**WINDMONTE Readme File**

# Introduction

WINDMONTE.py is a "WIND" tunnel "MONTE" Carlo simulation for uncertainty propagation and sensitivity analysis of wind tunnel test data.

The original code was developed independently and toward fulfillment of doctoral program requirements as published in "WIND TUNNEL DATA QUALITY ASSESSMENT AND IMPROVEMENT THROUGH INTEGRATION OF UNCERTAINTY ANALYSIS IN TEST DESIGN," PhD Dissertation by Drew Curriston, Texas A&M University, 2024.

This version is a generic copy that removes all code specific to data I/O at the Oran W. Nicks LSWT, removes code relating to the analysis of multiple objectives covered in the dissertation, provides additional in-line comments, and removes MCM convergence criteria in an effort to simplify. This version of the code limits some of the capabilities that were used in completion of the dissertation in an attempt to improve utility.

The code is separated into three regions, which are commented within the code: 1) Inputs, 2) MCM simulation, and 3) Plotting results.

# References

The primary references for uncertainty quantification are given below. These provide much more detail on terms and definitions, methodology, etc. References 1 and 2 will be replaced by an AIAA Guide that is pending publication in 2024. This Guide should be a primary reference once published as it will include the use of Monte Carlo methods and specifically address their use in wind tunnel testing.

[1] American Institute of Aeronautics and Astronautics, “Assessment of Experimental Uncertainty With Application to Wind Tunnel Testing,” Standard S-071A-1999, AIAA, 1999.

[2] American Institute of Aeronautics and Astronautics, “Assessing Experimental Uncertainty - Supplement to AIAA S-071A-1999,” Guide G-045-2003, AIAA, 2003.

[3] Advisory Group for Aerospace Research and Development (AGARD), “Wind tunnel flow quality and data accuracy requirements,” Advisory Report 184, AGARD, 1982.

[4] American Society of Mechanical Engineers, “Test Uncertainty,” Performance Test Codes PTC 19.1-2018, ASME, 6 2018.

[5] Joint Committee for Guides in Metrology, “Evaluation of Measurement Data - Supplement 1 to the "Guide to the Expression of Uncertainty in Measurement" - Propagation of distributions using a Monte Carlo method,” Guide, ISO, 2008.

[6] Curriston, D. A., “Wind Tunnel Data Quality Assessment and Improvement Through Integration of Uncertainty Analysis in Test Design,” Ph.D. thesis, Texas A&M University, College Station, TX, 2024.

# License

This software is open source and licensed under GNU GPL version 3. For more information, reference the COPYING.txt file.

# Inputs

In order to use WINDMONTE, the user must provide the following in region 1 of the WINDMONTE.py code:

* Nominal input data
  + The variable ‘testinfo’ is a dictionary that has test run information such as run number, date, model, etc.
  + The variable ‘data’ contains data for each data point. I can be imported as a .csv, .pkl file, or the code may be modified as appropriate to read test data directly. It must be given as a “list of dictionaries” where each item in the list is a datapoint, and the dictionary for each data point is the {measurand: value}. The code will check to ensure the data variable is in the right format before proceeding.
* Quantified uncertainties for error sources
  + The quantified uncertainties for each elemental systematic error source. See the code comments for how to add these.
  + The quantified uncertainties for each random error source. There are two options:
    - Define random uncertainty for the elemental error sources or measurands, then propagate through the DREs similar to the systematic uncertainty.
    - Use direct comparison of replicate data to quantify the random uncertainty of the VOIs directly, then combine the random and systematic uncertainty IAW the methodology listed in the references. This is the preferred methodology, but you must have replicate test data.
* MCM parameters such as number of Monte Carlo trials.

The user must also update the data reduction equations that map the measured input data to the test results or variables of interest (VOIs). These are in the “eval” function of DREs.py. Replace everything within DREs.eval() with a validated and verified set of DREs for your test that accept the “data” and “testinfo” variables as inputs and output results in the same “list of dictionaries” format as the inputs.

Example input files have been provided, along with example elemental uncertainties and DREs. The code should run as is with the provided example.

# Object Oriented Programming Structure

WINDMONTE is written in an object-oriented programming structure to help make the code more robust. The test run data RUNDATA, each DATAPOINT, and each VOI are defined in utilities.py as classes that have a specific structure, attributes, and behaviors. This forces the program to save uncertainty and other data in a specific format so that the MCM simulations and data plotting will not have as many issues that must be debugged by differences in the way data is being input. Each class also has behaviors such as specific plots so that VOIs and points may be selected easily in the included GUI for plotting with fewer errors. More to follow on this…

# Outputs

WINDMONTE saves and output file in .pkl format that includes a RUNDATA variable which contains all nominal and uncertainty interval data for all data points in the test run, and the U\_systematic and U\_random variables which have information on the elemental error sources. This data may be loaded into other code using pickle/dill, or visualized using the WINDMONTE GUI. The GUI allows the user to load any output file from WINDMONTE and plot data either for the entire test run or for specific data points and VOIs.