windpowerlib Documentation

Release

oemof developer group

Jun 26, 2017

Contents

Contents:

CHAPTER 1

Getting started

windpowerlib

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.

Introduction

Having weather data sets you can use the windpowerlib to calculate the electrical output of common wind turbines. 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. For a quick start download the example weather data and basic example file and execute it:

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

Documentation

Full documentation can be found at readthedocs. Use the project site of readthedocs to choose the version of the documentation.

Contributing

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

If you are interested in wind models and want to help improve the existing model do not hesitate to contact us. As the windpowerlib started with contributors from the oemof developer group we use the same developer rules.

Installation

Install the windpowerlib using pip3.

```
pip3 install windpowerlib
```

So far, the windpowerlib is mainly tested on python 3.4 but seems to work down to 2.7. Please see the installation page of the oemof documentation for complete instructions on how to install python on your operating system.

Optional Packages

To see the plots of the example file you 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

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

Classes

wind_turbine.WindTurbine(turbine_name, ...)	Defines a standard set of wind turbine attributes.
modelchain.Modelchain(wind_turbine[, ...])	Model to determine the output of a wind turbine

windpowerlib.wind_turbine.WindTurbine

class windpowerlib.wind_turbine.**WindTurbine**(*turbine_name*, *hub_height*, *d_rotor*, *cp_values=None*, *p_values=None*, *nominal_power=None*, *fetch_curve='cp'*)

Defines a standard set of wind turbine attributes.

Parameters

- **turbine_name** (*string*) – Name of the wind turbine type. Use get_turbine_types() to see a list of all wind turbines for which power (coefficient) curve data is provided.
- **hub_height** (*float*) – Hub height of the wind turbine in m.
- **d_rotor** (*float*) – Diameter of the rotor in m.
- **cp_values** (*pandas.DataFrame*) – Power coefficient curve of the wind turbine. The indices of the DataFrame are the corresponding wind speeds of the power coefficient curve, the power coefficient values are listed in the column 'cp'. Default: None.
- **p_values** (*pandas.DataFrame*) – Power curve of the wind turbine. The indices of the DataFrame are the corresponding wind speeds of the power curve, the power values are listed in the column 'P'. Default: None.
- **nominal_power** (*float*) – The nominal output of the wind turbine in kW.
- **fetch_curve** (*string*) – Parameter to specify whether the power or power coefficient curve should be retrieved from the provided turbine data. Default: cp.

turbine_name

string – Name of the wind turbine type. Use get_turbine_types() to see a list of all wind turbines for which power (coefficient) curve data is provided.

hub_height
float – Hub height of the wind turbine in m.

d_rotor
float – Diameter of the rotor in m.

cp_values
pandas.DataFrame – Power coefficient curve of the wind turbine. The indices of the DataFrame are the corresponding wind speeds of the power coefficient curve, the power coefficient values are listed in the column 'cp'. Default: None.

p_values
pandas.DataFrame – Power curve of the wind turbine. The indices of the DataFrame are the corresponding wind speeds of the power curve, the power values are listed in the column 'P'. Default: None.

nominal_power
float – The nominal output of the wind turbine in kW.

power_output
pandas.Series – The calculated power output of the wind turbine.

Examples

```
>>> from windpowerlib import wind_turbine
>>> enerconE126 = {
...     'hub_height': 135,
...     'd_rotor': 127,
...     'turbine_name': 'ENERCON E 126 7500'}
>>> e126 = wind_turbine.WindTurbine(**enerconE126)
>>> print(e126.d_rotor)
127
```

__init__(*turbine_name*, *hub_height*, *d_rotor*, *cp_values=None*, *p_values=None*, *nominal_power=None*, *fetch_curve='cp'*)

Methods

__init__(turbine_name, hub_height, d_rotor)	
fetch_turbine_data()	Fetches data of the requested wind turbine.

windpowerlib.modelchain.Modelchain

class windpowerlib.modelchain.**Modelchain**(*wind_turbine*, *obstacle_height=0*, *wind_model='logarithmic'*, *rho_model='barometric'*, *temperature_model='gradient'*, *power_output_model='cp_values'*, *density_corr=False*, *hellman_exp=None*, *hellman_z0=None*)

Model to determine the output of a wind turbine

Parameters

- **wind_turbine** (WindTurbine) – A *WindTurbine* object representing the wind tur-

bine.

- **obstacle_height** (*float*) – Height of obstacles in the surrounding area of the wind turbine in m. Set *obstacle_height* to zero for wide spread obstacles. Default: 0.
- **wind_model** (*string*) – Parameter to define which model to use to calculate the wind speed at hub height. Valid options are 'logarithmic' and 'hellman'. Default: 'logarithmic'.
- **rho_model** (*string*) – Parameter to define which model to use to calculate the density of air at hub height. Valid options are 'barometric' and 'ideal_gas'. Default: 'barometric'.
- **temperature_model** (*string*) – Parameter to define which model to use to calculate the temperature at hub height. Valid options are 'gradient' and 'interpolation'. Default: 'gradient'.
- **power_output_model** (*string*) – Parameter to define which model to use to calculate the turbine power output. Valid options are 'cp_values' and 'p_values'. Default: 'cp_values'.
- **density_corr** (*boolean*) – If the parameter is True the density corrected power curve is used for the calculation of the turbine power output. Default: False.
- **hellman_exp** (*float*) – The Hellman exponent, which combines the increase in wind speed due to stability of atmospheric conditions and surface roughness into one constant. Default: None.
- **hellman_z0** (*float*) – Roughness length. Default: None.

wind_turbine
WindTurbine – A *WindTurbine* object representing the wind turbine.

obstacle_height
float – Height of obstacles in the surrounding area of the wind turbine in m. Set *obstacle_height* to zero for wide spread obstacles. Default: 0.

wind_model
string – Parameter to define which model to use to calculate the wind speed at hub height. Valid options are 'logarithmic' and 'hellman'. Default: 'logarithmic'.

rho_model
string – Parameter to define which model to use to calculate the density of air at hub height. Valid options are 'barometric' and 'ideal_gas'. Default: 'barometric'.

temperature_model
string – Parameter to define which model to use to calculate the temperature at hub height. Valid options are 'gradient' and 'interpolation'. Default: 'gradient'.

power_output_model
string – Parameter to define which model to use to calculate the turbine power output. Valid options are 'cp_values' and 'p_values'. Default: 'cp_values'.

density_corr
boolean – If the parameter is True the density corrected power curve is used for the calculation of the turbine power output. Default: False.

hellman_exp
float – The Hellman exponent, which combines the increase in wind speed due to stability of atmospheric conditions and surface roughness into one constant. Default: None.

hellman_z0
float – Roughness length. Default: None.

power_output
pandas.Series – Electrical power output of the wind turbine in W.

Examples

```
>>> from windpowerlib import modelchain
>>> from windpowerlib import wind_turbine
>>> enerconE126 = {
...     'hub_height': 135,
...     'd_rotor': 127,
...     'wind_conv_type': 'ENERCON E 126 7500'}
>>> e126 = wind_turbine.WindTurbine(**enerconE126)
>>> modelchain_data = {'rho_model': 'ideal_gas',
...     'temperature_model': 'interpolation'}
>>> e126_md = modelchain.Modelchain(e126, **modelchain_data)
>>> print(e126.d_rotor)
127
```

__init__(*wind_turbine*, *obstacle_height=0*, *wind_model='logarithmic'*, *rho_model='barometric'*, *temperature_model='gradient'*, *power_output_model='cp_values'*, *density_corr=False*, *hellman_exp=None*, *hellman_z0=None*)

Methods

__init__(wind_turbine[, obstacle_height, ...])	
rho_hub(weather, data_height)	Calculates the density of air at hub height.
run_model(weather, data_height)	Runs the model.
turbine_power_output(v_wind, rho_hub)	Calculates the power output of the wind turbine.
v_wind_hub(weather, data_height)	Calculates the wind speed at hub height.

CHAPTER 4

Density

Functions for calculating air density at hub height.

density.temperature_gradient(temp_air, ...)	Calculates the temperature at hub height using a linear temperature gradient.
density.temperature_interpol(temp_air_1, ...)	Calculates the temperature at hub height by inter- or extrapolation.
density.rho_barometric(pressure, ...)	Calculates the density of air at hub height using the barometric height equation.
density.rho_ideal_gas(pressure, ...)	Calculates the density of air at hub height using the ideal gas equation.

windpowerlib.density.temperature_gradient

windpowerlib.density.**temperature_gradient**(*temp_air*, *temp_height*, *hub_height*)

Calculates the temperature at hub height using a linear temperature gradient.

A linear temperature gradient of -6.5 K/km is assumed. This function is carried out when the parameter *temperature_model* of an instance of the *Modelchain* class is 'temperature_gradient'.

Parameters

- **temp_air** (*pandas.Series or array*) – Air temperature in K.
- **temp_height** (*float*) – Height in m for which the parameter *temp_air* applies.
- **hub_height** (*float*) – Hub height of wind turbine in m.

Returns Temperature at hub height in K.

Return type pandas.Series or array

Notes

The following equation is used[1]:

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

with: T: temperature [K], h: height [m]

$h_{T,data}$ is the height in which the temperature is measured.

Assumptions:

•Temperature gradient of -6.5 K/km (-0.0065 K/m)

References

windpowerlib.density.temperature_interpol

windpowerlib.density.**temperature_interpol**(*temp_air_1*, *temp_air_2*, *temp_air_height_1*, *temp_air_height_2*, *hub_height*)

Calculates the temperature at hub height by inter- or extrapolation.

This fuction is carried out when the parameter *temperature_model* of an instance of the `Modelchain` class is 'interpolation'.

Parameters

- **temp_air_1** (*pandas.Series or array*) – Air temperature.
- **temp_air_2** (*pandas.Series or array*) – Second air temperature for interpolation.
- **temp_air_height_1** (*float*) – Height for which the parameter *temp_air_1* applies.
- **temp_air_height_2** (*float*) – Height for which the parameter *temp_air_2* applies.
- **hub_height** (*float*) – Hub height of wind turbine in m.

Returns Temperature at hub height.

Return type pandas.Series or array

Notes

The following equation is used:

$$T_{hub} = (T_2 - T_1)/(h_2 - h_1) * (h_{hub} - h_1) + T_1$$

with: T: temperature, h: height

Assumptions:

•linear temperature gradient

[1] ICAO-Standardatmosphäre (ISA). http://www.dwd.de/DE/service/lexikon/begriffe/S/Standardatmosphaere_pdf.pdf?__blob=publicationFile&v=3

windpowerlib.density.rho_barometric

windpowerlib.density.**rho_barometric**(*pressure*, *pressure_height*, *hub_height*, *T_hub*)

Calculates the density of air at hub height using the barometric height equation.

This fuction is carried out when the parameter *rho_model* of an instance of the `Modelchain` class is 'barometric'.

Parameters

- **pressure** (*pandas.Series or array*) – Pressure in Pa.
- **pressure_height** (*float*) – Height in m for which the parameter *pressure* applies.
- **hub_height** (*float*) – Hub height of wind turbine in m.
- **T_hub** (*pandas.Series or array*) – Temperature at hub height in K.

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

Return type pandas.Series

Notes

The following equation is used[1],[2] :

$$\rho_{hub} = \left(p/100 - (h_{hub} - h_{p,data}) \cdot \frac{1}{8}\right) \cdot \frac{\rho_0 T_0 \cdot 100}{p_0 T_{hub}}$$

with: T: temperature [K], h: height [m], ρ: density [kg/m^3], p: pressure [Pa]

$h_{p,data}$ is the height of the measurement or model data for pressure, p_0 the ambient air pressure, ρ_0 the ambient density of air, T_0 the ambient temperature and T_{hub} the temperature at hub height.

Assumptions:

•Pressure gradient of -1/8 hPa/m

References

windpowerlib.density.rho_ideal_gas

windpowerlib.density.**rho_ideal_gas**(*pressure*, *pressure_height*, *hub_height*, *T_hub*)

Calculates the density of air at hub height using the ideal gas equation.

This fuction is carried out when the parameter *rho_model* of an instance of the `Modelchain` class is 'ideal_gas'.

Parameters

- **pressure** (*pandas.Series or array*) – Pressure in Pa.
- **pressure_height** (*float*) – Height in m for which the parameter *pressure* applies.
- **hub_height** (*float*) – Hub height of wind turbine in m.
- **T_hub** (*pandas.Series or array*) – Temperature at hub height in K.

[1] Hau, E.: "Windkraftanlagen - Grundlagen, Technik, Einsatz, Wirtschaftlichkeit". 4. Auflage, Springer-Verlag, 2008, p. 560

[2] Deutscher Wetterdienst: http://www.dwd.de/DE/service/lexikon/begriffe/D/Druckgradient_pdf.pdf?__blob=publicationFile&v=4

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

Return type pandas.Series

Notes

The following equations are used[1],[2]:

$$\rho_{hub} = p_{hub}/(R_s T_{hub})$$

and[3]:

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

with: T: temperature [K], ρ: density [kg/m^3], p: pressure [Pa]

R_s is the specific gas constant of dry air (287.058 J/(kg*K)) and p_{hub} is the pressure at hub height.

References

[1] Ahrendts J., Kabelac S.: "Das Ingenieurwissen - Technische Thermodynamik". 34. Auflage, Springer-Verlag, 2014, p. 23

[2] Biank, M.: "Methodology, Implementation and Validation of a Variable Scale Simulation Model for Windpower based on the Georeferenced Installation Register of Germany". Master's Thesis at RLI, 2014, p. 57

[3] Deutscher Wetterdienst: http://www.dwd.de/DE/service/lexikon/begriffe/D/Druckgradient_pdf.pdf?__blob=publicationFile&v=4

CHAPTER 5

Wind speed

Functions for calculating wind speed at hub height.

wind_speed.logarithmic_wind_profile(v_wind, ...)	Calculates the wind speed at hub height using a logarithmic wind profile.
wind_speed.v_wind_hellman(v_wind, ...[, ...])	Calculates the wind speed at hub height using the hellman equation.

windpowerlib.wind_speed.logarithmic_wind_profile

windpowerlib.wind_speed.**logarithmic_wind_profile**(*v_wind*, *v_wind_height*, *hub_height*, *z_0*, *obstacle_height=0*)

Calculates the wind speed at hub height using a logarithmic wind profile.

The logarithmic height equation is used. There is the possibility of including the height of the surrounding obstacles in the calculation. This function is carried out when the parameter *wind_model* of an instance of the *Modelchain* class is 'logarithmic' or 'logarithmic_closest'.

Parameters

- **v_wind** (*pandas.Series or array*) – Wind speed time series.
- **v_wind_height** (*float*) – Height for which the parameter *v_wind* applies.
- **hub_height** (*float*) – Hub height of wind turbine.
- **z_0** (*pandas.Series or array or float*) – Roughness length.
- **obstacle_height** (*float*) – Height of obstacles in the surrounding area of the wind turbine. Set *obstacle_height* to zero for wide spread obstacles. Default: 0.

Returns Wind speed at hub height.

Return type pandas.Series or array

Notes

The following equation is used[1]:

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

with: v: wind speed, h: height, z_0: roughness length, d: boundary layer offset (estimated by d = 0.7 * *obstacle_height*)

For d = 0 it results in the following equation[2],[3]:

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

h_{data} is the height at which the wind speed $v_{wind,data}$ is measured.

Parameters *v_wind_height*, *z_0*, *hub_height* and *obstacle_height* have to be of the same unit.

References

windpowerlib.wind_speed.v_wind_hellman

windpowerlib.wind_speed.**v_wind_hellman**(*v_wind*, *v_wind_height*, *hub_height*, *hellman_exp=None*, *z_0=None*)

Calculates the wind speed at hub height using the hellman equation.

It is assumed that the wind profile follows a power law. This fuction is carried out when the parameter *wind_model* of an instance of the `Modelchain` class is 'hellman'.

Parameters

- **v_wind** (*pandas.Series or array*) – Wind speed time series.
- **v_wind_height** (*float*) – Height for which the parameter *v_wind* applies.
- **hub_height** (*float*) – Hub height of wind turbine.
- **hellman_exp** (*float*) – The Hellman exponent, which combines the increase in wind speed due to stability of atmospheric conditions and surface roughness into one constant. Default: 1/7. If roughness length is given *hellman_exp* = 1 / ln(h_hub/z_0).
- **z_0** (*float*) – Roughness length. Default: None.

Returns Wind speed at hub height.

Return type pandas.Series or array

[1] Quaschning V.: "Regenerative Energiesysteme". München, Hanser Verlag, 2011, p. 278
[2] Gasch, R., Twele, J.: "Windkraftanlagen". 6. Auflage, Wiesbaden, Vieweg + Teubner, 2010, p. 129
[3] Hau, E.: "Windkraftanlagen - Grundlagen, Technik, Einsatz, Wirtschaftlichkeit". 4. Auflage, Springer-Verlag, 2008, p. 515

Notes

The following equation is used[1],[2],[3]:

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

with: v: wind speed, h: height, α: Hellman exponent

h_{data} is the height in which the wind speed $v_{wind,data}$ is measured and h_{hub} is the hub height of the wind turbine.

For the Hellman exponent α many studies use a value of 1/7 for onshore and a value of 1/9 for offshore. The Hellman exponent can also be calulated by the following equation[2],[3]:

$$\alpha = \frac{1}{ln\left(\frac{h_{hub}}{z_0}\right)}$$

with: z_0: roughness length

References

[1] Sharp, E.: "Spatiotemporal disaggregation of GB scenarios depicting increased wind capacity and electrified heat demand in dwellings". UCL, Energy Institute, 2015, p. 83

[2] Hau, E.: "Windkraftanlagen - Grundlagen, Technik, Einsatz, Wirtschaftlichkeit". 4. Auflage, Springer-Verlag, 2008, p. 517

[3] Quaschning V.: "Regenerative Energiesysteme". München, Hanser Verlag, 2011, p. 279

CHAPTER 6

Wind turbine data

Functions and methods to obtain the nominal power as well as power curve or power coefficient curve needed by the *WindTurbine* class.

wind_turbine.WindTurbine.fetch_turbine_data()	Fetches data of the requested wind turbine.
wind_turbine.get_turbine_types([print_out])	Get the names of all possible wind turbine types for which the power coefficient curve or power curve is provided in the data files in the directory windpowerlib/data.
wind_turbine.read_turbine_data(**kwargs)	Fetches power coefficient curve or power curve from a file.

windpowerlib.wind_turbine.WindTurbine.fetch_turbine_data

WindTurbine.**fetch_turbine_data**()

Fetches data of the requested wind turbine.

Method fetches nominal power as well as power coefficient curve or power curve from a data file provided along with the windpowerlib.

Returns Nominal power of the requested wind turbine.

Return type float

Examples

```
>>> from windpowerlib import wind_turbine
>>> enerconE126 = {
...     'hub_height': 135,
...     'd_rotor': 127,
...     'turbine_name': 'ENERCON E 126 7500'}
>>> e126 = wind_turbine.WindTurbine(**enerconE126)
>>> print(e126.cp_values.cp[5.0])
```

```
0.423
>>> print(e126.nominal_power)
7500000.0
```

windpowerlib.wind_turbine.get_turbine_types

windpowerlib.wind_turbine.**get_turbine_types**(*print_out=True*, ***kwargs*)

Get the names of all possible wind turbine types for which the power coefficient curve or power curve is provided in the data files in the directory windpowerlib/data.

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

Examples

```
>>> from windpowerlib import wind_turbine
>>> valid_types_df = wind_turbine.get_turbine_types(print_out=False)
>>> print(valid_types_df.iloc[5])
turbine_id    DEWIND D8 2000
p_nom                   2000
Name: 5, dtype: object
```

windpowerlib.wind_turbine.read_turbine_data

windpowerlib.wind_turbine.**read_turbine_data**(***kwargs*)

Fetches power coefficient curve or power curve from a file.

The data files are provided along with the windpowerlib and are located in the directory windpowerlib/data.

Other Parameters

- **datapath** (*string, optional*) – Path where the data file is stored. Default: './data'
- **filename** (*string, optional*) – Name of data file. Default: 'cp_curves.csv'

Returns Power coefficient curve values or power curve values with the corresponding wind speeds as indices.

Return type pandas.DataFrame

CHAPTER 7

Power output

Functions for calculating power output of a wind turbine.

power_output.cp_curve(v_wind, rho_hub, ...)	Calculates the turbine power output using a cp curve.
power_output.cp_curve_density_corr(v_wind, ...)	Calculates the turbine power output using a density corrected cp curve.
power_output.p_curve(p_values, v_wind)	Calculates the turbine power output using a power curve.
power_output.p_curve_density_corr(v_wind, ...)	Calculates the turbine power output using a density corrected power curve.

windpowerlib.power_output.cp_curve

windpowerlib.power_output.**cp_curve**(*v_wind*, *rho_hub*, *d_rotor*, *cp_values*)

Calculates the turbine power output using a cp curve.

This function is carried out when the parameter *power_output_model* of an instance of the *Modelchain* class is 'cp_values' and the parameter *density_corr* is False.

Parameters

- **v_wind** (*pandas.Series or array*) – Wind speed at hub height in m/s.
- **rho_hub** (*pandas.Series or array*) – Density of air at hub height in kg/m^3.
- **d_rotor** (*float*) – Diameter of rotor in m.
- **cp_values** (*pandas.DataFrame*) – Power coefficient curve of the wind turbine. Indices are the wind speeds of the power coefficient curve in m/s, the corresponding power coefficient values are in the column 'cp'.

Returns Electrical power output of the wind turbine in W.

Return type pandas.Series

Notes

The following equation is used[1],[2]:

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

with: P: power [W], ρ: density [kg/m^3], d: diameter [m], v: wind speed [m/s], cp: power coefficient

It is assumed that the power output for wind speeds above the maximum wind speed given in the power coefficient curve is zero.

References

windpowerlib.power_output.cp_curve_density_corr

windpowerlib.power_output.**cp_curve_density_corr**(*v_wind*, *rho_hub*, *d_rotor*, *cp_values*)

Calculates the turbine power output using a density corrected cp curve.

This function is carried out when the parameter *power_output_model* of an instance of the *Modelchain* class is 'cp_values' and the parameter *density_corr* is True.

Parameters

- **v_wind** (*pandas.Series or array*) – Wind speed at hub height in m/s.
- **rho_hub** (*pandas.Series or array*) – Density of air at hub height in kg/m^3.
- **cp_values** (*pandas.DataFrame*) – Power coefficient curve of the wind turbine. Indices are the wind speeds of the power coefficient curve in m/s, the corresponding power coefficient values are in the column 'cp'.
- **d_rotor** (*float*) – Diameter of the rotor in m.

Returns Electrical power of the wind turbine in W.

Return type pandas.Series

Notes

See *cp_curve()* for further information on how the power values are calculated and *p_curve_density_corr()* for further information on how the density correction is implemented.

It is assumed that the power output for wind speeds above the maximum wind speed given in the power coefficient curve is zero.

windpowerlib.power_output.p_curve

windpowerlib.power_output.**p_curve**(*p_values*, *v_wind*)

Calculates the turbine power output using a power curve.

This function is carried out when the parameter *power_output_model* of an instance of the *Modelchain* class is 'p_values' and the parameter *density_corr* is False.

[1] Gasch, R., Twele, J.: "Windkraftanlagen". 6. Auflage, Wiesbaden, Vieweg + Teubner, 2010, pages 35ff, 208
[2] Hau, E.: "Windkraftanlagen - Grundlagen, Technik, Einsatz, Wirtschaftlichkeit". 4. Auflage, Springer-Verlag, 2008, p. 542

Parameters

- **p_values** (*pandas.DataFrame*) – Power curve of the wind turbine. Indices are the wind speeds of the power curve in m/s, the corresponding power values in W are in the column 'P'.
- **v_wind** (*pandas.Series or array*) – Wind speed at hub height in m/s.

Returns **power_output** – Electrical power output of the wind turbine in W.

Return type pandas.Series

Notes

It is assumed that the power output for wind speeds above the maximum wind speed given in the power curve is zero.

windpowerlib.power_output.p_curve_density_corr

windpowerlib.power_output.**p_curve_density_corr**(*v_wind*, *rho_hub*, *p_values*)

Calculates the turbine power output using a density corrected power curve.

This function is carried out when the parameter *power_output_model* of an instance of the *Modelchain* class is 'p_values' and the parameter *density_corr* is True.

Parameters

- **v_wind** (*pandas.Series or array*) – Wind speed time series at hub height in m/s.
- **rho_hub** (*pandas.Series or array*) – Density of air at hub height in kg/m^3.
- **p_values** (*pandas.DataFrame*) – Power curve of the wind turbine. The indices are the corresponding wind speeds of the power curve, the power values containing column is called 'P'.

Returns **power_output** – Electrical power output of the wind turbine in W.

Return type pandas.Series

Notes

The following equation is used for the wind speed at site[1],[2],[3]:

$$v_{site} = v_{std} \cdot \left(\frac{\rho_0}{\rho_{site}} \right)^{p(v)}$$

with:

$$p = \begin{cases} \frac{1}{3} & v_{std} \leq 7.5 \text{ m/s} \\ \frac{1}{15} \cdot v_{std} - \frac{1}{6} & 7.5 \text{ m/s} < v_{std} < 12.5 \text{ m/s} \\ \frac{2}{3} & \geq 12.5 \text{ m/s} \end{cases},$$

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

[1] Svenningsen, L.: "Power Curve Air Density Correction And Other Power Curve Options in WindPRO". 1st edition, Aalborg, EMD International A/S , 2010, p. 4

[2] Svenningsen, L.: "Proposal of an Improved Power Curve Correction". EMD International A/S , 2010

[3] Biank, M.: "Methodology, Implementation and Validation of a Variable Scale Simulation Model for Windpower based on the Georeferenced Installation Register of Germany". Master's Thesis at Reiner Lemoine Institute, 2014, p. 13

v_{std} is the standard wind speed in the power curve (v_{std}, P_{std}). v_{site} is the density corrected wind speed for the power curve (v_{site}, P_{std}).

It is assumed that the power output for wind speeds above the maximum wind speed given in the power curve is zero.

References

CHAPTER 8

Modelchain

Creating a Modelchain object.

modelchain.Modelchain(wind_turbine[, ...])	Model to determine the output of a wind turbine

Running the modelchain.

modelchain.Modelchain.run_model(weather, ...)	Runs the model.

windpowerlib.modelchain.Modelchain.run_model

Modelchain.**run_model**(*weather*, *data_height*)

Runs the model.

Parameters

- **weather** (*DataFrame or Dictionary*) – Containing columns or keys with the timeseries for wind speed *v_wind* in m/s ,roughness length *z0* in m, temperature *temp_air* in K and pressure *pressure* in Pa, as well as optionally wind speed *v_wind_2* in m/s and temperature *temp_air_2* in K at different height for interpolation.
- **data_height** (*DataFrame or Dictionary*) – Containing columns or keys with the heights in m for which the corresponding parameters in *weather* apply.

Returns

Return type self

Methods of the Modelchain object.

modelchain.Modelchain.rho_hub(weather, ...)	Calculates the density of air at hub height.

Continued on next page

Table 8.3 – continued from previous page

modelchain.Modelchain.v_wind_hub(weather, ...)	Calculates the wind speed at hub height.
modelchain.Modelchain.turbine_power_output(...)	Calculates the power output of the wind turbine.

windpowerlib.modelchain.Modelchain.rho_hub

Modelchain.**rho_hub**(*weather*, *data_height*)

Calculates the density of air at hub height.

The density is calculated using the method specified by the parameter *rho_model*. Previous to the calculation of density the temperature at hub height is calculated using the method specified by the parameter *temperature_model*.

Parameters

- **weather** (*DataFrame or Dictionary*) – Containing columns or keys with time-series for temperature *temp_air* in K and pressure *pressure* in Pa, as well as optionally the temperature *temp_air_2* in K at a different height for interpolation.
- **data_height** (*DataFrame or Dictionary*) – Containing columns or keys with the heights in m for which the corresponding parameters in *weather* apply.

Returns **rho_hub** – Density of air in kg/m^3 at hub height.

Return type pandas.Series or array

windpowerlib.modelchain.Modelchain.v_wind_hub

Modelchain.**v_wind_hub**(*weather*, *data_height*)

Calculates the wind speed at hub height.

The method specified by the parameter *wind_model* is used.

Parameters

- **weather** (*DataFrame or Dictionary*) – Containing columns or keys with the timeseries for wind speed *v_wind* in m/s and roughness length *z0* in m, as well as optionally wind speed *v_wind_2* in m/s at different height for interpolation.
- **data_height** (*DataFrame or Dictionary*) – Containing columns or keys with the heights in m for which the corresponding parameters in *weather* apply.

Returns **v_wind** – Wind speed in m/s at hub height.

Return type pandas.Series or array

windpowerlib.modelchain.Modelchain.turbine_power_output

Modelchain.**turbine_power_output**(*v_wind*, *rho_hub*)

Calculates the power output of the wind turbine.

The method specified by the parameter *power_output_model* is used.

Parameters

- **v_wind** (*pandas.Series or array*) – Wind speed at hub height in m/s.
- **rho_hub** (*pandas.Series or array*) – Density of air at hub height in kg/m^3.

Returns **output** – Electrical power in W of the wind turbine.

Return type pandas.Series

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Index

Symbols

C

D

F

G

H

L

M

N

O

P

R

T

V

W