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. Unfortunately, we do not have the resources to develop a truly standalone version of the software. However, after a brief review of the capabilities within MATLAB, we decided to release a limited "standalone" version of the 2019 UQ Challenge problem software. This standalone version was developed using the MATLAB Compiler to create a runtime version of the challenge code, which can be run from any shell, or called from other languages, such as python.

DISCLAMER: Given our limited experience using the MATLAB Compiler and the myriad of things that could go wrong, we are releasing this standalone version with EXTREMELY limited support. We have conducted only limited internal testing, but the code appears to exactly reproduce the native MATLAB code, but our testing was limited in the types of computing hardware and inputs provided.

Requirements:

MATLAB Runtime (available free of charge).
Install: https://www.mathworks.com/products/compiler/matlab-runtime.html

Testing was preformed using only the **R2019a version of the MATLAB Runtime software**.

Tested using only the default installation path:

- On MacOS: /Applications/MATLAB/MATLAB_Runtime
- On Linux: /usr/local/MATLAB/MATLAB_Runtime/v96

Limitation of the MATLAB Runtime software:

There is a significant start-up delay on every invocation of the **MATLAB Runtime software.** This delay is similar to the delay that occurs when starting the desktop version of MATLAB. There appears to be no way to eliminate this delay. The implication is that repeatedly calling the **MATLAB Runtime software** for single realizations of a, e, or θ will be prohibitively slow. To alleviate some of this burden, the interface was written to accept an arbitrarily large number of realizations of a, e, or θ. This will reduce the relative overhead caused by this startup delay.

Usage:

All inputs and outputs are passed via ascii text files.

Input files:

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

aleatory.dat Datafile containing the realizations of the aleatory variables. Each realization contains 5 floating point numbers. You may use any integer multiple of 5 when creating aleatory.dat. For example, if n_d realizations are desired, then the number of entries in aleatory.dat must be $5n_d$

epistemic.dat Datafile containing the realizations of the epistemic variables. Each realization contains 4 floating point numbers. You may use any integer multiple of 4 when creating epistemic.dat. For example, if n_d realizations are desired, then the number of entries in epistemic.dat must be $4n_d$

design.dat Datafile containing the realizations of the design variables. Each realization contains 9 floating point numbers. You may use any integer multiple of 4 when creating design.dat. For example, if n_d realizations are desired, then the number of entries in design.dat must be $9n_d$

****IMPORTANT**** Current requirement on input datafiles: **aleatory.dat**, **epistemic.dat**, and **design.dat** is that they must all contain the SAME number of realizations. For example, if there are 20 (n_d =20) realizations requested, then **aleatory.dat** must contain 20*5=100 lines, **epistemic.dat** must contain 20*4=80 lines, and **design.dat** must contain 20*9=180 lines.

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.

casefile.dat Datafile containing a single integer value between 1 and 7.

Usage of the **casefile.dat** (this file contains a single entry from the list below):

Value	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 (n_d = number of realizations requested**):**

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

*** Use the above input and output files to validate the installation on your local machine. ***

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

MacOS command line usage:

run_nasa_uq_2019.sh /Applications/MATLAB/MATLAB_Runtime/v96 "aleatory.dat" "epistemic.dat" "design.dat" "casefile.dat"

Linux command line usage:

./run_nasa_uq_2019.sh /usr/local/MATLAB/MATLAB_Runtime/v96 "aleatory.dat" "epistemic.dat" "design.dat" "casefile.dat"

Note: Text in RED is installation specific.