WindenergyTK API

Alec Koumjian

April 18, 2010

Contents

Contents **1**

1 Package windenergytk **3**
1.1 Modules . . . 3
1.2 Variables . . . 3

2 Module windenergytk.aerodyn **4**
2.1 Functions . . . 4
2.2 Variables . . . 6

3 Module windenergytk.analysis **7**
3.1 Functions . . . 7
3.2 Variables . . . 7

4 Module windenergytk.dynamics **8**
4.1 Variables . . . 8

5 Module windenergytk.electrical **9**
5.1 Functions . . . 9
5.2 Variables . . . 9

6 Module windenergytk.file_ops **10**
6.1 Functions . . . 10
6.2 Variables . . . 10

7 Module windenergytk.gwindtk **11**
7.1 Variables . . . 11
7.2 Class MyFrame . . . 11
7.2.1 Methods . . . 11
7.2.2 Properties . . . 15

8 Module windenergytk.mechanics **16**
8.1 Functions . . . 16
8.2 Variables . . . 17

9 Module windenergytk.performance **18**
9.1 Functions . . . 18
9.2 Variables . . . 18

10 Module windenergytk.synthesis **19**
10.1 Functions 19
10.2 Variables 19

1 Package windenergytk

1.1 Modules

- **aerodyn** *(Section 2, p. 4)*
- **analysis** *(Section 3, p. 7)*
- **dynamics** *(Section 4, p. 8)*
- **electrical** *(Section 5, p. 9)*
- **file_ops** *(Section 6, p. 10)*
- **gwindtk** *(Section 7, p. 11)*
- **mechanics** *(Section 8, p. 16)*
- **performance** *(Section 9, p. 18)*
- **synthesis** *(Section 10, p. 19)*

1.2 Variables

Name	Description
__package__	**Value:** `None`

2 Module windenergytk.aerodyn

2.1 Functions

q_terms(*local_pitch*, *local_tip_loss*, *lift_coef_slope*, *lift_coef_intercept*, *local_solidity*)

Create the q terms used in simplified angle of attack calculation.

See Manwell, et. al Section 3.11 p. 138-39 Please note that q1 and q3 are switched in Book vs. code INPUT local_pitch: (float) local_tsr: (floar) lift_coef_slope: (float) lift_coef_intercept: (float) local_solidity: (float)

OUTPUT These terms are used in linear approximation of alpha calculation q1: (float) q2: (float) q3: (float)

calc_attack_angle(*q1*, *q2*, *q3*)

Calculate angle of attack for linear/small angle approximation.

As seen in section 3.11 of Manwell, et. al.

INPUT Q-terms as returned from q_terms(). q1: (float) q2: (float) q3: (float)

OUTPUT angle_of_attack: (float) local angle of attack in radians

calc_axial_factor(*local_tip_loss*, *lift_coefficient*, *angle_of_rwind*, *local_solidity*)

calc_angular_factor(*axial_induc_factor*, *angle_of_rwind*, *local_tsr*)

tip_loss(*number_of_blades*, *fractional_radius*, *angle_of_rwind*)

This calculates the rotor tip loss using Prandtl method.

Reference: deVries, Fluid Dynamics Aspects of Wind Energy Conversion.

INPUT number_of_blades: (int) fractional_radius: (float) local radius / total radius angle_of_rwind: (float) angle of relative wind

OUTPUT tip_loss: (float)

rotor_coefs(*axial_induc_factor*, *angular_induc_factor*, *angle_of_rwind*, *tip_speed_ratio*, *local_tsr*, *num_stations*, *local_solidity*, *lift_coefficient*, *drag_coefficient*, *local_tip_loss*)

Calculate local thrust, torque, and power coefficients.

INPUT (all floating point numbers)

OUTPUT local_thrust_coef: (float) local_torque_coef: (float) local_power_coef: (float

linear_method_factors(*fradius*, *number_blades*, *local_pitch*, *local_tsr*, *lift_coef_slope*, *lift_coef_intercept*, *drag_coef_slope*, *drag_coef_intercept*, *local_solidity*)

Get angle of attack, relative wind, induction factors using linear curve.

INPUT fradius: (float) fractional radius of station number_blades: (int) number of blades local_pitch: (float) local pitch in radians local_tsr: (float) local tip speed ratio lift_coef_slope: (float) slope of linear lift coef vs. angle of attack lift_coef_intercept:(float) intercept of linear lift coef vs. AoA curve drag_coef_slope: (float) same as lift drag_coef_intercept:(float) same as lift local_solidity: (float) local solidity

OUTPUT local_tip_loss: (float) angle_of_attack: (float) angle_of_rwind: (float) lift_coefficient: (float) drag_coefficient: (float) axial_induc_factor: (float) angular_induc_factor: (float

nonlinear_method_factors(*fradius*, *number_blades*, *local_pitch*, *local_tsr*, *lift_curve*, *drag_curve*, *local_solidity*)

Get angle of attack, relative wind, induction factors w/ nonlinear curve.

INPUT fradius: (float) fractional radius of station number_blades: (int) number of blades local_pitch: (float) local pitch in radians local_tsr: (float) local tip speed ratio lift_curve: (float) array of empirical lift_coef vs. AoA curve drag_curve: (float) array of empirical drag_coef vs. AoA curve local_solidity: (float) local solidity

OUTPUT local_tip_loss: (float) angle_of_attack: (float) angle_of_rwind: (float) lift_coefficient: (float) drag_coefficient: (float) axial_induc_factor: (float) angular_induc_factor: (float)

optimum_rotor(*lift_coefficient*, *angle_of_attack*, *tip_speed_ratio*, *total_radius*, *hub_radius*, *number_blades*, *sections*)

```
Return blade station, chord, and twist for a given turbine.

INPUT
lift_coefficient: (float) airfoil lift coefficient at intended angle attack
angle_of_attack: (float) angle of attack in degrees
total_radius: (float) outer radius of turbine blades in meters
hub_radius: (float) radius of hub, where blades begin
number_blades: (int) number of turbine blades
sections: (int) number of sections to divide blade length into

OUTPUT
rotor_design: (numpy.ndarray) 3 x sections array with station, chord, twist
    station: (float) distance from hub in meters
    chord: (float)
    twist: (float)
```

rotor_analysis(*rct_matrix*, *tip_speed_ratio*, *number_blades*, *pitch_0*, *blade_radius*, *hub_radius*, *lift_curve*, *drag_curve*, *method*)

```
Returns performance statistics of a rotor.

INPUT
tip_speed_ratio: (float) The tip speed ratio
number_blades:   (int) the number of blades
pitch_0 :        (float) initial pitch angle relative to tip, deg
blade_radius:          (float) radius in meters
hub_radius:      (float) hub radius in meters
lift_curve:       (array-like) either linear slope and intercept or
                               emperical C_l vs. AoA points
drag_curve:       (array-like) either linear slope and intercept or
                               emperical C_d vs. C_l points



rct_matrix: (numpy.ndarray) 3 x n array of fradius, chord, twist on each line
    fradius: (float) nondimensional fractional radius along blade
    chord:   (float) nondimensional length
    twist:   (float) in degrees

OUTPUT
rotor_stats: (ndarray) 7 x n, of the following
    angle_of_attack: (float) estimated angle of attack in degrees
    angle_of_rwind: (float) estimated angle of relative wind in degrees
    lift_coef: (float) linear approximation of lift coefficient
    drag_coef: (float) linear approximation of drag coefficient
    axial_induc_factor: (float)
    angular_induc_factor: (float)
    tip_loss_factor: (float)
    local_power_coef: (float) local power coefficient
```

2.2 Variables

Name	Description
__package__	**Value:** 'windenergytk'

3 Module windenergytk.analysis

3.1 Functions

get_statistics(*timeseries*, *output*=`'dictionary'`)

Collects statistics from a timeseries object.

Input: A timeseries object, output format ('dictionary' or 'list') Output: Dictionary or array of mean, stdev, max, min, size

get_histogram_data(*timeseries*, *bins*=`10`, *normalized*=`True`)

Returns histogram data of timeseries object Input: timeseries obj. Optional: No. of bins (default is 10), normalized (boolean) Output: Tuple (histogram, bin_edges)

get_weibull_params(*mean*, *stdev*)

Returns Weibull parameters c and k. Input: mean, stdev Output: Weibull scale: c, shape: k Citation: Manwell 2000, chapter 2

crosscorrelate(*timeseries1*, *timeseries2*, *max_lag_increment*=`False`)

Returns crosscorrelation values at lag increments. Input: timeseries1, timeseries2 Optional: max_lag_increment (int) Output: array of lags, array of correlation values

autocorrelate(*timeseries*, *max_lag_increment*=`False`)

Returns autocorrelation values at lag increments. Input: a single timeseries object optional: max_lag_increment Output: Array of lags, Array of normalized autocorrelation values.

block_average(*timeseries*, *new_freq*=`''`)

Reduce size of timeseries by taking averages of larger block size. Input: timeseries, new_freq (str) See scikits.timeseries doc Output: block averaged timeseries obj. in new frequency

power_spectral_density(*data_array*, *frequency*, *segment_size*=`256`, *window_method*=`False`)

Return the power spectral density using matplotlib.pyplot.psd function.

3.2 Variables

Name	Description
__package__	**Value:** `'windenergytk'`

4 Module windenergytk.dynamics

4.1 Variables

Name	Description
__package__	**Value:** `None`

5 Module windenergytk.electrical

5.1 Functions

complex_arithmetic()

This procedure can be used to perform complex arithmetic as is useful for analysis of AC power

induction_gen_model()

This procedure is used to analyze an induction generator/motor.

synchronous_gen_model()

This procedure is used to analyze a round rotor synchronous generator.

5.2 Variables

Name	Description
__package__	**Value:** `None`

6 Module windenergytk.file_ops

6.1 Functions

sanitize(*a_string*)

Sanite string from leading/trailing whitespaces, make lowercase, etc.

parse_file(*dat_file*)

Return meta and timeseries objects from WEC dat file.

Input: opened WEC data file Output: Dictionary of timeseries with meta-data attached

parse_meta(*meta_array*)

Parse list of lines from .dat file into meaningful key/value pairs. Input: An array of lines from a WEC .dat file Output: Dictionary meta information

separate_timeseries(*timeseries*)

Take a multi column timeseries and separate into single timeseries.

assign_meta(*ts_dict*, *meta_dict*)

Assign meta information to recently separated timeseries. Input: ts_dict from separate_timeseries(), meta_dict from parse_meta() Output: Completed dictionary of timeseries with meta information

6.2 Variables

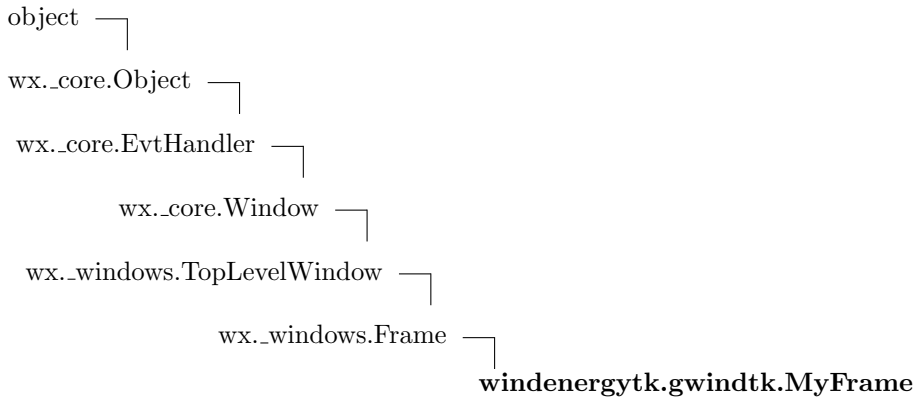
Name	Description
__package__	**Value:** `'windenergytk'`

7 Module windenergytk.gwindtk

7.1 Variables

Name	Description
__package__	**Value:** `'windenergytk'`

7.2 Class MyFrame



7.2.1 Methods

__init__(*self*, **args*, ***kwds*)

```
__init__(self, Window parent, int id=-1, String title=EmptyString,
    Point pos=DefaultPosition, Size size=DefaultSize,
    long style=DEFAULT_FRAME_STYLE, String name=FrameNameStr) -> Frame
```

Return Value
EvtHandler

Overrides: object.__init__ extit(inherited documentation)

valid_selections(*self*, *allowed_number*)

Checks number of selections in self.list_box_1 Input: allowed_number (int) is number of allowed selections Output: MsgDialog if incorrect number, else passes returns True

renumber_active_timeseries(*self*)

Renumber timeseries in active_timeseries dictionary.

sync_active_listbox(*self*)

Synchronize listbox to active_timeseries values.

refresh_timeseries(*self*)

Renumber and sync active ts to listbox.

add_timeseries(*self*, *timeseries_dict*)

Add a timeseries dictionary (metadata + ts) to active_timeseries

remove_timeseries(*self*, *index*)

Remove timeseries from active list

create_ts_dict(*self*, *new_ts*, *old_ts_dict*=`False`, *prepend_str*=`''`)

Place new timeseries inside meta_ts_dict, copy meta if applicable. Input: new timeseries Optional: old timeseries for meta_data, prepend string Output: new meta + ts dictionary

OnImport(*self*, *event*)

Import all timeseries from a *.dat file

OnHelpIndex(*self*, *event*)

OnAboutBox(*self*, *event*)

OnPlotTSButton(*self*, *event*)

Plot a timeseries object using matplotlib.

OnRemoveTSButton(*self*, *event*)

Remove selected timeseries from the active list and listbox.

OnTogglePanelButton(*self*, *event*)

Hide or show a panel when toggle button is pressed.

OnStatButton(*self*, *event*)

Generate statistics and print to panel.

OnHistButton(*self*, *event*)

Generate histogram and plot using matplotlib.

OnWeibullButton(*self*, *event*)

Retrieve Weibull parameters from selected timeseries.

OnCorrButton(*self*, *event*)

Generate and plot correlation values.

OnAutocorrButton(*self*, *event*)

Generate and plot autocorrelation values.

OnCrosscorrButton(*self*, *event*)

Generate and plot crosscorrelation vaules.

OnBlockButton(*self*, *event*)

Create block average of selected timeseries and add to list_box_1

OnPSDButton(*self*, *event*)

Generate power spectral density info and plot using matplotlib.

OnARMAButton(*self*, *event*)

Generate timeseries using autoregressive moving average method. Input: mean, stdev, npoints, autocorr (all from gui inputs) Output: Add generated timeseries to self.list_box_1

Inherited from wx._windows.Frame

Command(), Create(), CreateStatusBar(), CreateToolBar(), DoGiveHelp(), DoMenuUpdates(), GetClassDefaultAttributes(), GetMenuBar(), GetStatusBar(), GetStatusBarPane(), GetToolBar(), PopStatusText(), ProcessCommand(), PushStatusText(), SendSizeEvent(), SetMenuBar(), SetStatusBar(), SetStatusBarPane(), SetStatusText(), SetStatusWidths(), SetToolBar()

Inherited from wx._windows.TopLevelWindow

CenterOnScreen(), CentreOnScreen(), EnableCloseButton(), GetDefaultItem(), GetIcon(), GetTitle(), GetTmpDefaultItem(), Iconize(), IsActive(), IsAlwaysMaximized(), IsFullScreen(), IsIconized(), IsMaximized(), MacGetMetalAppearance(), MacGetTopLevelWindowRef(), MacGetUnifiedAppearance(), MacSetMetalAppearance(), Maximize(), RequestUserAttention(), Restore(), SetDefaultItem(), SetIcon(), SetIcons(), SetShape(), SetTitle(), SetTmpDefaultItem(), ShowFullScreen(), __repr__()

Inherited from wx._core.Window

AcceptsFocus(), AcceptsFocusFromKeyboard(), AddChild(), AdjustForLayoutDirection(), AssociateHandle(), CacheBestSize(), CanSetTransparent(), CaptureMouse(), Center(), CenterOnParent(), Centre(), CentreOnParent(), ClearBackground(), ClientToScreen(), ClientToScreenXY(), ClientToWindowSize(), Close(), ConvertDialogPointToPixels(), ConvertDialogSizeToPixels(), ConvertPixelPointToDialog(), ConvertPixelSizeToDialog(), DLG_PNT(), DLG_SZE(), Destroy(), DestroyChildren(), Disable(), DissociateHandle(), DragAcceptFiles(), Enable(), FindFocus(), FindWindowById(), FindWindowByLabel(), FindWindowByName(), Fit(), FitInside(), Freeze(), GetAcceleratorTable(), GetAdjustedBestSize(), GetAutoLayout(), GetBackgroundColour(), GetBackgroundStyle(), GetBestFittingSize(), GetBestSize(),

GetBestSizeTuple(), GetBestVirtualSize(), GetBorder(), GetCapture(), GetCaret(), GetCharHeight(), GetCharWidth(), GetChildren(), GetClientAreaOrigin(), GetClientRect(), GetClientSize(), GetClientSizeTuple(), GetConstraints(), GetContainingSizer(), GetCursor(), GetDefaultAttributes(), GetDropTarget(), GetEffectiveMinSize(), GetEventHandler(), GetExtraStyle(), GetFont(), GetForegroundColour(), GetFullTextExtent(), GetGrandParent(), GetGtkWidget(), GetHandle(), GetHelpText(), GetHelpTextAtPoint(), GetId(), GetLabel(), GetLayoutDirection(), GetMaxHeight(), GetMaxSize(), GetMaxWidth(), GetMinHeight(), GetMinSize(), GetMinWidth(), GetName(), GetParent(), GetPosition(), GetPositionTuple(), GetRect(), GetScreenPosition(), GetScreenPositionTuple(), GetScreenRect(), GetScrollPos(), GetScrollRange(), GetScrollThumb(), GetSize(), GetSizeTuple(), GetSizer(), GetTextExtent(), GetThemeEnabled(), GetToolTip(), GetTopLevelParent(), GetUpdateClientRect(), GetUpdateRegion(), GetValidator(), GetVirtualSize(), GetVirtualSizeTuple(), GetWindowBorderSize(), GetWindowStyle(), GetWindowStyleFlag(), GetWindowVariant(), HasCapture(), HasFlag(), HasMultiplePages(), HasScrollbar(), HasTransparentBackground(), Hide(), HitTest(), HitTestXY(), InheritAttributes(), InheritsBackgroundColour(), InitDialog(), InvalidateBestSize(), IsBeingDeleted(), IsDoubleBuffered(), IsEnabled(), IsExposed(), IsExposedPoint(), IsExposedRect(), IsFrozen(), IsRetained(), IsShown(), IsShownOnScreen(), IsTopLevel(), Layout(), LineDown(), LineUp(), Lower(), MakeModal(), Move(), MoveAfterInTabOrder(), MoveBeforeInTabOrder(), MoveXY(), Navigate(), NewControlId(), NextControlId(), PageDown(), PageUp(), PopEventHandler(), PopupMenu(), PopupMenuXY(), PostCreate(), PrepareDC(), PrevControlId(), PushEventHandler(), Raise(), Refresh(), RefreshRect(), RegisterHotKey(), ReleaseMouse(), RemoveChild(), RemoveEventHandler(), Reparent(), ScreenToClient(), ScreenToClientXY(), ScrollLines(), ScrollPages(), ScrollWindow(), SetAcceleratorTable(), SetAutoLayout(), SetBackgroundColour(), SetBackgroundStyle(), SetBestFittingSize(), SetCaret(), SetClientRect(), SetClientSize(), SetClientSizeWH(), SetConstraints(), SetContainingSizer(), SetCursor(), SetDimensions(), SetDoubleBuffered(), SetDropTarget(), SetEventHandler(), SetExtraStyle(), SetFocus(), SetFocusFromKbd(), SetFont(), SetForegroundColour(), SetHelpText(), SetHelpTextForId(), SetId(), SetInitialSize(), SetLabel(), SetLayoutDirection(), SetMaxSize(), SetMinSize(), SetName(), SetOwnBackgroundColour(), SetOwnFont(), SetOwnForegroundColour(), SetPosition(), SetRect(), SetScrollPos(), SetScrollbar(), SetSize(), SetSizeHints(), SetSizeHintsSz(), SetSizeWH(), SetSizer(), SetSizerAndFit(), SetThemeEnabled(), SetToolTip(), SetToolTipString(), SetTransparent(), SetValidator(), SetVirtualSize(), SetVirtualSizeHints(), SetVirtualSizeHintsSz(), SetVirtualSizeWH(), SetWindowStyle(), SetWindowStyleFlag(), SetWindowVariant(), ShouldInheritColours(), Show(), Thaw(), ToggleWindowStyle(), TransferDataFromWindow(), TransferDataToWindow(), UnregisterHotKey(), Update(), UpdateWindowUI(), UseBgCol(), Validate(), WarpPointer(), WindowToClientSize()

Inherited from wx._core.EvtHandler

AddPendingEvent(), Bind(), Connect(), Disconnect(), GetEvtHandlerEnabled(),

GetNextHandler(), GetPreviousHandler(), ProcessEvent(), ProcessPendingEvents(), SetEvtHandlerEnabled(), SetNextHandler(), SetPreviousHandler(), Unbind()

Inherited from wx._core.Object

GetClassName(), IsSameAs()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

7.2.2 Properties

Name	Description
Inherited from wx._windows.Frame MenuBar, StatusBar, StatusBarPane, ToolBar, thisown	
Inherited from wx._windows.TopLevelWindow DefaultItem, Icon, Title, TmpDefaultItem	
Inherited from wx._core.Window AcceleratorTable, AutoLayout, BackgroundColour, BackgroundStyle, BestSize, BestVirtualSize, Border, Caret, CharHeight, CharWidth, Children, ClientAreaOrigin, ClientRect, ClientSize, Constraints, ContainingSizer, Cursor, DefaultAttributes, DropTarget, EffectiveMinSize, Enabled, EventHandler, ExtraStyle, Font, ForegroundColour, GrandParent, GtkWidget, Handle, HelpText, Id, Label, LayoutDirection, MaxHeight, MaxSize, MaxWidth, MinHeight, MinSize, MinWidth, Name, Parent, Position, Rect, ScreenPosition, ScreenRect, Shown, Size, Sizer, ThemeEnabled, ToolTip, TopLevel, TopLevelParent, UpdateClientRect, UpdateRegion, Validator, VirtualSize, WindowStyle, WindowStyleFlag, WindowVariant	
Inherited from wx._core.EvtHandler EvtHandlerEnabled, NextHandler, PreviousHandler	
Inherited from wx._core.Object ClassName	
Inherited from object __class__	

8 Module windenergytk.mechanics

8.1 Functions

euler_beam_vibrations(*beam_length*, *area_moment*, *mass_per_length*, *elastic_modulus*, *mode*)

Estimate the natural freq of uniform cantilevered beam.

INPUT beam_length: (float) length of beam area_moment: (float) area moment of inertia for the beam mass_per_length: (float) the length density, mass per unit length elastic_modulus: (float) stress / strain mode: (int): The number mode to find a natural frequency for

OUTPUT natural_frequency: (float) frequency for input mode beta: (float) parameter calculated from beta_l

For reference see: Manwell Chapter 4 p. 153
http://en.wikipedia.org/wiki/Euler–Bernoulli_beam_equation

myklestad_beam_vibrations(*sec_lengths*, *sec_masses*, *e_i*, *density*, *rot_velocity*, *freq_start*, *freq_final*, *freq_step*)

Estimate the natural freq of a nonuniform vibrating cantilevered beam.

INPUT sec_lengths: (array-like) length of each section starting at free end sec_masses: (array-like) mass of each section e_i: (array-like) structural stiffness, young's modulus * area inertia density: (float) density of material rot_velocity: (float) rotational velocity in rad/s freq_start: (float) starting low guess for natural frequency, rad/s freq_final: (float) high guess fo rnatural frequency, rad/s freq_step: (float) frequency step

OUTPUT nat_frequencies: (array-like) list of natural frequencies

hinge_spring_flapping(*num_blades*, *blade_radius*, *blade_chord*, *blade_mass*, *lift_curve_slope*, *blade_pitch_angle*, *rot_nat_freq*, *non_nat_freq*, *yaw_to_blade*, *yaw_rate*, *cross_flow*, *linear_shear*, *air_density*, *rot_velocity*, *tip_speed_ratio*)

Calculate the terms of blade flap displacement based on azimuth angle.

INPUT num_blades: (int) number of blades blade_radius: (float) radius of blades in meters blade_chord: (float) blade average chord blade_mass: (float) mass of blade in kg lift_curve_slope: (float) slope of lift curve; d c_l/d alpha blade_pitch_angle: (float) in radians rot_nat_freq: (float) blade rotating natural flapping frequency non_nat_freq: (float) blade nonrotating natural flapping frequency yaw_to_blade: (float) distance from yaw axis to blade in meters yaw_rate: (float) yaw rate, rad/s cross_flow: (float) cross flow velocity in m/s linear_shear: (float) linear wind shear coefficient air_density: (float) in kg/m**3 rot_velocity: (float) rotor speed in rad/s tip_speed_ratio: (float)

OUTPUT beta_0: (float) collective flapping angle, degrees beta_1c: (float) cosine flapping term, vertical tilting term, degrees beta_1s: (float) sine flapping term, yaw or lateral tilting term, degrees

holzer_natural_freq(*number_of_nodes*, *list_of_inertias*, *list_shaft_stiffness*, *start_freq*, *ending_freq*, *freq_step*)

Holzer method to find the natural frequency of a rotating system.

rainflow_cycle_counting(*tseries*)

Perform a cycle counting analysis of timeseries using rainflow method.

8.2 Variables

Name	Description
__package__	**Value:** `'windenergytk'`

9 Module windenergytk.performance

9.1 Functions

power_curve_estimation()

average_power_output()

life_cycle_economics()

wind_diesel_system()

battery_discharge_capacity()

noise_estimation()

9.2 Variables

Name	Description
__package__	**Value:** `None`

10 Module windenergytk.synthesis

10.1 Functions

find_bin(*some_number*, *min*, *bins*, *value_range*)

Find the bin (index) that a number falls into.

weighted_choice(*cumu_prob_vector*)

Returns random number (0 < x < 1) weighted by the probability vector.

gen_arma(*mean*, *stdev*, *autocor1*, *npoints*)

Normally distributed timeseries using Autoregressive Moving Average.

gen_markov_tpm(*tseries*, *bins*)

Generate a Markov transition probability matrix from a timeseries.

gen_cumu_tpm(*tpm*)

Create cumulative transition probability matrix from regular tpm.

gen_ts_from_tpm(*tpm*, *bin_width*, *length*, *freq*='T')

```
Create timeseries using a Transisiton Probability Matrix
INPUT: tpm  = ndarray of n*n values
       length (int)
OUTPUT: tseries = timeseries of length
```

add_diurnal(*tseries*, *sine_period*, *peak_mag*)

Scales a time series to a sine wave of peak_mag with sine_period. Input: tseries, sine_period (float, hrs), peak_mag (float) Output: scaled_data (array-like)

gen_pdf(*desired_mean*, *desired_stdev*, *bin_width*)

10.2 Variables

Name	Description
__package__	**Value:** 'windenergytk'