Using Python for Wind Resource Assessment

Davis, Neil

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
The WAsP Model simulates the surface affects on the wind speed distribution. This distribution is found to be Weibull shaped, and surface affects can modify both the scale and shape parameters of the distribution. The WAsP model has been the leading software for wind resource assessment since its release in 1989.

Most users of WAsP do not call the Fortran itself, but interface with the model through an advanced Windows GUI program. This program provides GIS capabilities to add topographic information along with knobs to tweak the different model parameters, and set the required fields. In addition to the GUI provided by DTU, the WAsP model has been included in several other wind resource planning tools.

The WAsP GUI code has more than 1200 monthly users, who appreciate its simple design and ease of use. However, it was found that researchers at DTU were less likely to use the GUI tool for several reasons.

1. Researchers were increasingly using Linux and Mac environments.
2. Researchers wanted to execute large numbers of simulations using scripts.
3. Computers were now much faster and able to run on multiple cores.
4. Researchers used Python for analysis and wanted that for WAsP.

Due to these reasons, we investigated different ways of combining WAsP with Python to enable a cross platform solution for researchers.

In addition, it was hoped that by moving to a Python framework, the model could be verified and validated more easily through the use of unit tests and comprehensive evaluation against measurements on a routine basis.

To create an interface to WAsP that would be useful to today's researchers, we turned to F2Py to wrap the Fortran code, and developed custom classes for the different steps in the wind resource assessment workflow. This enabled us to create a simple 7 step process for running a resource assessment project, provided that you had the correct input files.

The class based structure we developed allowed us to hide the long subroutine calls behind easier to use interfaces for the user. In the code to the right, the generalize function calls the atlas_nt function from above, but here only requires 5 arguments.

In addition to simplifying the interface, the Python development has allowed us to easily run WAsP on our HPC system, allowing for us to run simulations with millions of variations across many processes.