**Python code for the 2-D wind multipliers**

The Python code was produced for the 2-D wind multipliers based on the existing Octave code written by Chris Thomas (2009). To keep consistence, the Python code was structured and functioned closely to the Octave code.

**py\_north.py** - the main code for calculating 2-D multipliers based on the input DEM and the wind direction (north).

To adopt Chris Thomas (2009)’ modifications, the code does not rotate matrices but extracts a line for that direction from the array holding the DEM and operates on that line to produce the hill shape multipliers. The calculated line of multipliers is then written back to the same array that holds the DEM data. The smoothing is done upon output. In the end, outputs including smoothed and unsmoothed data are written into ascii files.

It calls the following functions:

**ascii\_read.py** # read input DEM data

**make\_path.py** # generate indices of a data line depending on the direction

**multiplier\_calc.py** # calculate the multipliers for a data line extracted from the dataset

multiplier\_calc.py needs to call three functions:

**Mh.py** # calculate Mh using the simpler formula modified by C.Thomas 2009

**findpeaks.py** # get the indices of the ridges in a data line

**findvalleys.py** # get the indices of the valleys in a data line

Codes for other wind directions are named similarly: py\_south.py, py\_west.py, py\_east.py, py\_northeast.py, py\_northwest.py, py\_southeast.py, py\_southwest.py.

**run\_all.py** will run codes from all eight directions.

**compare\_results.py** compares outputs from the Python and the Octave codes. It calls output\_oct.py and output\_py.py to read input ascii files, which are multiplier outputs. Please ensure output files are specified accordingly to the wind direction in compare\_results.py. The comparison results include differences between each element in either smoothed or unsmoothed outputs calculated by the Python and Octave codes. For wind from the eight directions, the comparison is done individually and the result is saved separately.

Directory structure:

**../Input**/**:** store input DEM.

**../python code**/: store all the Python codes including main and functions, as well as comparison code.

**../python\_output**/: generated automatically to store output files.

**../comparison\_output**/: store comparisons between outputs from the Python and the Octave codes

**Notes:**

Comparisons show small discrepancies between outputs from the Python and the Octave codes. The difference varies depending on the wind direction. The number of inconsistent data is up to 0.13% of the total data (west wind) with the greatest magnitude of 0.02, which is 2% of the minimum magnitude of output data.

The discrepancies are believed due to the inexact representation of decimal fractions in the IEEE floating point standard[[1]](#footnote-1) and errors introduced when scaling by powers of ten[[2]](#footnote-2). Regardless of implementation language, similar problems would arise.

It also should be noted that some Python functions have slightly different functionality. For example, for values exactly halfway between rounded decimal values, Numpy rounds to the nearest even value. Thus, to maintain consistence with other languages, these functions should be used with some modifications if needed.

1. “Lecture Notes on the Status of IEEE 754”, William Kahan, http://www.cs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF [↑](#footnote-ref-1)
2. Numpy and Scipy Documentation, http://scipy.org/doc [↑](#footnote-ref-2)