NASA Langley originally released the 2019 UQ Challenge problem in only native MATLAB code. Since this initial release, we have received a few requests for alternate implementations of the challenge software. Previously, we released a limited “standalone” version of the 2019 UQ Challenge problem software developed using the MATLAB Compiler to create a runtime version of the challenge code. The latest release is a fully “standalone” version of the 2019 UQ Challenge problem software and does not depend on Matlab libraries. This version of the challenge code can be run from any shell, or called from other languages, such as python. We only support MacOS and Linux. NO windows support is currently planned. Windows users are encouraged to work with the Native Matlab version originally released.

Usage example:

```
uqsim 7 aleatory.dat epistemic.dat design.dat
```

Most inputs and all outputs are passed via ascii text files. The first input, `casenum`, is an integer between 1 and 7 and defines which outputs are computed (see below).

Input files:

- aleatory.dat
- epistemic.dat
- design.dat

aleatory.dat Datafile containing the realizations of the aleatory variables. Each realization contains 5 floating point numbers.

epistemic.dat Datafile containing the realizations of the epistemic variables. Each realization contains 4 floating point numbers.

design.dat Datafile containing the realizations of the design variables. Each realization contains 9 floating point numbers.

See the included MATLAB file: **make_data.m** for an example of generating datafiles with the required format necessary to run the standalone version of NASA UQ Challenge 2019.

`Casenum:` Datafile containing a single integer value between 1 and 7.

Usage of the `casenum` is fully defined in the table below.

Casenum	**Action**	**Result: computes realizations of**
1	calls: yfun.m	uncertain subsystem, $y(a,e,t)$
2	calls: zfun.m	integrated system, $z_1(a,e,\theta,t)$, $z_2(a,e,\theta,t)$
3	calls: gfun.m	requirements vector $g(a,e,\theta)$

4	calls: yfun.m and zfun.m	*y* and *z* using a single system command
5	calls: yfun.m and gfun.m	*y* and *g* using a single system command
6	calls: zfun.m and gfun.m	*z* and *g* using a single system command
7	calls: yfun.m, zfun.m, and gfun.m	*y, z*, and *g* using a single system command

Output files:

yout – response of uncertain subsystem, $y(a,e,t)$, dimensions (5001 x 1)
z1out – response of integrated system, $z_1(a,e,\theta,t)$, dimensions ((5001 x 1)
z2out – response of integrated system, $z_2(a,e,\theta,t)$, dimensions (5001 x 1)
gout – requirements vector $g(a,e,\theta)$, dimensions (1 x 3)

The time vector file, previously called "tout", is no longer written to disk. Please create the time vector with T_initial=0, delta_T=0.001, and T_final=5. This will result in a vector with dimensions (5001 x 1). This is the time vector used internally for all simulations. A sample "tout" file is provided in the zip files below.

Python usage:

Python usage example script: see the included file "test.py" for an example of usage within python. **Note this was only tested with python 2.7**.

To run using python, use: python test.py