windpowerlib Documentation

Release

Uwe Krien, oemof developing group

Jun 26, 2017

Contents

Contents:

CHAPTER 1

Getting started

The windpowerlib is designed to calculate feedin time series of wind power plants. The windpowerlib is an out-take from the feedinlib (windpower and pv) to build up a community concentrating on wind power models.

The windpowerlib is ready to use but may have some teething troubles. The used model is still very simple but we found some new contributors and the windpowerlib will improve soon. If you are interested in wind models and want to improve the existing model, publish an alternative model or extend it to wind farms do not hesitate to contact us.

Table of contents

Introduction

Having weather data sets you can use the windpowerlib to calculate the electrical output of common wind power plants. Basic parameters for many manufacturers are provided with the library so that you can start directly using one of these parameter sets. Of course you are free to add your own parameter set.

The cp-values for the wind turbines are provided by the Reiner Lemoine Institut and can be found here:

- http://vernetzen.uni-flensburg.de/~git/cp_values.csv

Actual Release

Download/Install: https://pypi.python.org/pypi/windpowerlib/

Documentation: http://pythonhosted.org/windpowerlib/

Developing Version

Clone/Fork: https://github.com/wind-python/windpowerlib

Documentation: http://windpowerlib.readthedocs.org/en/latest/

As the windpowerlib started with contributors from the oemof developer group we use the same developer rules: http://oemof.readthedocs.io/en/stable/developer_notes.html

Installation

Using the windpowerlib

So far, the windpowerlib is mainly tested on python 3.4 but seems to work down to 2.7.

Install the windpowerlib using pip3.

```
pip3 install windpowerlib
```

Developing the Windpowerlib

If you have push rights clone this repository to your local system.

```
git clone git@github.com:oemof/windpower.git
```

If you do not have push rights, fork the project at github, clone your personal fork to your system and send a pull request.

If the project is cloned you can install it using pip3 with the -e flag. Using this installation, every change is applied directly.

```
pip3 install -e <path/to/the/windpowerlib/root/dir>
```

Optional Packages

To see the plots of the example file one should install the matplotlib package.

Matplotlib can be installed using pip3 but some Linux users reported that it is easier and more stable to use the pre-built packages of your Linux distribution.

http://matplotlib.org/users/installing.html

Example

Download the .csv data and the example file and execute it:

https://github.com/wind-python/windpowerlib/tree/master/example

Basic Usage

You need three steps to get a time series.

Warning

Be accurate with the units. In the example all units are given without a prefix.

- pressure [Pa]
- wind speed [m/s]
- installed capacity [W]
- nominal power [W]
- area [m^2]
- temperature [°C]

You can also use kW instead of W but you have to make sure that all units are changed in the same way.

1. Initialise your Turbine or Module

To initialise your specific turbine you need a dictionary that contains your basic parameters.

The most import parameter is the name of the turbine to get technical parameters from the provided libraries. Use the *get_wind_pp_types* function to get a list of all available converter types.

```
from windpowerlib import basicmodel
basicmodel.get_wind_pp_types()
```

The other parameters are related to location of the plant like diameter of the rotor or the hub height. The existing model needs the following parameters:

- h_hub: height of the hub in meters
- d_rotor: diameter of the rotor in meters
- wind_conv_type: Name of the wind converter according to the list in the csv file

```
your_wind_turbine = basicmodel.SimpleWindTurbine(**your_parameter_set)
```

If you pass a valid model the nominal_power and the cp-values are read from a file. If you want to use your own converter you can pass your own cp-series and nominal power instead of the converter type. This can be done with a dictionary (as shown above) or directly.

```
your_wind_turbine = basicmodel.SimpleWindTurbine(cp_values=my_cp_values,
                                                 nominal_power=my_nominal_power,
                                                 d_rotor=my_d_rotor,
                                                 h_hub=my_h_hub)
```

2. Get your Feedin Time Series

To get your time series you have to pass a weather DataFrame (or dictionary) to your model.The DataFrame needs to have pressure, wind speed, temperature and the roughness length. The following names are used:

- 'pressure'
- 'temp_air'
- 'v_wind'
- 'z0'

In an additional dictionary the height of the weather data has to be defined. The example shows a dictionary for the coasdat2 weather data set:

```
coastDat2 = {
    'dhi': 0,
    'dirhi': 0,
    'pressure': 0,
    'temp_air': 2,
    'v_wind': 10,
    'Z0': 0}
```

If your weather DataFrame has different column names you have to rename them. This can easily be done by using a conversion dictionary:

```
name_dc = {
    'your pressure data set': 'pressure',
    'your ambient temperature': 'temp_air',
    'your wind speed': 'v_wind',
    'your roughness length': 'z0'}

your_weather_DataFrame.rename(columns=name_dc)
```

Now you can pass the weather data to the output method:

```
your_wind_turbine.turbine_power_output(weather=weather_df, data_height=coastDat2)
```

You will get the output of one wind_turbine in [W] if you followed the united recommendations from above.

CHAPTER 2

What's New

These are new features and improvements of note in each release

Releases

v0.0.4 ()

New features

- cp-values database file is now part of the repository and will be installed using pip/setup.py. The former download server was down due to technical problems and it might be safer not to be reliant on an external server.

Documentation

Testing

Bug fixes

Other changes

Contributors

- Uwe Krien

v0.0.3 (November 18, 2016)

Other changes

Allow installation of windpowerlib for python versions >3.4 Import requests package instead of urllib5

Contributors

- Uwe Krien
- Stephen Bosch
- Birgit Schachler

Comment

Release of v0.0.2 has been skipped due to a mistake in the release process.

v0.0.1 (August 29, 2016)

The wind part of the feedinlib was transferd to the winpowerlib.

Contributors

- Uwe Krien

CHAPTER 3

Model Description

So far only two simple models are provided but there will improvements and a larger variety in future version hopefully with the help of the github community.

Wind Model

The wind model is a simple model based on the cp-values of a specific type of a wind turbine. The cp-values are provided by the manufacturer of the wind turbine as a list of cp-values for discrete wind speeds in steps of 0.5 or 1 m/s. The feedinlib makes an linear interpolation of these values to get a continuous cp-curve over the wind speeds.

$$P_{wpp} = \frac{1}{8} \cdot \rho_{air,hub} \cdot d_{rotor}^2 \cdot \pi \cdot v_{wind,hub}^3 \cdot cp\left(v_{wind,hub}\right)$$

with d_{rotor} the diameter of the rotor in meters, $\rho_{air,hub}$ the density of the air at hub height, $v_{wind,hub}$ the wind speed at hub height and cp the cp-values against the wind speed.

The wind speed at hub height is determined by the following equation, assuming a logarithmic wind profile.

$$v_{wind,hub} = v_{wind,data} \cdot \frac{\ln\left(\frac{h_{hub}}{z_0}\right)}{\ln\left(\frac{h_{wind,data}}{z_0}\right)}$$

with $v_{wind,hub}$ the wind speed at the height of the weather model or measurement, h_{hub} the height of the hub and $h_{wind,data}$ the height of the wind speed measurement or the height of the wind speed within the weather model.

The density of the air is calculated assuming a temperature gradient of -6.5 K/km and a pressure gradient of -1/8 hPa/m.

$$T_{hub} = T_{air,data} - 0.0065 \cdot (h_{hub} - h_{T,data})$$

with $T_{air,data}$ the temperature at the height of the weather model or measurement, h_{hub} the height of the hub and $h_{T,Data}$ the height of temperature measurement or the height of the temperature within the weather model.

$$\rho_{air,hub} = \left(p_{data}/100 - (h_{hub} - h_{p,data}) * \frac{1}{8}\right) / \left(2.8706 \cdot T_{hub}\right)$$

with p_{data} the pressure at the height of the weather model or measurement, T_{hub} the temperature of the air at hub height, h_{hub} the height of the hub and $h_{p,data}$ the height of pressure measurement or the height of pressure within the weather model.

Weather Data

The weather data manly used by the oemof developing group is the coastDat2 of the Helmholtz-Zentrum Geesthacht

http://wiki.coast.hzg.de/pages/viewpage.action?pageId=4751533

Due to licence problems we are not allowed to ship the dataset with the windpowerlib. Please contact the Helmholtz-Zentrum Geesthacht (HZG), the data set for Europe might be free for non commercial use.

CHAPTER 4

windpowerlib

windpowerlib package

Submodules

windpowerlib.basicmodel module

class windpowerlib.basicmodel.**SimpleWindTurbine**(*wind_conv_type=None*, *h_hub=None*, *d_rotor=None*, *cp_values=None*, *nominal_power=None*)

Bases: object

Model to determine the output of a wind turbine

Parameters

- **wind_conv_type** (*string*) – Name of the wind converter type. Use get_wind_pp_types() to see a list of all possible wind converters.
- **h_hub** (*float*) – Height of the hub of the wind turbine.
- **d_rotor** (*float*) – Diameter of the rotor.
- **cp_values** (*pandas.DataFrame*) – The index should be the wind speed and a column should be named 'cp'.
- **nominal_power** (*float*) – The nominal output of the wind power plant.

wind_conv_type
string – Name of the wind converter type. Use get_wind_pp_types() to see a list of all possible wind converters.

h_hub
float – Height of the hub of the wind turbine.

d_rotor
float – Diameter of the rotor.

cp_values
pandas.DataFrame – The index should be the wind speed and a column should be named 'cp'.

nominal_power
float – The nominal output of the wind power plant.

Examples

```
>>> from windpowerlib import basicmodel
>>> enerconE126 = {
...     'h_hub': 135,
...     'd_rotor': 127,
...     'wind_conv_type': 'ENERCON E 126 7500'}
>>> e126 = basicmodel.SimpleWindTurbine(**enerconE126)
>>> print(e126.d_rotor)
127
```

cp_series(*v_wind*)
Interpolates the cp value as a function of the wind velocity between data obtained from the power curve of the specified wind turbine type.

Parameters **v_wind** (*pandas.Series or numpy.array*) – Wind speed at hub height [m/s]

Returns

- *numpy.array* – cp values, wind converter type, installed capacity
- *>>> import numpy*
- *>>> from windpowerlib import basicmodel*
- *>>> e126 = basicmodel.SimpleWindTurbine('ENERCON E 126 7500')*
- *>>> v_wind = numpy.array([1,2,3,4,5,6,7,8])*
- *>>> print(e126.cp_series(v_wind))*
- *[0. 0. 0.191 0.352 0.423 0.453 0.47 0.478]*

fetch_wpp_data(***kwargs*)
Fetch data of the requested wind converter.

Returns cp values and the nominal power of the requested wind converter

Return type tuple with pandas.DataFrame and float

Examples

```
>>> from windpowerlib import basicmodel
>>> e126 = basicmodel.SimpleWindTurbine('ENERCON E 126 7500')
>>> print(e126.cp_values.cp[5.0])
0.423
>>> print(e126.nominal_power)
7500000.0
```

rho_hub(*weather*, *data_height*)

Calculates the density of air in kg/m^3 at hub height. (temperature in K, height in m, pressure in Pa)

Parameters

- **weather** (*DataFrame or Dictionary*) – Containing columns or keys with the timeseries for Temperature (temp_air) and pressure (pressure).
- **data_height** (*DataFrame or Dictionary*) – Containing columns or keys with the height of the measurement or model data for temperature (temp_air) and pressure (pressure).

Returns density of air in kg/m^3 at hub height

Return type float

Notes

Assumptions:

- Temperature gradient of -6.5 K/km
- Pressure gradient of -1/8 hPa/m

The following equations are used[22]:

$$T_{hub} = T_{air,data} - 0.0065 \cdot (h_{hub} - h_{T,data})$$

$$p_{hub} = \left(p_{data}/100 - (h_{hub} - h_{p,data}) * \frac{1}{8} \right) / (2.8706 \cdot T_{hub})$$

with T: temperature [K], h: height [m], p: pressure [Pa]

ToDo: Check the equation and add references.

References

See also:

v_wind_hub()

turbine_power_output(*weather*, *data_height*)
Calculates the power output in W of one wind turbine.

Parameters

- **weather** (*feedinlib.weather.FeedinWeather object*) – Instance of the feedinlib weather object (see class `FeedinWeather` for more details)
- **data_height** (*dictionary*) – Containing the heights of the weather measurements or weather model in meters with the keys of the data parameter
- **TODO Move the following parameters to a better place** (*#*) –

Returns Electrical power of the wind turbine

Return type pandas.Series

[22] ICAO-Standardatmosphäre (ISA). http://www.deutscher-wetterdienst.de/lexikon/download.php?file=Standardatmosphaere.pdf

Notes

The following equation is used for the power output P_{wpp}[21]:

$$P_{wpp} = \frac{1}{8} \cdot \rho_{air,hub} \cdot d_{rotor}^2 \cdot \pi \cdot v_{wind}^3 \cdot cp\left(v_{wind}\right)$$

with: v: wind speed [m/s], d: diameter [m], ρ: density [kg/m^3]

ToDo: Check the equation and add references.

References

v_wind_hub(*weather*, *data_height*)

Calculates the wind speed in m/s at hub height.

Parameters

- **weather** (*DataFrame or Dictionary*) – Containing columns or keys with the timeseries for wind speed (v_wind) and roughness length (z0).
- **data_height** (*DataFrame or Dictionary*) – Containing columns or keys with the height of the measurement or model data for temperature (temp_air) and pressure (pressure).

Returns wind speed [m/s] at hub height

Return type float

Notes

The following equation is used for the logarithmic wind profile[20]:

$$v_{wind,hub} = v_{wind,data} \cdot \frac{\ln\left(\frac{h_{hub}}{z_0}\right)}{\ln\left(\frac{h_{data}}{z_0}\right)}$$

with: v: wind speed [m/s], h: height [m], z0: roughness length [m]

h_{data} is the height in which the wind velocity is measured. (height in m, velocity in m/s)

ToDo: Check the equation and add references.

References

See also:

rho_hub()

windpowerlib.basicmodel.**get_wind_pp_types**(*print_out=True*)

Get the names of all possible wind converter types.

Parameters **print_out** (*boolean (default: True)*) – Directly prints the list of types if set to True.

[21] Gasch R., Twele J.: "Windkraftanlagen". 6. Auflage, Wiesbaden, Vieweg + Teubner, 2010, pages 35ff, 208
[20] Gasch R., Twele J.: "Windkraftanlagen". 6. Auflage, Wiesbaden, Vieweg + Teubner, 2010, page 129

Examples

```
>>> from windpowerlib import basicmodel
>>> valid_types_df = basicmodel.get_wind_pp_types(print_out=False)
>>> valid_types_df.shape
(91, 2)
>>> print(valid_types_df.iloc[5])
rli_anlagen_id    DEWIND D8 2000
p_nenn                      2000
Name: 5, dtype: object
```

windpowerlib.basicmodel.**read_wpp_data**(***kwargs*)

Fetch cp values from a file or download it from a server.

The files are located in the data folder of the package root.

Returns cp values, wind converter type, installed capacity or the full table if the given wind converter cannot be found in the table.

Return type pandas.DataFrame

Other Parameters

- **datapath** (*string, optional*) – Path where the cp file is stored. Default: '$PACKAGE_ROOT/data'
- **filename** (*string, optional*) – Filename of the cp file.

Module contents

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index