

# UQpy - Uncertainty Quantification with Python

Dimitris G. Giovanis, M. D. Shields

*Johns Hopkins University, USA*

---

---

## 1. Installing UQpy

Prerequisites: You have at least one Python interpreter 3.6+ properly installed on your computer. In order to get the latest experimental version of UQpy the code can be installed from Github directly as follows:

```
$git clone https://github.com/SURGroup/UQpy.git
```

```
$cd UQpy/
```

```
$pip install -r requirements.txt.
```

```
$python setup.py install.
```

The last command might need **sudo** prefix, depending on your python setup.

## 2. Overview

UQpy (Uncertainty Quantification (UQ) using python) is a software toolbox containing a collection of modules written in Python that provide standardized solutions for many UQ problems that occur in physical model. Connection between UQpy and the user-defined computational model is made with text-based and bash shell script(s) provided by the user. Execution of UQpy results in realizations of the parametric space of interest using advanced techniques, as well as evaluations the corresponding model responses. UQpy is entirely code-agnostic and gives users a fully functional tool for performing UQ with nearly any computational analysis code. UQpy performs submission, execution, monitoring and post-process analysis, specifically tailored to the

21 analysis tool and the available platform and thus, it is amenable to perform-  
22 ing adaptive UQ methods. `UQpy` is written in the Python 3 programming  
23 language.

## 24 2.1. Compiled version of `UQpy`

25 We need to address the Windows version

## 26 2.2. Interpreted version of `UQpy`

27 The interpreted version of `UQpy` requires a Python shell supporting Python  
28 3.6+ as well as several common Python libraries as well. After downloading  
29 and installing `UQpy`, the following `UQpy`-specific files are required and must  
30 be co-located in the subdirectory `lib/UQpy`, which is in the same directory  
31 as `UQpy_cmd.py`:

- 32 · `UQpyModules.py` - Contains various functions.
- 33 · `SampleMethods.py` - Contains the available sampling methods used for  
34 exploring the parameter space.
- 35 · `ReadInputFile.py` - Reads the necessary UQ Parameter data file in case  
36 of running `UQpy` via command line, and converts it to python variables.
- 37 · `PDFs.py` - Contains the percent point functions of all the supported  
38 distributions; any new distribution can be added here.

## 39 3. Using `UQpy`- Required files

40 `UQpy` may be run using either an Integrated Development Environment (IDE)  
41 used in computer programming, specifically for the Python language or via  
42 the command line. The interpreted version of `UQpy`, has been tested to run  
43 in IDE PyCharm 2017.3.3.

44 In order to use `UQpy` for evaluating the response of any computational  
45 model for a number of parameter realizations, `UQpy` requires three **exe-**  
46 **cutable**<sup>1</sup> bash shell scripts:

---

<sup>1</sup>`$chmod +x name1*.sh`

- `name1*.sh` for linking the analysis software to `UQpy`
- `name2*.sh` for converting the the file containing the parameter values (text-based file) into appropriate input file for the analysis code
- `name3*.sh` converting the result of the software analysis into an appropriate (text-based) file to be read from `UQpy`. This is necessary in case of running adaptive UQ methods and/or post-processing of the results.

The names of these files are user defined. Additional to these files, if the user wants to generate the realizations of the random parameters according to one of the available sampling methods provided in `UQpy`, it is necessary to provide an text-based file under the name (`UQpy_params.txt`), which will enclose all the probabilistic information required for running the selected sampling method. T

The aforementioned files are directly specified by the user and may be in any directory.

## 4. UQpy Usage

`UQpy` is user friendly since it only requires the user to have basic knowledge in writing bash shell scripts.

### 4.1. Using the UQpy Command Line Mode

`UQpy` can be executed directly through the command line. It is provided as an option to the user who doesn't have sufficient familiarity and experience with python. Command line execution is advantageous when analyses need to be performed on a high-performance computing systems without direct graphics capability. In order to execute the interpreted version `UQpy` from the command line the user needs to change to the `UQpy` directory and then type in terminal :

```
$python UQpy_cmd.py --dir pathToModel --model name1*.sh --input name2*.sh --output name3*.sh
```

Where

- `UQpy_cmd.py` is the python script that actually runs `UQpy` via command line and needs to be located in the directory `UQpy`.

- 76 • `--dir` is the absolute path to the folder which contains the necessary  
77 files `{name1*.sh , name2*.sh , name3*.sh and UQpy_params.txt}`.
- 78 • `--model` points to the `name1*.sh` bash script
- 79 • `--input` points to the `name2*.sh` bash script
- 80 • `--output` points to the `name3*.sh` bash script

81 In order for UQpy to run from command line the file `UQpy_params.txt` is nec-  
82 essary to be located inside `--dir` otherwise, the execution will return an er-  
83 ror. However the user may skip the entries `{--input, --output, --model}`  
84 if UQpy is utilized only for generating realizations of the random parameter  
85 and not for model evaluations. Another optional entry for the user is `--CPUs`  
86 which sets the number of processors used for the evaluation of the model, in  
87 case of parallel processing. The user can see all the available options (Fig.  
88 1) by typing in terminal

```
89 $python UQpy_cmd.py --help
```

90 which results in:

```
python UQpy.py --{options}

optional arguments:
  -h, --help            show this help message and exit
  --dir MODEL_DIRECTORY Specify the location of the model's directory.
  --input INPUT_SHELL_SCRIPT
                        Specify the name of the shell script *.sh used to
                        transform the output of UQpy (UQpyOut_*.txt file) into
                        the appropriate model input file
  --output OUTPUT_SHELL_SCRIPT
                        Specify the name of the shell script *.sh used to
                        transform the output of the model into the appropriate
                        UQpy input file (UQpyInp_*.txt)
  --model SOLVER        Specify the name of the shell script used for running
                        the model
  --CPUs CPUS           Number of local cores to be used for the analysis
```

Figure 1:

## 91 4.2. Using the UQpy IDE Mode

92 After installation, UQpy is build in the local Python's standard library and  
93 thus, it runs from any Integrated Development Environment (PyCharm,  
94 Atom, Eclipse, e.t.c) which provides code analysis and debugging. In or-  
95 der to use UQpy libraries in a project the user needs to import the specific  
96 module to its workspace. This can be done by writing in a python script

```
97         from UQpy import *
```

98 which will load all modules of UQpy. If a specific class from the sample  
99 methods (e.g Monte Carlo simulation) is required then the user can selectively  
100 load it to the project by typing

```
101         from UQpy.SampleMethods import MCS
```

102 This functionality of UQpy enables the independent usage of its modules,  
103 which makes UQpy a powerful tool for UQ analysis and communication be-  
104 tween python and various computational codes of different nature. In order  
105 to generate 100 realizations of two random parameters using MCS the user  
106 needs to type:

```
107         from UQpy.SampleMethods import MCS
108         x = MCS(dimension=2, pdf_type=['Uniform', 'Uniform'])
109         pdf_params=[[0, 1], [0, 1] ], nsamples=100)
```

110 This will create the object `x` with its properties:

- 111 1. `pdf_type`: type of distribution for each parameter
- 112 2. `pdf_params`: distribution parameters
- 113 3. `nsamples`: number of samples to be generated
- 114 4. `dimension`: number of random parameters
- 115 5. `samples`: generated samples in the parameter space
- 116 6. `samplesU01`: generated samples in the Uniform space,  $U[0, 1]^{\text{dimension}}$

## 117 5. UQpy workflow

## 118 6. Templates for the required Files

119 The interaction between UQpy and any external solver is made with text-  
120 based files which are simple to process and easy to work with in python.

### 121 6.1. Probabilistic Parameter File

122 The file that keeps the probabilistic properties of the parameters should  
123 always be under the name:

124 `UQpy-params.txt`

125 Creating `UQpy-params.txt` is simple and straightforward; Each property that  
126 is required for the selected sampling method, is defined in a line that starts  
127 with a hash-tag (`#`), followed by a key-word and/or key-phrase (case sensi-  
128 tive) describing the property<sup>2</sup>. The file ends with the key-word `#end`. Under  
129 that line, the specific attributes of the property are defined, according to the  
130 UQpy available options. Thus, different sampling methods require different  
131 parameter file .

#### 132 6.1.1. Required properties for various sampling methods

133 The properties that need to be specified by the user inside the parameter  
134 file in order to run different sampling methods, for exploring the parameter  
135 space. A summary of these properties is given next:

---

<sup>2</sup>The order that the properties are declared in `UQpy-params.txt` is not important .

136

| Monte Carlo simulation   |           |               |
|--------------------------|-----------|---------------|
| Property                 | Mandatory | Optional      |
| #method                  | ★         |               |
| #number of samples       | ★         |               |
| #number of parameters    | ★         |               |
| #distribution type       | ★         |               |
| #distribution parameters | ★         |               |
| #names of parameters     |           | ★             |
| #SROM                    |           | True or False |

137

| Latin hypercube simulation |           |               |
|----------------------------|-----------|---------------|
| Property                   | Mandatory | Optional      |
| #method                    | ★         |               |
| #number of samples         | ★         |               |
| #number of parameters      | ★         |               |
| #distribution type         | ★         |               |
| #distribution parameters   | ★         |               |
| #names of parameters       |           | ★             |
| #criterion                 |           | ★             |
| #distance                  |           | ★             |
| #metric                    |           | ★             |
| #SROM                      |           | True or False |

138

| Stratified sampling      |           |               |
|--------------------------|-----------|---------------|
| Property                 | Mandatory | Optional      |
| #method                  | ★         |               |
| #distribution type       | ★         |               |
| #number of parameters    |           | ★             |
| #distribution parameters | ★         |               |
| #design                  | ★         |               |
| #names of parameters     |           | ★             |
| #SROM                    |           | True or False |

139

| Partially Stratified sampling |           |               |
|-------------------------------|-----------|---------------|
| Property                      | Mandatory | Optional      |
| #method                       | ★         |               |
| #distribution type            | ★         |               |
| #distribution parameters      | ★         |               |
| #number of parameters         |           | ★             |
| #design                       | ★         |               |
| #strata                       | ★         |               |
| #names of parameters          |           | ★             |
| #SROM                         |           | True or False |

140

| Markov chain Monte Carlo simulation |           |               |
|-------------------------------------|-----------|---------------|
| Property                            | Mandatory | Optional      |
| #method                             | ★         |               |
| #algorithm                          |           | ★             |
| #number of samples                  | ★         |               |
| #number of parameters               |           | ★             |
| #proposal distribution type         |           | ★             |
| #proposal distribution width        |           | ★             |
| #target distribution                | ★         |               |
| #target distribution parameters     | ★         |               |
| #names of parameters                |           | ★             |
| #jump                               |           | ★             |
| #initial seed                       |           | ★             |
| #SROM                               |           | True or False |

141

| Stochastic reduced order model |           |          |
|--------------------------------|-----------|----------|
| Property                       | Mandatory | Optional |
| If #SROM property is True      |           |          |
| #moments                       | ★         |          |
| #error function weights        | ★         |          |
| #properties to match           |           | ★        |
| #sample weights                |           | ★        |

### 142 6.1.2. Examples of parameter files

143 Special instruction on how to create the parameter file that will enclose the  
 144 required properties of the selected sampling method are the following :

- A complete parameter file for e.g. Monte Carlo simulation can be defined like Fig.2(a).
- If all random parameters follow the same distribution type with the same distribution parameters then a parameter file can be defined like Fig.2(b) where the distribution type and parameters need to be defined once. In this case existence of the property "number of random parameters" is mandatory.
- For the case the number of distribution type is equal to the number of distribution parameters (Fig.2(c)) then, definition of property "number of parameters" is optional.

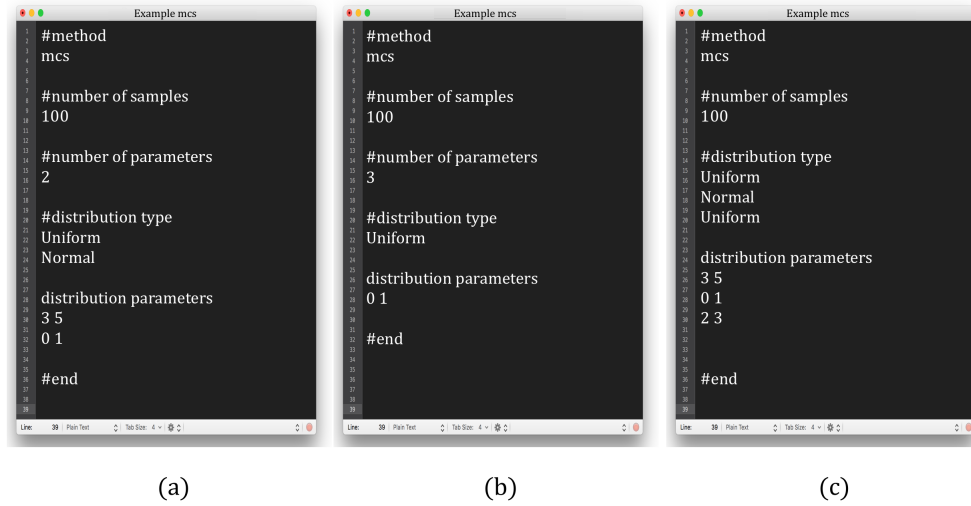
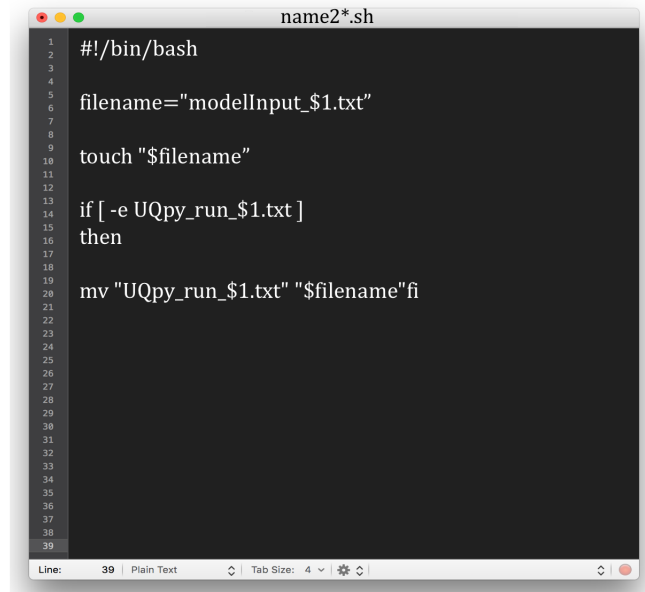


Figure 2:

## 6.2. Template Input File

The functionality of the `name2*.sh` bash shell script file is to convert the text-based output file of `UQpy` (`UQpy_run.i.txt`) that contains the realization `i` of the parameter vector into appropriate input for the analysis code. The user is responsible for creating the appropriate bash script for performing this action. For example, if the software code reads a text-based file called

161 `modellInput.i.txt` then a possible `name2*.sh` script would be the one depicted  
162 in Fig.3; it is used for renaming `UQpy_run.i.txt` to `modellInput.i.txt`.



```
1 #!/bin/bash
2 filename="modellInput_$1.txt"
3 touch "$filename"
4 if [ -e UQpy_run_$1.txt ]
5 then
6 mv "UQpy_run_$1.txt" "$filename"fi
```

Figure 3:

### 163 6.3. Template Model File

164 In order for `UQpy` to execute the software code a bash script (`name1*.sh`) is  
165 necessary.

### 166 6.4. Template Output File

167 The functionality of the `name3*.sh` bash shell script file is to convert the  
168 output of the code analysis (which can be at any format) into a text file file  
169 under the name `UQpy_eval.i.txt`, where `i` refers to the number of simulation,  
170 ready to be processed by `UQpy`. This step is required for running adaptive  
171 UQ methods as well as for post-processing of the result but in any case it  
172 is mandatory to provide such file. For example, if the software code gener-  
173 ates a text-based file called `solution.i.txt` then a possible `name3*.sh` script  
174 would be the one depicted in Fig.5; it is used for renaming `solution.i.txt` to  
175 `UQpy_eval.i.txt`.

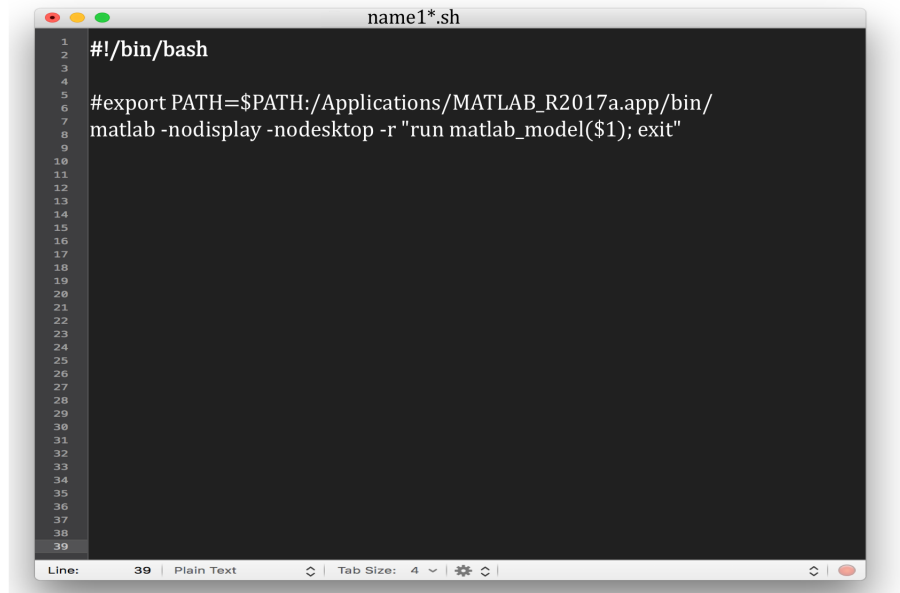


Figure 4:

## 176 7. Probabilistic sampling

| Property           | Options  |
|--------------------|--|
| #method            | 'mcs', 'lhs', 'sts', 'pss', 'mcmc'   |
| #distribution type | 'Uniform', 'Normal', 'Lognormal',<br>'Weibull', 'Beta', 'Exponential', 'Gamma' |

### 178 7.1. Latin hypercube sampling

| Property   | Type          | Options   |
|------------|---------------|---|
| #criterion | <i>string</i> | 'random', 'centered', 'maximin', 'correlate'  |
| #distance  | <i>string</i> | 'braycurtis', 'canberra', 'chebyshev', 'cosine',<br>'dice', 'euclidean', 'hamming', 'jaccard', 'cityblock',<br>'matching', 'minkowski', 'rogerstanimoto', 'correlation',<br>'sokalmichener', 'sokalsneath', 'sqeuclidean',<br>'kulsinski', 'mahalanobis', 'russellrao', 'seuclidean', |

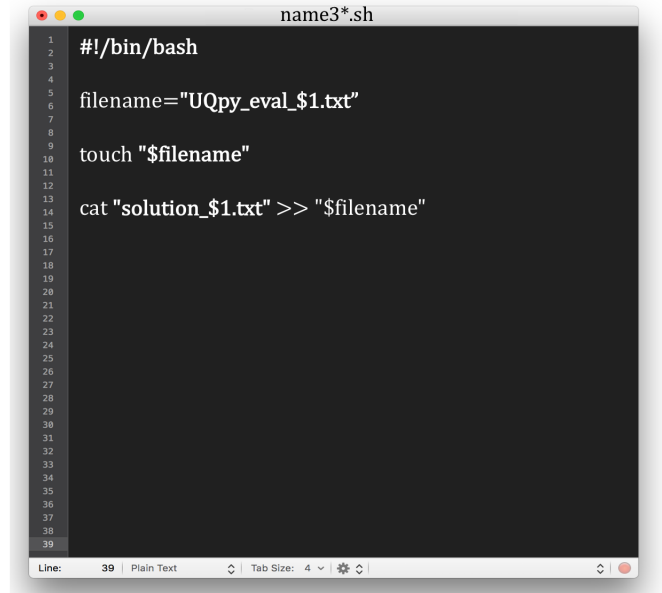


Figure 5:

## 181 7.2. Markov Chain Monte Carlo sampling

| Property               | Options  |
|------------------------|--|
| #target distribution   | 'multivariate_pdf', 'marginal_pdf', 'normal_pdf' |
| #proposal distribution | 'Uniform', 'Normal'                              |
| #algorithm             | 'MH', 'MMH'                                      |

## 184 7.3. Partially Stratified sampling

## 185 7.4. Latinized stratified sampling

## 186 7.5. Stochastic reduced order method

## 187 7.6. Adding a sampling method in UQpy

## 188 References