functions Namespace Reference

Functions

def **derivative** (t, x)
Derivation of a signal. More...

def **integration** (t, x)
Integration of a signal. More...

def **myhighpass** (t, u, tc)
First order high pass filter. More...

def **mylowpass** (t, u, tc)
First order low pass filter. More...

def **mw_dft** (data, t, omega)
Function required for calculating windowed phasor of a signal. More...

def **window_phasor_mag** (t, x, sr, cycles, dom_freq=50)
Magnitude of moving discrete fourier transform. More...

def **window_phasor_angle** (t, x, sr, cycles, dom_freq=50)
Angle of moving discrete fourier transform. More...

def **trendfilter** (t, x, lamda1)
Returns a smoothened version of the input signal, depending on parameter 'lambda', considered as a median filter, also known as 'Hodrick Prescott' filter. More...

def **avgMovWin** (t, v, t_win)
Moving window average of the signal. More...

def **rmsMovWin** (t, v, t_win)
Moving window RMS of the signal. More...

def **clarkestranform** (t, va, vb, vc)
Transforms 3 phase to alpha, beta, zero components. More...

def **inv_clarkestransform** (t, va, vb, vc)
Inverse of Clarke's transform, converts alpha, beta, zero component to a,b,c component. More...

def **parkstransform** (t, va, vb, vc, w, gamma)
Transforms 3 phase to D,Q,0 components. More...

def **inv_parkstransform** (t, va, vb, vc, w, gamma)
Inverse of Park's transform, transforms D,Q,0 phase to a,b,c components. More...

def **sequencetransform** (t, va, vb, vc)

def **instaLL_RMSVoltage** (t, va, vb, vc)
Instantaneous line to line RMS voltage. More...

def **insta_RMSCurrent** (t, ia, ib, ic)
Instantaneous line current. More...

Function Documentation

◆ avgMovWin()

```
def functions.avgMovWin (  t,
                           v,
                           t_win
                        )
```

Moving window average of the signal.

Parameters

t	Time array
v	Signal array
t_win	Window length in seconds

Returns

Array representing the average of signal using a moving window.

◆ clarkestranform()

```
def functions.clarkestranform (  t,
                                 va,
                                 vb,
                                 vc
                              )
```

Transforms 3 phase to alpha, beta, zero components.

Parameters

t	Time array
va	a phase voltage array
vb	b phase voltage array
vc	c phase voltage array

Returns

3 arrays corresponding to alpha, beta, zero component.

◆ derivative()

```
def functions.derivative ( t,
                           x
                         )
```

Derivation of a signal.

Parameters

- **t** Time array
- **x** Signal array

Returns

Array containing the derivative of the signal 'x'.

◆ insta_RMSCurrent()

```
def functions.insta_RMSCurrent ( t,
                                 ia,
                                 ib,
                                 ic
                               )
```

Instantaneous line current.

Parameters

- **t** Time array
- **ia** a phase current
- **ib** b phase current
- **ic** c phase current

Returns

Array corresponding to the instantaneous line current.

◆ instaLL_RMSVoltage()

```
def functions.instaLL_RMSVoltage (   t,
                                     va,
                                     vb,
                                     vc
                                 )
```

Instantaneous line to line RMS voltage.

Parameters

- **t** Time array
- **va** a phase voltage
- **vb** b phase voltage
- **vc** c phase voltage

Returns

Array corresponding to the instantaneous RMS voltage

◆ integration()

```
def functions.integration (   t,
                              x
                          )
```

Integration of a signal.

Parameters

- **t** Time array
- **x** Signal array

Returns

Array containing the integration of the signal 'x'.

◆ inv_clarkestransform()

```
def functions.inv_clarkestransform (   t,
                                       va,
                                       vb,
                                       vc
                                   )
```

Inverse of Clarke's transform, converts alpha, beta, zero component to a,b,c component.

Parameters

t	Time array
va	alpha component array
vb	beta component array
vc	zero component array

Returns

3 arrays corresponding to a,b,c component.

◆ inv_parkstransform()

```
def functions.inv_parkstransform (   t,
                                     va,
                                     vb,
                                     vc,
                                     w,
                                     gamma
                                 )
```

Inverse of Park's transform, transforms D,Q,0 phase to a,b,c components.

Parameters

t	Time array
va	D component array
vb	Q component array
vc	0 component array
w	Frequency (in Hertz)
gamma	Phase angle (in Radian)

Returns

3 arrays corresponding to a,b,c component respectively.

◆ mw_dft()

```
def functions.mw_dft (  data,
                        t,
                        omega
                     )
```

Function required for calculating windowed phasor of a signal.

Parameters

data	Signal array
t	Time array
omega	Frequency

Returns

A complex number representing phasor quantity.

◆ myhighpass()

```
def functions.myhighpass (  t,
                            u,
                            tc
                         )
```

First order high pass filter.

Parameters

t	Time array
u	Signal array
tc	Cut-off frequency for the filter

Returns

Array showing the high-pass output of signal 'x' based on cutoff frequency 'tc'.

◆ mylowpass()

```
def functions.mylowpass (   t,
                            u,
                            tc
                        )
```

First order low pass filter.

Parameters

- **t** Time array
- **u** Signal array
- **tc** Cut-off frequency for the filter

Returns

Array showing the low-pass output of signal 'x' based on cutoff frequency 'tc'.

◆ parkstransform()

```
def functions.parkstransform (   t,
                                 va,
                                 vb,
                                 vc,
                                 w,
                                 gamma
                             )
```

Transforms 3 phase to D,Q,0 components.

Parameters

- **t** Time array
- **va** a component array
- **vb** b component array
- **vc** c component array
- **w** Frequency (in Hertz)
- **gamma** Phase angle (in Radian)

Returns

3 arrays corresponding to D,Q,0 component respectively.

◆ rmsMovWin()

```
def functions.rmsMovWin (  t,
                           v,
                           t_win
                        )
```

Moving window RMS of the signal.

Parameters

- **t** Time array
- **v** Signal array
- **t_win** Window length in seconds

Returns

Array representing the RMS of signal using a moving window.

◆ sequencetransform()

```
def functions.sequencetransform (  t,
                                   va,
                                   vb,
                                   vc
                                )
```

Parameters

- **t** Time array
- **va** a component array
- **vb** b component array
- **vc** c component array :return:

◆ trendfilter()

```
def functions.trendfilter (  t,
                             x,
                             lamda1
                          )
```

Returns a smoothened version of the input signal, depending on parameter 'lambda', considered as a median filter, also known as 'Hodrick Prescott' filter.

Parameters

t	Time array
x	Signal array
lamda1	Hyperparameter

Returns

An array of smooth version of signal 'x', depending on parameter 'lambda'

◆ window_phasor_angle()

```
def functions.window_phasor_angle (  t,
                                     x,
                                     sr,
                                     cycles,
                                     dom_freq = 50
                                  )
```

Angle of moving discrete fourier transform.

Parameters

t	Time array
x	Signal array
sr	Down-sampling factor
cycles	Window size as per number of cycles
dom_freq	Fundamental frequency

Returns

A complex number array of calculated fundamental phasor.

◆ window_phasor_mag()

```
def functions.window_phasor_mag (   t,
                                    x,
                                    sr,
                                    cycles,
                                    dom_freq = 50
                                )
```

Magnitude of moving discrete fourier transform.

Parameters

t	Time array
x	Signal array
sr	Down-sampling factor
cycles	Window size as per number of cycles
dom_freq	Fundamental frequency

Returns

A complex number array of calculated fundamental phasor.