Using Python for Wind Resource Assessment

Davis, Neil

Publication date: 2018

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

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.

The WAsP Model GUI

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.

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.

F2Py vs CTYPES for Research Code

In our case we found that F2py is an invaluable tool for wrapping Fortran. The killer features for us are the ease of use, in particular the ability to wrap routines without having to code any Python and the ability to build the Fortran as part of the setup.py process. There are however some limitations to the use of F2py that we are experiencing, and would like to help rectify. First is the lack of derived type support, which is a feature that is becoming common in our codebase. Second is the dependency on specific Python and Numpy versions. For Windows, there is a desire for a single .dll file that can be used for both the GUI and the Python implementation.

Lessons Learned

Spend upfront time on design to ensure that you understand your model code, and the specific pieces you wish to make available to Python. Develop a class-based structure to enable easier changes to your Fortran API, and allow your backend structure to change. Don’t be afraid to start with Fortran like data-structures, we have successfully moved to more Pythonic structures on the Python side hidden by the class variables. Develop any new I/O formats first, and ensure they are in an open package. This is why we made the pywasp package, since we want others to use our formats even if they aren’t paying for the use of the model.