elegant Documentation

Release 1.0b1

Eduardo Nunes & Fernando Dantas

Jun 21, 2022

Contents:

`elegant` is a Pure Python Power Systems Simulator. The latest development version can be found here.

CHAPTER 1

Installation

`elegant` requires `numpy` (for linear algebra), `PyQt5` (for the GUI), `pylatex` (to generate reports), `matplotlib` (for plotting), and `networkx` (for multigraph data structures)

Installing the most recent stable version of the package is as easy as:

```
python3 -m pip install elegant
```

CHAPTER 2

Changelog

2.1 1.0 (2020-00-00)

Initial beta release

2.1.1 Case Study

Inserting buses

Inserting lines

Inserting loads

2.1.2 API

This page details the methods and classes provided by the `elegant` module.

Top-Level classes

class elegant.core.**Bus**(*bus_id*, *v=1.0*, *delta=0.0*, *pg=0.0*, *qg=0.0*, *pl=0.0*, *ql=0.0*, *xd=inf*, *iTPG=None*, *iSLG=None*, *iDLGb=None*, *iDLGc=None*, *iLL=None*, *gen_ground=False*, *load_ground=1*)

Bases: object

P

Q

Z

class elegant.core.**Keys**
Bases: object

add_keyobj(*extremities*, *path*, *obj*)

get_keyobj(*extremities*, *path*)

have_extremities(*extremities*)

class elegant.core.**PowerSystem**
Bases: object

M

N

Y

Y0

Y1

add_bus()

add_line(*line*, *path=None*)

add_trafo(*trafo*, *path=None*)

good_ids

hsh

id2n(*k*)

masked_buses

masked_lines

masked_trafos

remove_bus(*n*)

remove_elements_linked_to(*bus*)

remove_line(*line*, *key=None*)

remove_trafo(*trafo*, *key=None*)

sort_buses()

update(*Nmax=100*)

update_flow(*Nmax=100*)

update_short()

class elegant.core.**Transformer**(*orig*, *dest*, *snom=100000000.0*, *jx0=0.5*, *jx1=0.5*, *primary=0*, *secondary=0*, *v1=0.0*, *v2=0.0*)
Bases: object

Ipu

S1

S2

Sper

Z0

Z1

class elegant.core.**TransmissionLine**(*orig*, *dest*, *ell=10000.0*, *r=0.01*, *d12=1*, *d23=1*, *d31=1*, *d=0.5*, *rho=1.78e-08*, *m=1*, *vbase=10000.0*, *imax=inf*, *v1=0.0*, *v2=0.0*, *z=None*, *y=None*)

Bases: object

Ia

Ipu

Rb

Rm

S1

S2

Sper

Tpu

Y

Ypu

Z

Zc

Zcpu

Zpu

gamma

param

elegant.core.**gmean**(*arr*)

Numerical Methods

elegant.methods.**short**(*Y1*, *Y0*, *V*)

Calculates three-phase short circuit current levels for each bus

Parameters

Y1: array, shape (N,N) Positive-sequence bus admittance matrix

Y0: array, shape (N,N) Zero-sequence bus admittance matrix

V: array, shape (N,) Pre-fault voltage levels for each bus

Returns

I: array, shape (N, 4, 3) Three-phase current levels for each of the N buses for each of the following fault types:

- Three-phase to ground (TPG);
- Single-line to ground (SLG);
- Double-line to ground (DLG);

- Line-to-line (LL)

elegant.methods.**gauss_seidel**(*Y*, *V0*, *S*, *eps=None*, *Niter=1*, *Nmax=1000*)

Gauss-Seidel Method

Parameters

Y: array, shape (N,N) Ybus matrix

V0: array, shape (N,) Complex initial guess

S: array, shape (N,2) Specified apparent power

eps: float, optional Tolerance

Niter: int, optional Minimum number of iterations (default=1)

Returns

V: array, shape (N,) Bus voltage approximations

elegant.methods.**newton_raphson**(*Y*, *V0*, *S*, *eps=None*, *Niter=1*, *Nmax=1000*)

Parameters

Y: admittance matrix

V0: array with initial estimates (1, N)

S: array with specified powers in each bar (N, 2)

eps: defined tolerance, default = None

Niter: max number of iterations, default = 1

Returns

V0: updated array with estimates to the node tensions (1, N)

Report files

elegant.report.**create_report**(*system*, *curves*, *grid*, *filename*)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index

T

U

Y

Z