3.3.1 and 2.3.3.2, since 2-node Gaussian EQMOM is used[6]. The procedure is as follows.

- Type **python3 post_qbuq.py** to run the module.
- Type **2** to select reconstruction of the PDF.
- Type **3** to use 2-node Gaussian EQMOM to reconstruct the PDF of vertical solid velocity.
- Enter **10** for 10 samples.
- The file name of vertical solid velocity in this example is **V_s.txt**.

Three more files are generated in the run directory once the reconstruction is completed: "GaussianEQMOM_sigma.txt",
"GaussianEQMOM_weights_nodes.txt", and
"data_set_for_GaussianEQMOM.txt". Table 3 gives results on author's screen for your reference.

sigma	1.93623130711	
weights	0.694653795117	0.305346204883
nodes	-5.1032997211	3.10647288573

Table 3: Reconstruction results of vertical solid velocity using 2-node Gaussian EQMOM.

References

[1] NumPy. http://www.numpy.org/.

[2] openSUSE. http://www.opensuse.org.

[3] Python. http://www.python.org/.

[4] SciPy. http://www.scipy.org/.

[5] SymPy. http://sympy.org/en/index.html.

[6] C. Chalons, R.O. Fox, and M. Massot. A multi-Gaussian quadrature method of moments for gas-particle flows in a LES framework. In *Proceedings of the Summer Program*, pages 347 – 358. Center for Turbulence Research, Stanford University, 2010.

[7] C. Yuan and R. O. Fox. Conditional quadrature method of moments for kinetic equations. *Journal of Computational Physics*, 230(22):8216–8246, 2011.

[8] C. Yuan, F. Laurent, and R. O. Fox. An extended quadrature method of moments for population balance equations. *Journal of Aerosol Science*, 51:1–23, 2012.