hrosailing-Module Documentation

Dependencies

The hrosailing-module has the following third-party dependencies

- numpy
- matplotlib
- pynmea2
- scipy

How To Use This Module

After installing/downloading one can easily use the hrosailing-module via

```
>>> import hrosailing
```

or

```
>>> from hrosailing import ...
```

Contents Of This Module

The hrosailing-module defines the following public functions:

hrosailing.**apparent_wind_to_true**(wind_arr)

hrosailing.**true_wind_to_apparent**(wind_arr)

The hrosailing-module has the following public submodules:

- hrosailing.polardiagram
- hrosailing.processing

The hrosailing.polardiagram-module defines the following public functions:

polardiagram.**to_csv**(csv_path, obj)

Calls the **.to_csv**-method of the hrosailing.**PolarDiagram** instance.

Parameters :

csv_path : string

Path where a .csv-file is located or where a new .csv-file will be created

obj : PolarDiagram

polardiagram.**PolarDiagram** instance which will be written to the .csv-file

Raises a FileWritingException if the file can't be written to

polardiagram.**from_csv**(csv_path, fmt='hro', tw=True)

Reads a .csv file and returns the polardiagram.**PolarDiagram** instance contained in it

Parameters :

`csv_path` : `string`

Path to a .csv file which will be read

`fmt` : `string`

The "format" of the .csv file. Currently supported formats are:

'hro': format created by the `polardiagram`.**to_csv** function
'orc': format found at ORC
'opencpn': format created by the OpenCPN Polar Plugin
'array':

`tw` : `bool`

Specifies if wind data in file should be viewed as true wind

Defaults to `True`

Raises a FileReadingException if

- an unknown format was specified

- the file can't be found, opened or read

`polardiagram`.**pickling**(`pkl_path, obj`)

Calls the **.pickling**-method of the `polardiagram`.**PolarDiagram** instance

Parameters :

`pkl_path` : `string`

Path where a .pkl file is located or where a new .pkl file will be created

`obj` : `PolarDiagram`

`polardiagram`.**PolarDiagram** instance which will be written to the .csv-file

Raises a FileWritingException if the file can't be written to

`polardiagram`.**depickling**(`pkl_path`)

Reads a .pkl file and returns the `polardiagram`.**PolarDiagram** instance contained in it

Parameters :

`pkl_path` : `string`

Path to a .pkl file which will be read

Raises a FileReadingException if file can't be found, opened, or read

`polardiagram`.**symmetric_polar_diagram**(obj)

Symmetrizeses an `polardiagram`.**PolarDiagram** instance, meaning for every datapoint with wind speed, wind angle and boat speed (w, phi, s)
a new data point with wind speed, wind angle and boat speed (w, 360 - phi, s)
will be added

Parameters :

`obj` : `PolarDiagram`

`polardiagram`.**PolarDiagram** instance which will be symmetrized

Returns :

`symmetric` : `PolarDiagram`

"symmetrized" version of `obj`

Raises a PolarDiagramException if `obj` is not of type `PolarDiagramTable` or `PolarDiagramPointcloud`

The `polardiagram`-module defines the following public classes:

`polardiagram`.**PolarDiagram**()

An abstract base class for the `polardiagram` classes

Methods :

`PolarDiagram`.**pickling**(`self, pkl_path`)

Writes `self` to a .pkl file

Parameters :

`pkl_path` : `string`

Path where a .pkl file is located or where a new .pkl file will be created

Raises a FileWritingException if the file can't be written to

`PolarDiagram`.**plot_polar_slice**(`self, ws, ax=None, **plot_kw`)

Creates a polar plot of a given slice of the polar diagram

Parameters :

`ws` : `int` or `float`

Slice of the polar diagram, given as either

- an element of self.wind_speeds for PolarDiagramTable

  Slice then equals the corresponding column of self.boat_speeds together with the wind angles in self.wind_angles

  Same with PolarDiagramMultiSails

- as a single wind speed for PolarDiagramCurve

  Slice then equals self(ws, wa), where wa will go through a fixed number of angles between 0° and 360°

- a single wind speed for PolarDiagramPointcloud

  Slice then consists of all rows of self.points with the first entry being equal to ws

`ax` : `matplotlib.projections.polar.PolarAxes`, optional

Axes instance where the plot will be created

If nothing is passed, the function will create a suitable axes

`plot_kw` : Keyword arguments

Keyword arguments that will be passed to the matplotlib.axes.Axes.plot function, to change certain appearences of the plot

Raises a PolarDiagramException if

- ws is not in self.wind_speed for PolarDiagramTable and PolarDiagramMultiSails

- there are no rows in self.points with first entry ws for PolarDiagramPointcloud

`PolarDiagram.`**plot_flat_slice**(`self, ws, ax=None, **plot_kw`)

Creates a cartesian plot of a given slice of the polar diagram

Parameters :

`ws` : `int` or `float`

Slice of the polar diagram, given as either

- an element of self.wind_speeds for PolarDiagramTable

Slice then equals the corresponding column of self.boat_speeds together with the wind angles in self.wind_angles

Same with PolarDiagramMultiSails

- as a single wind speed for PolarDiagramCurve

Slice then equals self(ws, wa), where wa will go through a fixed number of angles between 0° and 360°

- a single wind speed for PolarDiagramPointcloud

Slice then consists of all rows of self.points with the first entry being equal to ws

`ax` : `matplotlib.axes.Axes`, optional

Axes instance where the plot will be created

If nothing is passed, the function will create a suitable axes

`plot_kw` : Keyword arguments

Keyword arguments that will be passed to the matplotlib.axes.Axes.plot function, to change certain appearences of the plot

Raises a PolarDiagramException if

- ws is not in self.wind_speed for PolarDiagramTable and PolarDiagramMultiSails

- there are no rows in self.points with first entry ws for PolarDiagramPointcloud

`PolarDiagram.`**plot_convex_hull_slice**(`self, ws, ax=None, **plot_kw`)

Computes the convex hull of a given slice of the polar diagram
and creates a polar plot of it

Parameters :

`ws` : `int` or `float`

Slice of the polar diagram, given as either

- an element of self.wind_speeds for
PolarDiagramTable

Slice then equals the corresponding
column of self.boat_speeds together
with the wind angles in self.wind_angles

Same with PolarDiagramMultiSails

- as a single wind speed for PolarDiagramCurve

Slice then equals self(ws, wa), where wa will
go through a fixed number of angles between
0° and 360°

- a single wind speed for PolarDiagramPointcloud

Slice then consists of all rows of self.points
with the first entry being equal to ws

`ax` : `matplotlib.projections.polar.PolarAxes`, optional

Axes instance where the plot will be created

If nothing is passed, the function will create
a suitable axes

`plot_kw` : Keyword arguments

Keyword arguments that will be passed to the
matplotlib.axes.Axes.plot function, to change
certain appearences of the plot

Raises a PolarDiagramException if

- ws is not in self.wind_speed for PolarDiagramTable
and PolarDiagramMultiSails

- there are no rows in self.points with first entry ws
for PolarDiagramPointcloud

Abstract Methods :

`PolarDiagram.`**to_csv**`(self, csv_path)`

`PolarDiagram.`**plot_polar**`(self, ws, ax=None,`

`colors=('green', 'red'), show_legend=False, legend_kw=None, **plot_kw)`

`PolarDiagram.`**plot_flat**`(self, ws, ax=None,`

`colors=('green', 'red'), show_legend=False, legend_kw=None, **plot_kw)`

`PolarDiagram.`**plot_3d**`(self, ax=None, **plot_kw)`

`PolarDiagram.`**plot_color_gradient**`(self, ax=None,` `colors=('green', 'red'), marker=None, show_legend=False, legend_kw=None`)

`PolarDiagram`.**plot_convex_hull**()

`polardiagram`.**PolarDiagramTable**(`ws_res=None, wa_res=None, bsps=None, tw=True`)

A class to represent, visualize and work with a polar diagram in form of a table

Parameters :

`ws_res` : `Iterable` or `int` or `float`, optional

Wind speeds that will correspond to the columns of the table

Can either be a sequence of length cdim or a number

If a number num is passed, `numpy.arange(num, 40, num)` will be assigned to ws_res

If nothing is passed, it will default to `numpy.arange(2, 42, 2)`

`wa_res` : `Iterable` or `int` or `float`, optional

Wind angles that will correspond to the columns of the table

Can either be sequence of length rdim or a number

If a number num is passed, `numpy.arange(num, 360, num)` will be assigned to wa_res

If nothing is passed, it will default to `numpy.arange(0, 360, 5)`

`bsps` : `array_like`, optional

Sequence of corresponding boat speeds, should be broadcastable to the shape (rdim, cdim)

If nothing is passed it will default to `numpy.zeros((rdim, cdim))`

`tw` : `bool`, optional

Specifies if the given wind data should be viewed as true wind

If `False`, wind data will be converted to true wind

Defaults to `True`

Raises an exception if `bsps` can't be broadcasted to shape (rdim, cdim)

Methods :

`PolarDiagramTable`.**wind_speeds**

Returns a read only version of `self.`*_resolution_wind_speed*

`PolarDiagramTable`.**wind_angles**

Returns a read only version of `self.`*_resolution_wind_angle*

`PolarDiagramTable`.**boat_speeds**

Returns a read only version of `self.`*_bsps*

`PolarDiagramTable`.**to_csv**(`self, csv_path, fmt='hro'`)

Creates a .csv file with delimiter ',' and the following format:

PolarDiagramTable
Wind speed resolution:
`self`.**wind_speeds**
Wind angle resolution:
`self`.**wind_angles**
Boat speeds:
`self`.**boat_speeds**

Parameters :

`csv_path` : `string`

Path where a .csv file is located or where a new .csv file will be created

`fmt` : `string`

Specifies the format of the created csv

Raises an exception if the file can't be written to

`PolarDiagramTable`.**change_entries**(`self`,`new_bsps, ws=None,` wa=None``)

Changes specified entries in the table

Parameters :

`new_bsps` : `array_like`

Sequence containing the new data to be inserted in the specified entries

`ws` : `Iterable`, or `int` or `float`, optional

Element(s) of `self`.**wind_speeds**, specifying the columns, where `new_bsps` will be inserted

If nothing is passed it will default to `self`.**wind_speeds**

`wa` : `Iterable`, or `int` or `float`, optional

Element(s) of `self`.**wind_angles**, specifiying the rows, where `new_bsps` will be inserted

If nothing is passed it will default to `self`.**wind_angles**

Raises an exception if

`ws` is not contained in `self`.**wind_speeds**
`wa` is not contained in `self`.**wind_angles**
`new_bsps` can't be broadcasted to a fitting shape

`PolarDiagramTable`.**plot_polar** (`self, ws=None, ax=None, colors=('green', 'red'), show_legend=False, legend_kw=None, **plot_kw`)

Creates a polar plot of multiple slices (columns) of the polar diagram

Parameters :

`ws` : `Iterable`, optional

Slices (columns) of the polar diagram table, given as an Iterable of elements of `self`.**wind_speeds**

If nothing it passed, it will default to `self`.**wind_speeds**

`ax` : `matplotlib.projections.polar.PolarAxes`, optional

Axes instance where the plot will be created.

If nothing is passed, the function will create a suitable axes

`colors` : `tuple`, optional

Specifies the colors to be used for the different slices

Accepts all colors and representations as given in colors and repr

There are four options for the tuple

If as many or more colors as slices are passed, each slice will be plotted in the specified color

Otherwise if exactly 2 colors are passed, the slices will be plotted with a color gradient consiting of the given two colors

If more than 2 colors are passed, either the first n_color slices will be plotted in the specified colors, and the rest will be plotted in the default color 'blue', or one can specify certain slices to be plotted in a certain color by passing a tuple of (ws, color) pairs

Defaults to the tuple ('green', 'red')

`show_legend` : `bool`, optional

Specifies wether or not a legend will be shown next to the plot

The type of legend depends on the color options:

If the slices are plotted with a color gradient, a `matplotlib.colorbar.Colorbar` object will be created and assigned to ax

Otherwise a `matplotlib.legend.Legend` object will be created and assigned to ax

Default to `False`

`legend_kw` : `dict`, optional

Keyword arguments to be passed to either the `matplotlib.colorbar.Colorbar` class or the `matplotlib.legend.Legend` class to change position and appearence of the legend

Will only be used if 'show_legend=True'

If noting is passed, it will default to `{}`

`plot_kw` : Keyword arguments

Keyword arguments that will be passed to the `matplotlib.axes.Axes.plot` function, to change certain appearences of the plot

Raises an exception if at least one element of `ws_range` is not in `self`.**wind_speeds**

`PolarDiagramTable`.**plot_flat** (`self, ws=None, ax=None, colors=('green', 'red'), show_legend=False, legend_kw=None, **plot_kw`)

Creates a cartesian plot of multiple slices (columns) of the polar diagram

Parameters :

`ws` : `Iterable`, optional

Slices (columns) of the polar diagram table, given as an Iterable of elements of `self`.**wind_speeds**

If nothing it passed, it will default to `self`.**wind_speeds**

`ax` : `matplotlib.axes.Axes`, optional

Axes instance where the plot will be created.

If nothing is passed, the function will create a suitable axes

`colors` : `tuple`, optional

Specifies the colors to be used for the different slices

Accepts all colors and representations as given in colors and repr

There are four options for the tuple

If as many or more colors as slices are passed, each slice will be plotted in the specified color

Otherwise if exactly 2 colors are passed, the slices will be plotted with a color gradient consiting of the given two colors

If more than 2 colors are passed, either the first n_color slices will be plotted in the specified colors, and the rest will be plotted in the default color 'blue', or one can specify certain slices to be plotted in a certain color by passing a tuple of (ws, color) pairs

Defaults to the tuple ('green', 'red')

`show_legend` : `bool`, optional

Specifies wether or not a legend will be shown
next to the plot

The type of legend depends on the color options:

If the slices are plotted with a color gradient,
a `matplotlib.colorbar.Colorbar` object
will be created and assigned to `ax`

Otherwise a `matplotlib.legend.Legend` object
will be created and assigned to `ax`

Default to `False`

`legend_kw` : `dict`, optional

Keyword arguments to be passed to either the
`matplotlib.colorbar.Colorbar` class or the
`matplotlib.legend.Legend` class to change
position and appearence of the legend

Will only be used if 'show_legend=True'

If noting is passed, it will default to `{}`

`plot_kw` : Keyword arguments

Keyword arguments that will be passed to the
`matplotlib.axes.Axes.plot` function, to change
certain appearences of the plot

Raises an exception if at least one element of `ws_range` is not in
`self`.**wind_speeds**

`PolarDiagramTable.`**plot_3d**`(self, ax=None, colors=('blue', 'blue'))`

Creates a 3d plot of the polar diagram

Parameters :

`ax`: `mpl_toolkits.mplot3d.axes3d.Axes3D`, optional

Axes instance where the plot will be created

If nothing is passed, the function will create
a suitable axes

`colors` : `tuple` of length 2, optional

Colors which specify the color gradient with
which the polar diagram will be plotted

Accepts all colors and representations as given in colors and repr

If no color gradient is desired, set both elements
to the same color

Defaults to ('blue', 'blue')

`PolarDiagramTable.`**plot_color_gradient**`(self, ax=None,`

`colors=('green', 'red'), marker=None, show_legend=False, *legend_kw)`

Creates a 'wind speed vs. wind angle' color gradient plot of the polar diagram with respect to the respective boat speeds

Parameters :

`ax` : `matplotlib.axes.Axes`, optional

Axes instance where the plot will be created.

If nothing is passed, the function will create a suitable axes

`colors` : `tuple` of length 2, optional

Colors which specify the color gradient with which the polar diagram will be plotted

Accepts all colors and representations as given in colors and repr

Defaults to ('green', 'red')

`marker` : `matplotlib.markers.Markerstyleor` equivalent, optional

Markerstyle for the created scatter plot

If nothing is passed, it will default to 'o'

`show_legend` : `bool`, optional

Specifies wether or not a legend will be shown next to the plot

Legend will be a `matplotlib.colorbar.Colorbar` object.

Defaults to `False`

`legend_kw` : Keyword arguments

Keyword arguments to be passed to the `matplotlib.colorbar.Colorbar` class to change position and appearence of the legend

Will only be used if 'show_legend=True'

`PolarDiagramTable`.**plot_convex_hull**()

`polar_diagram`.**PolarDiagramCurve**(`f, params, radians=False`)

A class to represent, visualize and work with a polar diagram given by a fitted curve/surface

Parameters :

`f` : `function`

Curve/surface that describes the polar diagram, given as a function, with the signature `f(x, *params) -> y`, where x is a `numpy.ndarray` of shape (n, 2) which corresponds to pairs of wind speed and wind angle and y is a `numpy.ndarray` of shape (n,) or (n, 1) which corresponds to the boat speed at the resp. wind speed and wind angle.

`params` : `tuple` or Sequence

Optimal parameters for f

`radians` : `bool`, optional

Specifies if f takes the wind angles to be in radians or degrees

Defaults to `False`

Methods :

`PolarDiagramCurve.`**curve**

Returns a read only version of `self.`*_f*

`PolarDiagramCurve.`**radians**

Returns a read only version of `self.`*_radians*

`PolarDiagramCurve.`**parameters**

Returns a read only version of `self.`*_params*

`PolarDiagramCurve.`**to_csv**(`self, csv_path`)

Creates a .csv file with delimiter ':' and the following format:

PolarDiagramCurve
Function: `self.`**curve**.__name__
Radians: `self.`**radians**
Parameters: `self.`**parameters**

Parameters :

`csv_path` : `string`

Path where a .csv file is located or where a new
.csv file will be created

Raises an exception if the file can't be written to

`PolarDiagramCurve.`**plot_polar**(`self, ws=(0, 20, 5), ax=None, colors=('green', 'red'), show_legend=False, legend_kw=None, **plot_kw`)

Creates a polar plot of multiple slices of the polar diagram

Parameters :

`ws` : `tuple` of length 3 or `list`, optional

Slices of the polar diagram given either as a
tuple of three values, which will be interpreted
as a start and end point of an interval aswell as
a number of slices, which will be evenly spaced
in the given interval, or as a list of specific
wind speed values

Defaults to (0, 20, 5)

`ax` : `matplotlib.projections.polar.PolarAxes`, optional

Axes instance where the plot will be created.

If nothing is passed, the function will create
a suitable axes

`colors` : `tuple`, optional

Specifies the colors to be used for the different
slices

Accepts all colors and representations as given in colors and repr

There are four options for the tuple

> If as many or more colors as slices are passed, each slice will be plotted in the specified color
>
> Otherwise if exactly 2 colors are passed, the slices will be plotted with a color gradient consiting of the given two colors
>
> If more than 2 colors are passed, either the first n_color slices will be plotted in the specified colors, and the rest will be plotted in the default color 'blue', or one can specify certain slices to be plotted in a certain color by passing a tuple of (ws, color) pairs
>
> Defaults to the tuple ('green', 'red')

`show_legend` : `bool`, optional

Specifies wether or not a legend will be shown next to the plot

The type of legend depends on the color options:

> If the slices are plotted with a color gradient, a `matplotlib.colorbar.Colorbar` object will be created and assigned to `ax`
>
> Otherwise a `matplotlib.legend.Legend` object will be created and assigned to `ax`

Default to `False`

`legend_kw` : `dict`, optional

Keyword arguments to be passed to either the `matplotlib.colorbar.Colorbar` class or the `matplotlib.legend.Legend` class to change position and appearence of the legend

Will only be used if 'show_legend=True'

If noting is passed, it will default to `{}`

`plot_kw` : Keyword arguments

Keyword arguments that will be passed to the `matplotlib.axes.Axes.plot` function, to change certain appearences of the plot

`PolarDiagramCurve.`**flat_plot**`(self, ws=(0, 20, 5), ax=None, colors=('green', 'red'), show_legend=False, legend_kw=None, **plot_kw)`

Creates a cartesian plot of multiple slices of the polar diagram

Parameters :

`ws` : `tuple` of length 3 or `list`, optional

> Slices of the polar diagram given either as a
> tuple of three values, which will be interpreted
> as a start and end point of an interval aswell as
> a number of slices, which will be evenly spaced
> in the given interval, or as a list of specific
> wind speed values
>
> Defaults to (0, 20, 5)

`ax` : `matplotlib.axes.Axes`, optional

> Axes instance where the plot will be created.
>
> If nothing is passed, the function will create
> a suitable axes

`colors` : `tuple`, optional

> Specifies the colors to be used for the different
> slices
>
> Accepts all colors and representations as given
> in colors and repr
>
> There are four options for the tuple
>
> > If as many or more colors as slices are passed,
> > each slice will be plotted in the specified
> > color
> >
> > Otherwise if exactly 2 colors are passed, the
> > slices will be plotted with a color gradient
> > consiting of the given two colors
> >
> > If more than 2 colors are passed, either
> > the first n_color slices will be plotted
> > in the specified colors, and the rest will
> > be plotted in the default color 'blue',
> > or one can specify certain slices to be
> > plotted in a certain color by passing a
> > tuple of (ws, color) pairs
> >
> > Defaults to the tuple ('green', 'red')

`show_legend` : `bool`, optional

> Specifies wether or not a legend will be shown
> next to the plot
>
> The type of legend depends on the color options:
>
> > If the slices are plotted with a color gradient,
> > a `matplotlib.colorbar.Colorbar` object
> > will be created and assigned to `ax`
> >
> > Otherwise a `matplotlib.legend.Legend` object
> > will be created and assigned to `ax`
>
> Default to `False`

`legend_kw` : `dict`, optional

> Keyword arguments to be passed to either the
> `matplotlib.colorbar.Colorbar` class or the
> `matplotlib.legend.Legend` class to change
> position and appearence of the legend
>
> Will only be used if 'show_legend=True'
>
> If noting is passed, it will default to `{}`

`plot_kw` : Keyword arguments

> Keyword arguments that will be passed to the
> `matplotlib.axes.Axes.plot` function, to change
> certain appearences of the plot

`PolarDiagramCurve.`**plot_3d**`(self, ws=(0, 20, 100), ax=None, colors=('blue', 'blue'))`

Creates a 3d plot of a part of the polar diagram

Parameters :

`ws_range` : `tuple` of length 3, optional

> A region of the polar diagram given as a tuple
> of three values, which will be interpreted as
> a start and an end point of an interval aswell as
> a number of samples in this interval. The more
> samples there are, the "smoother" the resulting
> plot will be
>
> Defaults to (0, 20, 100)

`ax`: `mpl_toolkits.mplot3d.axes3d.Axes3D`, optional

> Axes instance where the plot will be created.
>
> If nothing is passed, the function will create
> a suitable axes

`colors` : `tuple` of length 2, optional

> Colors which specify the color gradient with
> which the polar diagram will be plotted
>
> Accepts all colors and representations as given in colors and repr
>
> If no color gradient is desired, set both elements
> to the same color
>
> Defaults to ('blue', 'blue')

`PolarDiagramCurve.`**plot_color_gradient**`(self, ws=(0, 20, 100), ax=None, colors=('green', 'red'), marker=None, show_legend=False, **legend_kw)`

Creates a 'wind speed vs. wind angle' color gradient plot of a part
of the polar diagram with respect to the respective boat speeds

Parameters :

`ws_range` : `tuple` of length 3, optional

A region of the polar diagram given as a tuple
of three values, which will be interpreted as
a start and an end point of an interval aswell as
a number of samples in this interval.

Defaults to (0, 20, 100)

`ax` : `matplotlib.axes.Axes`, optinal

Axes instance where the plot will be created.

If nothing is passed, the function will create
a suitable axes

`colors` : `tuple` of length 2, optional

Colors which specify the color gradient with
which the polar diagram will be plotted

Accepts all colors and representations as given in colors and repr

Defaults to ('green', 'red')

`marker` : `matplotlib.markers.Markerstyleor` equivalent, optional

Markerstyle for the created scatter plot

If nothing is passed, it will default to 'o'

`show_legend` : `bool`, optional

Specifies wether or not a legend will be shown
next to the plot

Legend will be a `matplotlib.colorbar.Colorbar`
object.

Defaults to `False`

`legend_kw` : Keyword arguments

Keyword arguments to be passed to the
`matplotlib.colorbar.Colorbar` class to change
position and appearence of the legend

Will only be used if 'show_legend=True'

`PolarDiagramCurve.`**plot_convex_hull**()

`polar_diagram.`**PolarDiagramPointcloud**(`pts=None, tw=True`)

A class to represent, visualize and work with a polar diagram given by a point cloud

Parameters :

`pts` : `array_like`, optional

Initial points of the point cloud, given as a sequence of points
consisting of wind speed, wind angle and boat speed

If nothing is passed, point cloud will be initialized
with an empty array

`tw` : `bool`, optional

Specifies if the given wind data should be viewed as true wind

If `False`, wind data will be converted to true wind

Defaults to `True`

Raises an exception if `pts` can't be broadcasted to shape (n, 3)

Methods :

`PolarDiagramPointcloud.`**wind_speeds**

Returns a list of all the different wind speeds in the point cloud

`PolarDiagramPointcloud.`**wind_angles**

Returns a list of all the different wind angles in the point cloud

`PolarDiagramPointcloud.`**points**

Returns a read only version of `self.`*_pts*

`PolarDiagramPointcloud.`**to_csv**(`self, csv_path`)

Creates a .csv file with delimiter ',' and the following format

PolarDiagramPointcloud
True wind speed ,True wind angle ,Boat speed
`self.`**points**

Parameters :

`csv_path` : `string`

Path where a .csv file is located or where a new .csv file will be created

Raises an exception if the file can't be written to

`PolarDiagramPointcloud.`**add_points**(`self, new_pts, tw=True`)

Adds additional points to the point cloud

Parameters :

`new_points` : `array_like`

New points to be added to the point cloud given as a sequence of points consisting of wind speed, wind angle and boat speed

`tw` : `bool`, optional

Specifies if the given wind data should be viewed as true wind

If `False`, wind data will be converted to true wind

Defaults to `True`

Raises an exception if

`new_pts` can't be broadcasted to shape (n, 3)
`new_pts` is an empty array

`PolarDiagramPointcloud.`**plot_polar**(`self, ws=(0, numpy.inf), ax=None, colors=('green', 'red'), show_legend=False, legend_kw=None, **plot_kw`)

Creates a polar plot of multiple slices of the polar diagram

Parameters :

`ws` : `tuple` of length 2 or `list`, optional

Slices of the polar diagram given as either a tuple of two values which will be interpreted as a lower and an upper bound of the wind speed, such that all slices that correspond a wind speed that fits within these bounds will be plotted, or as a list of specific slices given as the values of the corresponding wind speed

Defaults to (0, np.inf)

`ax` : `matplotlib.projections.polar.PolarAxes`, optional

Axes instance where the plot will be created.

If nothing is passed, the function will create a suitable axes

`colors` : `tuple`, optional

Specifies the colors to be used for the different slices

Accepts all colors and representations as given in colors and repr

There are four options for the tuple

If as many or more colors as slices are passed, each slice will be plotted in the specified color

Otherwise if exactly 2 colors are passed, the slices will be plotted with a color gradient consiting of the given two colors

If more than 2 colors are passed, either the first n_color slices will be plotted in the specified colors, and the rest will be plotted in the default color 'blue', or one can specify certain slices to be plotted in a certain color by passing a tuple of (ws, color) pairs

Defaults to the tuple ('green', 'red')

`show_legend` : `bool`, optional

Specifies wether or not a legend will be shown next to the plot

The type of legend depends on the color options:

If the slices are plotted with a color gradient, a `matplotlib.colorbar.Colorbar` object will be created and assigned to `ax`

Otherwise a `matplotlib.legend.Legend` object will be created and assigned to `ax`

Default to `False`

`legend_kw` : `dict`, optional

Keyword arguments to be passed to either the `matplotlib.colorbar.Colorbar` class or the `matplotlib.legend.Legend` class to change position and appearence of the legend

Will only be used if 'show_legend=True'

If noting is passed, it will default to `{}`

`plot_kw` : Keyword arguments

Keyword arguments that will be passed to the `matplotlib.axes.Axes.plot` function, to change certain appearences of the plot

Raises an exception in the case that `ws_range` is a list and there is a wind_speed `ws` in `ws_range` such that there are no rows in `self`.**pts** with first entry `ws`

`PolarDiagramPointcloud.`**plot_flat**`(self, ws=(0, numpy.inf), ax=None, colors=('green', 'red'), show_legend=False, legend_kw=None, **plot_kw)`

Parameters :

`ws` : `tuple` of length 2 or `list`, optional

Slices of the polar diagram given as either a tuple
of two values which will be interpreted as
a lower and an upper bound of the wind speed,
such that all slices that correspond a wind speed
that fits within these bounds will be plotted,
or as a list of specific slices given as the values
of the corresponding wind speed

Defaults to (0, np.inf)

`ax` : `matplotlib.axes.Axes`, optional

Keyword arguments that will be passed to the `matplotlib.axes.Axes.plot` function, to change certain appearences of the plot

`colors` : `tuple`, optional

Specifies the colors to be used for the different slices

Accepts all colors and representations as given in colors and repr

There are four options for the tuple

If as many or more colors as slices are passed, each slice will be plotted in the specified color

Otherwise if exactly 2 colors are passed, the slices will be plotted with a color gradient consiting of the given two colors

If more than 2 colors are passed, either the first n_color slices will be plotted in the specified colors, and the rest will be plotted in the default color 'blue', or one can specify certain slices to be plotted in a certain color by passing a tuple of (ws, color) pairs

Defaults to the tuple ('green', 'red')

`show_legend` : `bool`, optional

Specifies wether or not a legend will be shown next to the plot

The type of legend depends on the color options:

If the slices are plotted with a color gradient, a `matplotlib.colorbar.Colorbar` object will be created and assigned to `ax`

Otherwise a `matplotlib.legend.Legend` object will be created and assigned to `ax`

Default to `False`

`legend_kw` : `dict`, optional

Keyword arguments to be passed to either the `matplotlib.colorbar.Colorbar` class or the `matplotlib.legend.Legend` class to change position and appearence of the legend

Will only be used if 'show_legend=True'

If noting is passed, it will default to `{}`

`plot_kw` : Keyword arguments

Keyword arguments that will be passed to the `matplotlib.axes.Axes.plot` function, to change certain appearences of the plot

Raises an exception in the case that `ws_range` is a list and there is a wind_speed `ws` in `ws_range` such that there are no rows in `self`.**pts** with first entry `ws`

`PolarDiagramPointcloud.` **plot_3d**(`self, ax=None, **plot_kw`)

Creates a 3d plot of the polar diagram

Parameters :

`ax` : `mpl_toolkits.mplot3d.axes3d.Axes3D`, optional

Keyword arguments that will be passed to the `matplotlib.axes.Axes.plot` function, to change certain appearences of the plot

`plot_kw` : Keyword arguments

Keyword arguments that will be passed to the `matplotlib.axes.Axes.plot` function, to change certain appearences of the plot

`PolarDiagramPointcloud.`**plot_color_gradient**(`self, ax=None,`

`colors=('green', 'red'), marker=None, show_legend=False, **legend_kw`):

Creates a 'wind speed vs. wind angle' color gradient plot of the polar diagram with respect to the respective boat speeds

Parameters :

`ax` : `matplotlib.axes.Axes`, optional

Keyword arguments that will be passed to the `matplotlib.axes.Axes.plot` function, to change certain appearences of the plot

`colors` : `tuple` of length 2, optional

Colors which specify the color gradient with which the polar diagram will be plotted

Accepts all colors and representations as given in colors and repr

Defaults to ('green', 'red')

`marker` : `matplotlib.markers.Markerstyleor` equivalent, optional

Markerstyle for the created scatter plot

If nothing is passed, it will default to 'o'

`show_legend` : `bool`, optional

Specifies wether or not a legend will be shown next to the plot

Legend will be a `matplotlib.colorbar.Colorbar` object.

Defaults to `False`

`legend_kw` : Keyword arguments

Keyword arguments to be passed to the `matplotlib.colorbar.Colorbar` class to change position and appearence of the legend

Will only be used if 'show_legend=True'

`PolarDiagramPointcloud`**plot_convex_hull**()