pydgrid Documentation

Release 0.4.5

pydgrid

Nov 26, 2018

Contents:

pydgrid is an open source (MIT) electric distribution grid simulator. Grid elements are represented in phasor coordinates. Some futures are:

- Balanced and unbalanced power flow
- Time series simulation
- Time domain simulation

The source code is hosted on GitHub, and all contributions and feedback are more than welcome. You can test pydgrid in your browser using binder, a cloud Jupyter notebook server:

CHAPTER 1

Getting started

1.1 Requirements

pydgrid requires the following Python packages:

- NumPy, for basic numerical routines
- numba, for accelerating the code
- json, for reading network data
- matplotlib, for results plotting
- bokeh, for results visualization
- pytest, for running the tests from the package

pydgrid is usually tested on Linux and Windows on Python 3.5 and 3.6 against latest NumPy.

1.2 Installation

The easiest and fastest way to get the package up and running is to install anaconda with Python 3.5 or earlier.

Then you can install pydgrid from PyPI using pip:

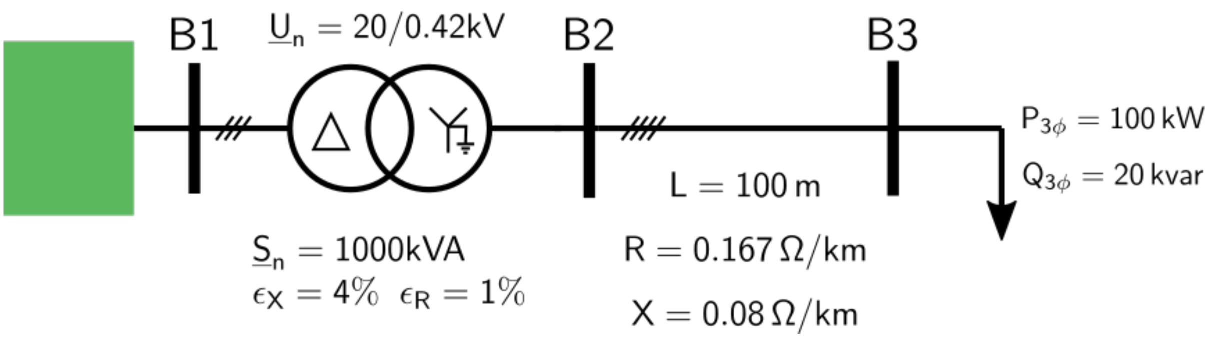
```
$ pip install pydgrid
```

Warning: It is recommended that you **never ever use sudo** with distutils, pip, setuptools and friends in Linux because you might seriously break your system [1][2][3][4]. Options are per user directories, virtualenv or local installations.

CHAPTER 2

User guide

The simplest use can be understood with an example. Suppose we want to calculate the power flow of the following system:



The following steps should be considered:

1. Import modules
2. Define or load grid parameters
3. Generate a grid instance
4. Read grid parameters
5. Run power flow
6. Post process results
7. Plot results

2.1 Import modules

First of all, we have to import the relevant modules and classes:

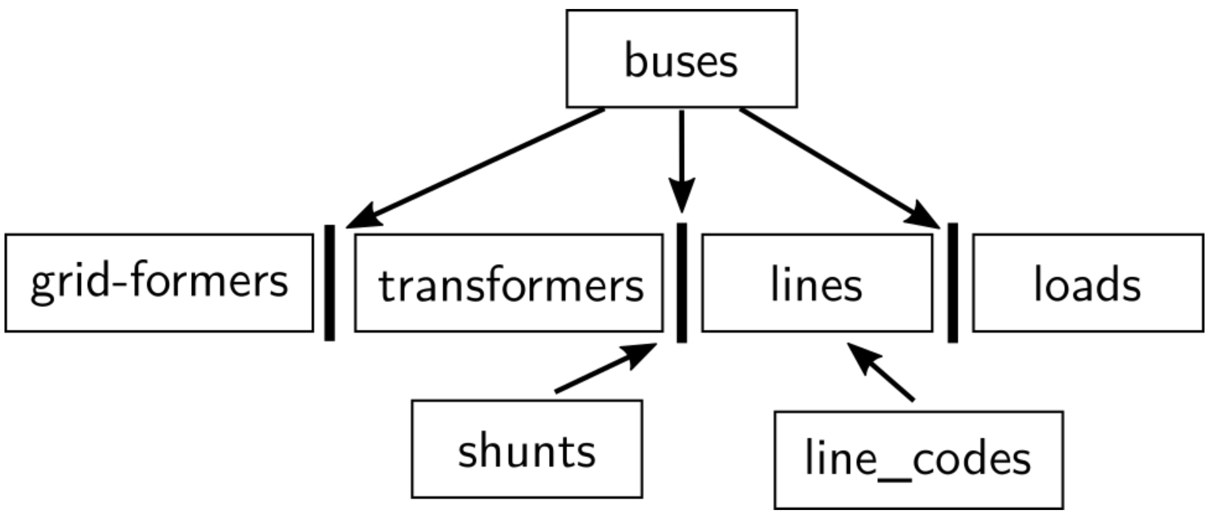
```
import numpy as np

from pydgrid import grid
```

2.2 Define or write grid parameters

The network can be introduced in two ways:

- Python dictionary
- json file with the same structure as in the case of the previous python dictionary

For the proposed system example the following elemnts from pydgrid should be considered:




```
data = {
        "buses":[
                 {"bus": "B1",  "pos_x":   0, "pos_y":   0, "units": "m", "U_kV":20.0}
↪,
                 {"bus": "B2",  "pos_x":  10, "pos_y":   0, "units": "m", "U_kV":0.4},
                 {"bus": "B3",  "pos_x": 100, "pos_y":  0, "units": "m", "U_kV":0.4}
                ],
        "grid_formers":[
                        {"bus": "B1",
                        "bus_nodes": [1, 2, 3], "deg": [0, -120, -240],
                        "kV": [11.547, 11.547, 11.547]}
                       ],
        "transformers":[
                        {"bus_j": "B1",  "bus_k": "B2",  "S_n_kVA": 1000.0, "U_j_kV
↪":20, "U_k_kV":0.42,
                         "R_cc_pu": 0.01, "X_cc_pu":0.04, "connection": "Dyn11",
↪"conductors_j": 3, "conductors_k": 4},
                       ],
        "lines":[
                 {"bus_j": "B2",  "bus_k": "B3",  "code": "lv_cu_150", "m": 100.0},
                ],
        "loads":[
                 {"bus": "B3" , "kVA": 300.0, "pf": 0.85,"type":"3P+N"}
                ],
```

(continues on next page)

(continued from previous page)

```
        "shunts":[
                 {"bus": "B2" , "R": 0.001, "X": 0.0, "bus_nodes": [4,0]}
                 ],
        "line_codes":
             {"lv_cu_150":  {"Rph":0.167,"Xph":0.08, "Rn":0.167, "Xn": 0.08}
             }
       }
```

2.3 Generate a grid instance

```
grid_1 = grid()
```

2.4 Read grid parameters

```
grid_1.read(data)
```

2.5 Execute power flow

```
grid_1.pf()
```

2.6 Plot results

In the case of using jupyter notebook results can be visualized with a bokeh plot that includes hover tools.

```
from pydgrid.plot_bokeh import plot_results
plot_results(grid_1)
```

An on-line working jupyter notebook with the same example can be obtained here:

CHAPTER 3

Definitions

3.1 Buses and Nodes

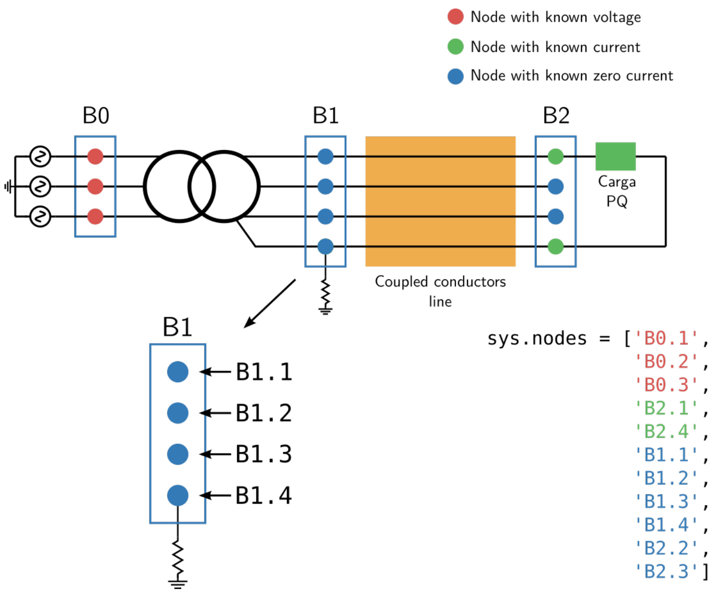
to_do

3.2 Nodes order

The nodes order in the voltages and currents vectors `V_node` and `I_node` is as follows:

- nodes with grid formers (known voltages)
- nodes with loads or greed feeders (known currents)
- transition nodes (zeros current injections)

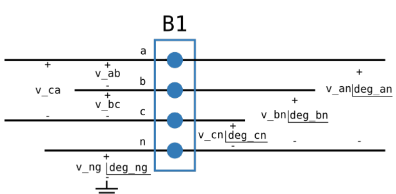
3.2.1 Example



V_known has the nodes with grid formers sources V_unknown has the rest of the nodes

V_sorted . . . Buses order is as defined in the key `buses` of the `.json` file or data dictionary.

3.3 Buses results voltages

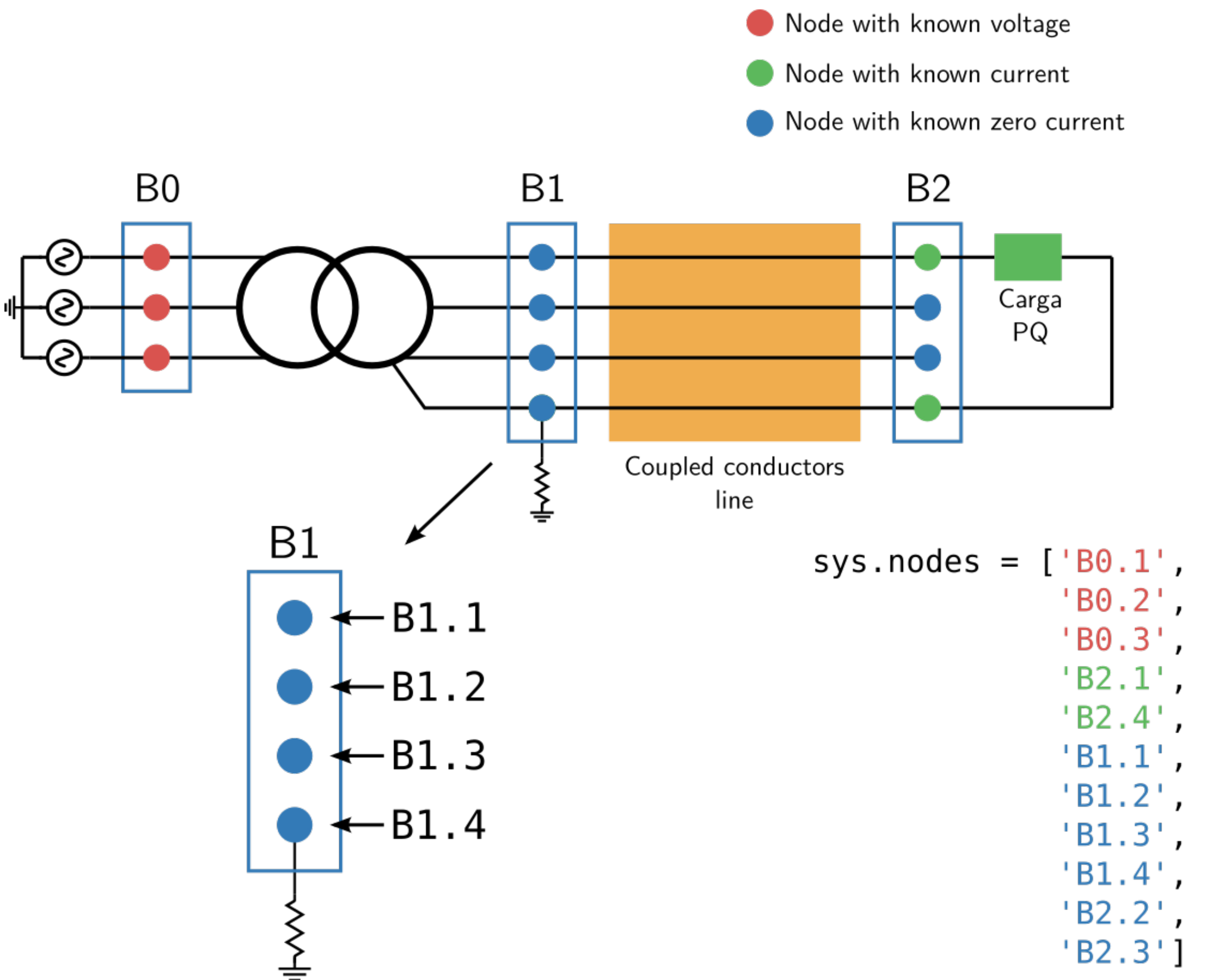


CHAPTER 4

Elements

4.1 Buses

Buses are composed by nodes.




```
"buses":[
                {"bus": "Bus_0",  "pos_x": 0, "pos_y":   0, "units": "m", "U_kV":20.0}
↪,
                {"bus": "Bus_1",  "pos_x": 0, "pos_y":  10, "units": "m", "U_kV":0.4},
                {"bus": "Bus_2",  "pos_x": 0, "pos_y": 200, "units": "m", "U_kV":0.4}
                ],
```

where:

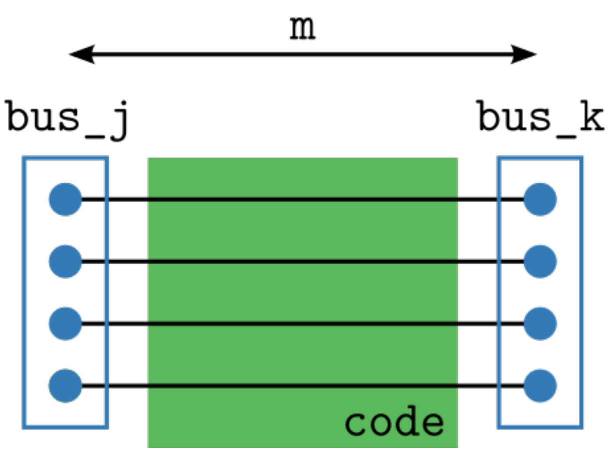
- "bus": name of the bus
- "pos_x": x position of the bus
- "pos_y": y position of the bus
- "units": units for positions (only m is available)
- "U_kV": RMS phase-phase base voltage (kV)

4.1.1 Indices and tables

- genindex
- modindex

- search

4.2 Lines




```
"lines":[
         {"bus_j": "Bus_1",  "bus_k": "Bus_2",  "code": "UG1", "m": 200.0}
        ]
```

where:

- "bus_j": name of the j bus
- "bus_k": name of the k bus
- "code": line code
- "m": line length in meters

4.3 Line codes

Two types of lines models can be considered:

- Serie Impedance (only R and X)
- PI Section Line (R, X ans shunt C)

Line parammeters can be introduced as:

- Sequence coordinates
- Primitive matrices
- Unitary voltage drop for p.f.= 1.0 and p.f.= 0.8

4.3.1 Serie impedance sequence parameters

Sequence coordinates

```
"line_codes":
    {"mv_al_50":  {"R1":0.8,    "X1":       0.148, "R0":0.8,   "X0":        0.148},
     "mv_al_95":  {"R1":0.403,  "X1":       0.129, "R0":0.403, "X0":        0.129},
     "mv_al_120": {"R1":0.321,  "X1":       0.123, "R0":0.321, "X0":        0.321},
     "mv_al_185": {"R1":0.209,  "X1":       0.113, "R0":0.209, "X0":        0.209},
     "mv_al_300": {"R1":0.128,  "X1":       0.105, "R0":0.128, "X0":        0.128}
    }
```

where:

- "R1": Positive sequence resistance (Ω/km)
- "X1": Positive sequence reactance (Ω/km)
- "R0": Zero sequence resistance (Ω/km)
- "X0": Zero sequence reactance (Ω/km)

Primitive matrices

```
"line_codes":
  {"UG1":
    {"R":[[ 0.211,  0.049,  0.049,  0.049],
          [ 0.049,  0.211,  0.049,  0.049],
          [ 0.049,  0.049,  0.211,  0.049],
          [ 0.049,  0.049,  0.049,  0.211]],
     "X":[[ 0.747,  0.673,  0.651,  0.673],
          [ 0.673,  0.747,  0.673,  0.651],
          [ 0.651,  0.673,  0.747,  0.673],
          [ 0.673,  0.651,  0.673,  0.747]]
    },
  "UG3":
    {"R":[[ 0.871,  0.049,  0.049,  0.049],
          [ 0.049,  0.871,  0.049,  0.049],
          [ 0.049,  0.049,  0.871,  0.049],
          [ 0.049,  0.049,  0.049,  0.871]],
     "X":[[ 0.797,  0.719,  0.697,  0.719],
          [ 0.719,  0.797,  0.719,  0.697],
          [ 0.697,  0.719,  0.797,  0.719],
          [ 0.719,  0.697,  0.719,  0.797]]
    }
  }
```

where:

- "R": Resistance primitive (Ω/km)
- "X": Reactance primitive (Ω/km)

Unitary voltage drop

```
"line_codes":
    {
    "lv_cu_150":  {"u90_pf10":0.27,"u90_pf08":0.31, 'T_deg':90.0, 'alpha':0.004},
    "lv_cu_240":  {"u90_pf10":0.17,"u90_pf08":0.27, 'T_deg':90.0, 'alpha':0.004}
    }
```

where:

- "u90_pf10": Unitary voltage drop with load cos(ϕ) = 1.0 at 90ºC (V/(km A))
- "u90_pf08": Unitary voltage drop with load cos(ϕ) = 0.8 at 90ºC (V/(km A))
- "T_deg": Current conductor temperature (ºC)
- "alpha": Temperature coefficient (1/ºC)

PI section sequence parameters

```
"line_codes":
    {
     "mv_cu_50_pi":  {"R1":0.387,   "X1":    0.152, "R0":0.387, "X0":         0.152, "C_
↪1_muF":0.135, "C_0_muF":0.135},
     "mv_cu_95_pi":  {"R1":0.193,   "X1":    0.136, "R0":0.193, "X0":         0.136, "C_
↪1_muF":0.175, "C_0_muF":0.175},
     "mv_cu_120_pi": {"R1":0.153,   "X1":    0.132, "R0":0.153, "X0":         0.132, "C_
↪1_muF":0.186, "C_0_muF":0.186},
     "mv_cu_185_pi": {"R1":0.099,   "X1":    0.121, "R0":0.099, "X0":         0.121, "C_
↪1_muF":0.226, "C_0_muF":0.226},
     "mv_cu_300_pi": {"R1":0.060,   "X1":    0.112, "R0":0.060, "X0":         0.112, "C_
↪1_muF":0.275, "C_0_muF":0.275}
    }
```

where:

- "R1": Positive sequence resistance (Ω/km)
- "X1": Positive sequence reactance (Ω/km)
- "R0": Zero sequence resistance (Ω/km)
- "X0": Zero sequence reactance (Ω/km)
- "C_1_muF": Zero sequence resistance (μF/km)
- "C_0_muF": Zero sequence reactance (μF/km)

Serie impedance primitives

PI section sequence parameters

```
"line_codes":
{
 "K1":
    {"R":[[0.8667, 0.2955, 0.2907],
          [0.2955, 0.8837, 0.2992],
          [0.2907, 0.2992, 0.8741]],
     "X":[[2.0417,0.9502, 0.7290],
          [0.9502,1.9852, 0.8023],
          [0.7290,0.8023, 2.0172]],
     "B_mu":[[10.7409, -3.4777, -1.3322],
             [-3.4777, 11.3208, -2.2140],
             [ -1.3322, -2.2140, 10.2104]]}
}
```

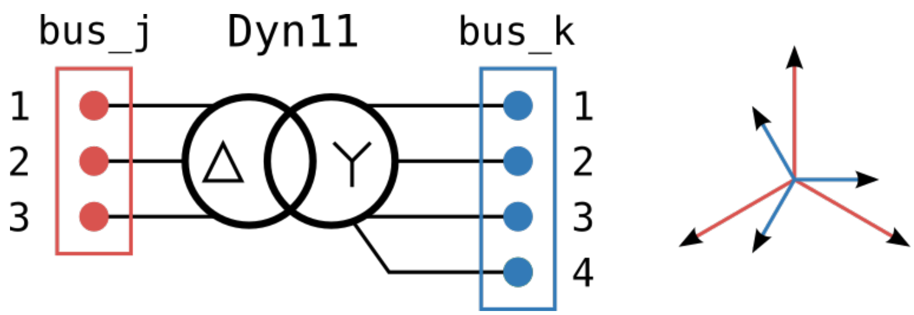
where:

- "R": Resistance primitive (Ω/km)
- "X": Reactance primitive (Ω/km)
- "B_mu": Zero sequence resistance (μ℧/km)

4.4 Transformers

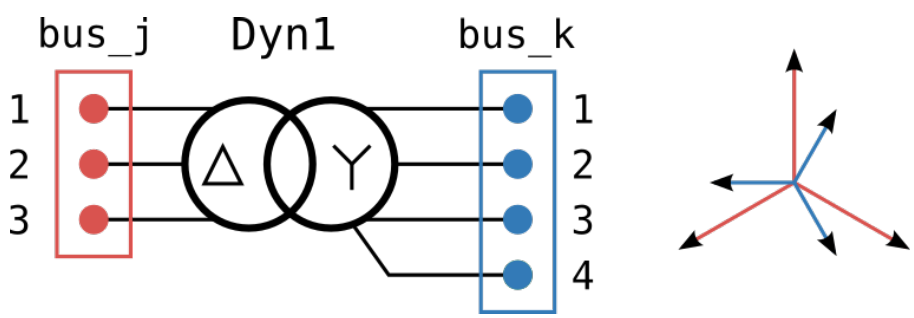
Transformers are modeled as in *[T1]*.

4.4.1 Dyn11



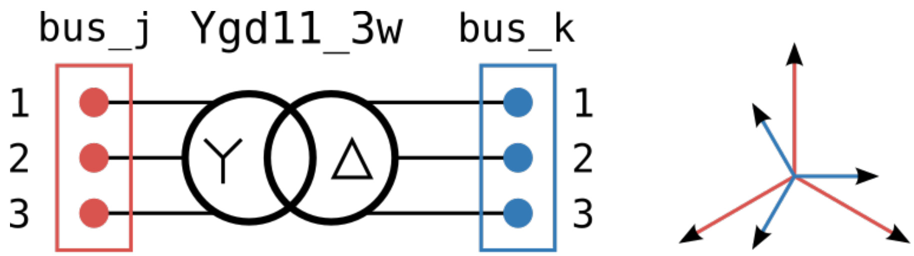

```
"transformers":[
 {"bus_j": "Bus_0",  "bus_k": "Bus_1",  "S_n_kVA": 1000.0, "U_j_kV":20, "U_k_kV":0.42,
  "R_cc_pu": 0.01, "X_cc_pu":0.04, "connection": "Dyn11",   "conductors_j": 3,
↪"conductors_k": 4},
                ],
```

4.4.2 Dyn1




```
"transformers":[
 {"bus_j": "Bus_0",  "bus_k": "Bus_1",  "S_n_kVA": 1000.0, "U_j_kV":20, "U_k_kV":0.42,
  "R_cc_pu": 0.01, "X_cc_pu":0.04, "connection": "Dyn1",   "conductors_j": 3,
↪"conductors_k": 4},
                ],
```

4.4.3 Ygd11_3w




```
"transformers":[
            {"bus_j": "Bus_0",  "bus_k": "Bus_1",  "S_n_kVA": 2500.0, "U_j_kV":20, "U_
↪k_kV":0.69,
             "R_cc_pu": 0.01, "X_cc_pu":0.04, "connection": "Ygd11_3w",   "conductors_
↪j": 3, "conductors_k": 3},
                 ],
```

where:

- "bus_j": name of the j bus
- "bus_k": name of the k bus
- "pos_x": x position of the bus
- "pos_y": y position of the bus
- "S_n_kVA": based power in kVA
- "U_j_kV": HV side nominal RMS phase-phase voltage in kV
- "U_k_kV": LV side nominal RMS phase-phase voltage in kV
- "connection": connection type (see available connections)
- "conductors_j": HV side conductors
- "conductors_k": LV side conductors

4.5 Grid formers

Grid formers are considered as fix voltage sources in the power flow calculation.

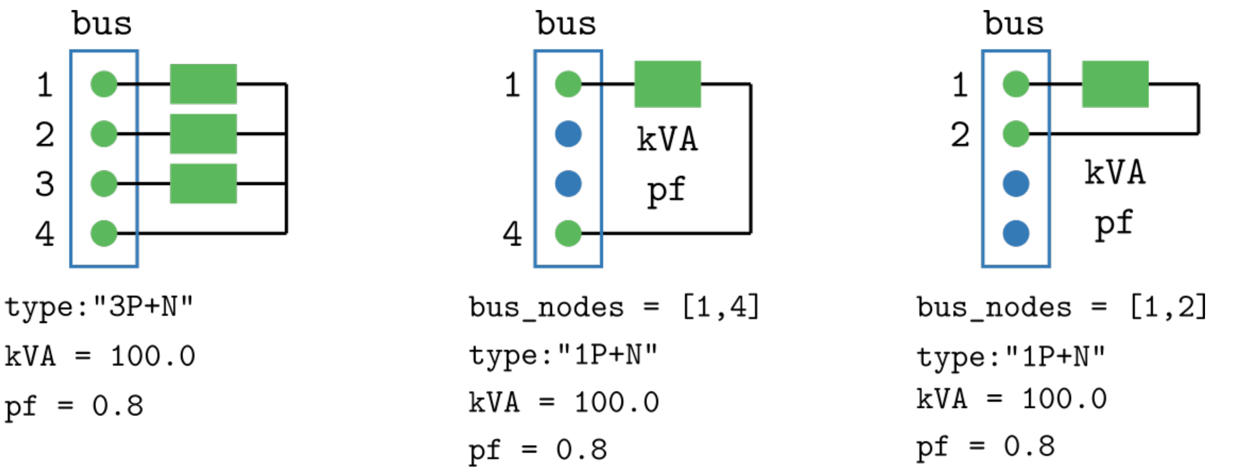
```
"grid_formers":[
                  {"bus": "Bus_1",
                   "bus_nodes": [1, 2, 3], "deg": [0, -120, -240],
                   "kV": [0.23094, 0.23094, 0.23094]}
                ]
```

where:

- "bus": name of the bus
- "bus_nodes": list of nodes where the grid former source is connected
- "kV": phase-neutral RMS voltages list (kV)
- "deg": phase-neutral voltage angles list (deg)

4.6 Loads

Loads.




```
"loads":[
         {"bus": "Bus_2" , "kVA": 50.0, "pf": 0.85,"type":"1P+N","bus_nodes": [1,4]},
         {"bus": "Bus_2" , "kVA": 30.0, "pf": 0.85,"type":"1P+N","bus_nodes": [2,4]},
         {"bus": "Bus_2" , "kVA": 20.0, "pf": 0.85,"type":"1P+N","bus_nodes": [3,4]}
        ],
```

where:

- `"bus"`: name of the bus
- `"bus_nodes"`: list of nodes where the load is connected
- `"type"`: available types are: - `"3P+N"`: three phase load with neutral - `"3P"`: three phase load without neutral - `"1P+N"`: single phase load connected between one phase and other or neutral
- `"kVA"`: aparent power (kW)
- `"pf"`: load power factor

4.7 Grid feeders

Grid feeders are considered as fix power or current sources in the power flow calculation.

```
"grid_feeders":[{"bus": "Bus_2","bus_nodes": [1, 2, 3,4],
                 "kW": [0.5,0.5,0.5], "kvar": [0,0,0],
                 "kA": [0,0,0], "phi_deg":[30, 30, 30]}
               ]
```

where:

- `"bus"`: name of the bus
- `"bus_nodes"`: list of nodes where the grid former source is connected
- `"kW"`: active power for each phase
- `"kvar"`: reactive power for each phase
- `"kA"`: RMS value of the current in each phase

- "phi_deg": angle between voltages and currents

4.7.1 Voltage Source Converter (VSC)

```
"grid_feeders":[{"bus": "Bus_2","bus_nodes": [1, 2, 3],
                 "type":"vsc","control_mode":"pq_leon",
                 "kW": 500.0, "kvar": 200.0,
                 "L":400e-6, "R":0.01,"V_dc":800.0}
               ]
```

4.8 Shunt elements

Impedances that can be connected beetwen phases, phases and neutral and phases and neutral and ground.

```
"shunts":[
          {"bus": "Bus_1" , "R": 0.001, "X": 0.0, "bus_nodes": [4,0]},
          {"bus": "Bus_2" , "R":  40.0, "X": 0.0, "bus_nodes": [4,0]}
         ]
```

where:

- "bus": name of the bus
- "bus_node": list of nodes where the shunt element is connected
- "R": shunt element resistance (Ω)
- "X": shunt element reactance (Ω)

CHAPTER 5

Jupyter Notebooks

```
In [1]: import numpy as np
        from pydgrid.pydgrid import grid
        from pydgrid.pf import pf_eval,time_serie
```

5.1 Kersting book example 6.1

5.1.1 Data

```
In [2]: data = {
        "lines":[
                {"bus_j": "Bus_1",  "bus_k": "Bus_2",  "code": "Kersting", "m": 1609.34}
                ],
        "buses":[
                    {"bus": "Bus_1",  "pos_x": 10, "pos_y":  0, "units": "m", "U_kV":12.47},
                    {"bus": "Bus_2",  "pos_x": 200, "pos_y": 0, "units": "m", "U_kV":12.47}
                    ],
        "grid_formers":[
                    {"bus": "Bus_1","bus_nodes": [1, 2, 3],
                            "kV": [7.53869, 7.45125, 7.48512],
                            "deg": [1.57248, -118.30047, 121.93184]
                    }
                    ],
        "loads":[
                    {"bus": "Bus_2" , "kVA": 6000.0, "pf": 0.9,"type":"3P"}
                ],
        "line_codes":
                    {"Kersting":
                    {"R":[
                                            [0.8667, 0.2955, 0.2907],
                                            [0.2955, 0.8837, 0.2992],
                                            [0.2907, 0.2992, 0.8741]
                                            ],
```

```
                    "X":[
                                   [2.0417,0.9502, 0.7290],
                 [0.9502,1.9852, 0.8023],
                 [0.7290,0.8023, 2.0172]
                                       ],
            "B_mu":[
                                                  [10.7409, -3.4777, -1.3322],
                                                  [-3.4777, 11.3208, -2.2140],
                                                  [ -1.3322, -2.2140, 10.2104]
                                         ],
        "unit":"miles"
                    }
                    }
        }
```

5.1.2 Execute power flow

```
In [3]: grid_1 = grid()
        grid_1.read(data)  # Load data
        grid_1.pf_solver = 2
        grid_1.pf()  # solve power flow
```

5.1.3 Graph with obtained results

```
In [4]: from pydgrid.plot_bokeh import plot_results
        plot_results(grid_1)
```

Data type cannot be displayed: application/vnd.bokehjs_load.v0+json, application/javascript

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
Out[4]: Figure(id='1003', ...)
```

5.1.4 Get element transfomers results

```
In [6]: grid_1.buses

Out[6]: [{'bus': 'Bus_1',
          'pos_x': 10,
          'pos_y': 0,
          'units': 'm',
          'U_kV': 12.47,
          'N_nodes': 3,
          'v_an': 7538.69,
          'v_bn': 7451.25,
          'v_cn': 7485.12,
          'v_ng': 0.0,
          'deg_an': 1.57248,
          'deg_bn': -118.30046999999999,
          'deg_cn': 121.93184,
          'deg_ng': 0.0,
```

```
           'v_ab': 12973.42498856987,
           'v_bc': 12920.120310382239,
           'v_ca': 13034.52171783379,
           'p_a': 1858935.7393203387,
           'p_b': 1835419.9515336645,
           'p_c': 1839830.0481158614,
           'q_a': 963491.4173624129,
           'q_b': 956338.0192537266,
           'q_c': 968270.9001076091},
          {'bus': 'Bus_2',
           'pos_x': 200,
           'pos_y': 0,
           'units': 'm',
           'U_kV': 12.47,
           'N_nodes': 3,
           'v_an': 7199.551320077657,
           'v_bn': 7199.557391530307,
           'v_cn': 7199.55602482274,
           'v_ng': 0.0,
           'deg_an': -3.170444377827802e-05,
           'deg_bn': -120.00002917542324,
           'deg_cn': 119.9999628259816,
           'deg_ng': 0.0,
           'v_ab': 12469.993777213469,
           'v_bc': 12469.998513070677,
           'v_ca': 12469.992408860844,
           'p_a': -1799998.8481163539,
           'p_b': -1800000.327593055,
           'p_c': -1800000.107596966,
           'q_a': -871778.0011666914,
           'q_b': -871778.8157972179,
           'q_c': -871778.399022273}]
```

In [ ]:

In [ ]:

```
In [3]: import numpy as np
        from pydgrid.pydgrid import grid
```

5.2 3 bus 4 wire system with transformer

5.2.1 Data

```
In [13]: data = {
                 "buses":[
                          {"bus": "B1",  "pos_x":   0, "pos_y":   0, "units": "m", "U_kV":20.0},
                          {"bus": "B2",  "pos_x":  10, "pos_y":   0, "units": "m", "U_kV":0.4},
                          {"bus": "B3",  "pos_x": 100, "pos_y":  0, "units": "m", "U_kV":0.4}
                         ],
                 "grid_formers":[
                                 {"bus": "B1",
                                 "bus_nodes": [1, 2, 3], "deg": [0, -120, -240],
                                 "kV": [11.547, 11.547, 11.547]}
                                ],
                 "transformers":[
                                 {"bus_j": "B1",  "bus_k": "B2",  "S_n_kVA": 1000.0, "U_j_kV":20, "U_
                                  "R_cc_pu": 0.01, "X_cc_pu":0.04, "connection": "Dyn11",   "conducto
```

```
                        ],
                "lines":[
                         {"bus_j": "B2",  "bus_k": "B3",  "code": "lv_cu_150", "m": 100.0},
                        ],
                "loads":[{"bus": "B3" , "kVA": 300.0/3, "pf": 0.85,"type":"3P+N"},
                         {"bus": "B3" , "kVA": 300.0/3, "pf": 0.85,"type":"1P+N", 'bus_nodes':[1,4]
                        ],
                "shunts":[
                         {"bus": "B2" , "R": 0.001, "X": 0.0, "bus_nodes": [4,0]}
                         ],
                "line_codes":
                    {"lv_cu_150":  {"Rph":0.167,"Xph":0.08, "Rn":0.167, "Xn": 0.08}
                    }
               }
```

5.2.2 Execute power flow

```
In [14]: grid_1 = grid()
         grid_1.read(data)  # Load data

         grid_1.pf()  # solve power flow
```

5.2.3 Graph with obtained results

```
In [15]: from pydgrid.plot_bokeh import plot_results
         plot_results(grid_1)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
Out[15]: Figure(id='1309', ...)
```

```
In [ ]:
```

```
In [ ]:
```

```
In [1]: import numpy as np
        import time
        from pydgrid.pydgrid import grid
        from pydgrid.pf import pf_eval,time_serie
        from bokeh.io import output_notebook, show
        from bokeh.plotting import figure
        from bokeh.models import ColumnDataSource, HoverTool
        from bokeh.io import push_notebook
        from bokeh.resources import INLINE
        output_notebook(INLINE)
```

Data type cannot be displayed: application/vnd.bokehjs_load.v0+json, application/javascript

5.3 907 bus 3 wire system with transformer

5.3.1 Execute power flow

```
In [2]: sys1 = grid()
        sys1.read('n1_f1.json')  # Load data
        sys1.read_loads_shapes('n1_f1_load_shapes.json')
        sys1.pf_solver = 2
        sys1.pf()  # solve power flow

        sys1.get_v()      # post process voltages
        sys1.get_i()      # post process currents
```

5.3.2 Graph with obtained results

```
In [7]: sys1.s_radio_scale =0.5
        sys1.s_radio_min =5
        sys1.s_radio_max =100
        sys1.snapshot(60*60*12)
        sys1.pf()
        sys1.get_v()      # post process voltages
        sys1.get_i()      # post process currents
        sys1.bokeh_tools()

        p_grid = figure(width=900, height=800,
                   title='3 bus 4 wire system with transformer')

        # lines:
        source = ColumnDataSource(sys1.line_data)
        lin = p_grid.multi_line(source=source, xs='x_s', ys='y_s', color="red", alpha=0.5, line_width

        # buses:
        source = ColumnDataSource(sys1.bus_data)
        cr = p_grid.circle(source=source, x='x', y='y', size='s_radio', color="s_color", alpha=0.5)

        p_grid.add_tools(HoverTool(renderers=[lin], tooltips=sys1.line_tooltip))
        p_grid.add_tools(HoverTool(renderers=[cr], tooltips=sys1.bus_tooltip))


        def update_grid(t_h=0.0):
            sys1.snapshot(60*60*t_h)

            sys1.get_v()
            sys1.get_i()
            sys1.bokeh_tools()

            source.data = sys1.bus_data
            push_notebook()



        #p_grid = gridplot([[p], [p_2]])
        show(p_grid, notebook_handle=True)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
Out[7]: <bokeh.io.notebook.CommsHandle at 0x7fe010a07208>

In [8]: from ipywidgets import interact
        interact(update_grid, t_h=(0,24, 0.01), continuous_update=False)

interactive(children=(FloatSlider(value=0.0, description='t_h', max=24.0, step=0.01), Output()), _dor

Out[8]: <function __main__.update_grid(t_h=0.0)>
```

5.3.3 Interaction

```
In [5]: p = figure(width=600, height=300,
                   title='Voltage vs load powers',
                   y_range = [0.95,1.01], #x_range = [50,-300],
                   x_axis_label='Distance (m)',
                   y_axis_label='Voltage (V)')
        source = ColumnDataSource(sys1.bus_data)
        #cr = p1.circle(source=source, x='y', y='v_an_pu', size=15, color="red", alpha=0.5)
        p.circle(source=source, x='x', y='v_an_pu', size=15, color="red", alpha=0.5)
        p.circle(source=source, x='x', y='v_bn_pu', size=15, color="green", alpha=0.5)
        p.circle(source=source, x='x', y='v_cn_pu', size=15, color="blue", alpha=0.5)

        #p1.circle(source=source, x='y', y='v_bn_pu', size=15, color="green", alpha=0.5)
        #p.circle(source=source, x='y', y='v_cn', size=15, color="blue", alpha=0.5)
        #p.line([-300, 50],[231*1.05,231*1.05], color='red', line_width=5)
        #p.line([-300, 50],[231*0.90,231*0.90], color='blue', line_width=5)
        #p.add_tools(HoverTool(renderers=[cr], tooltips=sys1.bus_tooltip))

        def update(t_h=0.0):
            sys1.snapshot(60*60*t_h)

            sys1.get_v()
            sys1.get_i()
            sys1.bokeh_tools()

            source.data = sys1.bus_data
            push_notebook()

        #p_grid = gridplot([[p], [p_2]])
        show(p, notebook_handle=True)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
Out[5]: <bokeh.io.notebook.CommsHandle at 0x7fe01c26fda0>

In [6]: from ipywidgets import interact
        interact(update, t_h=(0,24, 0.01))

A Jupyter Widget

Out[6]: <function __main__.update>

In [ ]:

In [ ]:
```

```
In [11]: import numpy as np
         from pydgrid.pydgrid import grid
```

5.4 CIGRE LV European System

```
In [14]: grid_1 = grid()
         grid_1.read('cigre_europe_residential.json')  # Load data
         grid_1.pf()
```

5.4.1 Graph with obtained results

```
In [15]: from pydgrid.plot_bokeh import plot_results
         plot_results(grid_1)
```

Data type cannot be displayed: application/vnd.bokehjs_load.v0+json, application/javascript

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
Out[15]: Figure(id='1310', ...)
```

5.4.2 Analysis using Pandas

```
In [16]: import pandas as pd
```

```
In [17]: df = pd.DataFrame()
         df['nodes'] = sys1.nodes
         df['I_node_m'] = np.abs(sys1.I_node)
         df['V_node_m'] = np.abs(sys1.V_node)
         #df['phi'] = np.angle(sys1.I_node, deg=True) - np.angle(sys1.V_node, deg=True)
         s  = np.conjugate(sys1.I_node)*sys1.V_node
         df['p_kW']  = s.real/1000
         df['q_kVA'] = s.imag/1000

         df
```

Out[17]:

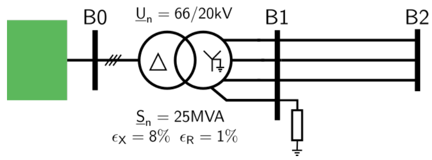
	nodes	I_node_m	V_node_m	p_kW	q_kVA
0	R0.1	12.279686	11547.000000	133.228235	48.534983
1	R0.2	12.194776	11547.000000	132.224879	48.424213
2	R0.3	12.219433	11547.000000	132.822486	47.610666
3	R1.1	294.305597	226.183736	-63.206394	-20.883865
4	R1.2	294.318022	226.377454	-63.338571	-20.673117
5	R1.3	294.268246	227.024469	-63.455017	-20.892937
6	R1.4	0.129777	0.488038	-0.000018	-0.000061
7	R11.1	22.619924	221.082177	-4.751117	-1.560611
8	R11.2	22.542696	221.818706	-4.749990	-1.562532
9	R11.3	22.431808	222.842141	-4.748898	-1.560605
10	R11.4	0.111885	0.056907	0.000006	-0.000001
11	R15.1	82.678522	210.089330	-16.512848	-5.388732
12	R15.2	82.063130	211.381109	-16.463661	-5.463720
13	R15.3	81.216693	212.814945	-16.424028	-5.384451
14	R15.4	0.869274	0.627272	0.000537	-0.000093
15	R16.1	85.643246	214.560691	-17.470587	-5.695963

```
16   R16.2   84.972116    215.951166  -17.414043  -5.785090
17   R16.3   84.050216    217.432385  -17.366033  -5.692562
18   R16.4    0.947507      0.712773    0.000663  -0.000129
19   R17.1   55.050752    212.626217  -11.132419  -3.616867
20   R17.2   54.506010    214.317220  -11.080836  -3.697878
21   R17.3   53.767616    216.004660  -11.037499  -3.613848
22   R17.4    0.762900      1.009420    0.000755  -0.000153
23   R18.1   74.293759    211.633036  -14.954995  -4.853998
24   R18.2   73.514087    213.405120  -14.879548  -4.972049
25   R18.3   72.461012    215.150835  -14.816623  -4.849461
26   R18.4    1.089448      1.091264    0.001165  -0.000237
27    R2.1    0.000000    223.960103    0.000000  -0.000000
28    R2.2    0.000000    224.415535   -0.000000   0.000000
29    R2.3    0.000000    225.243290    0.000000   0.000000
..     ...         ...           ...         ...         ...
45    R6.3    0.000000    219.425766    0.000000   0.000000
46    R6.4    0.000000      0.634712    0.000000  -0.000000
47    R7.1    0.000000    215.787875    0.000000  -0.000000
48    R7.2    0.000000    217.212013   -0.000000   0.000000
49    R7.3    0.000000    218.710282    0.000000   0.000000
50    R7.4    0.000000      0.741778    0.000000  -0.000000
51    R8.1    0.000000    214.889967    0.000000  -0.000000
52    R8.2    0.000000    216.422120   -0.000000   0.000000
53    R8.3    0.000000    217.994913    0.000000   0.000000
54    R8.4    0.000000      0.848844    0.000000  -0.000000
55    R9.1    0.000000    213.992505    0.000000  -0.000000
56    R9.2    0.000000    215.631806   -0.000000   0.000000
57    R9.3    0.000000    217.279749    0.000000   0.000000
58    R9.4    0.000000      0.956068    0.000000  -0.000000
59   R10.1    0.000000    213.476942    0.000000  -0.000000
60   R10.2    0.000000    215.178180   -0.000000   0.000000
61   R10.3    0.000000    216.869265    0.000000   0.000000
62   R10.4    0.000000      1.017707    0.000000  -0.000000
63   R12.1    0.000000    217.273533    0.000000  -0.000000
64   R12.2    0.000000    218.317615   -0.000000   0.000000
65   R12.3    0.000000    219.559931    0.000000   0.000000
66   R12.4    0.000000      0.367154    0.000000  -0.000000
67   R13.1    0.000000    214.877482    0.000000  -0.000000
68   R13.2    0.000000    216.004358   -0.000000   0.000000
69   R13.3    0.000000    217.310579    0.000000   0.000000
70   R13.4    0.000000      0.453609    0.000000  -0.000000
71   R14.1    0.000000    212.482806    0.000000  -0.000000
72   R14.2    0.000000    213.692088   -0.000000   0.000000
73   R14.3    0.000000    215.062258    0.000000   0.000000
74   R14.4    0.000000      0.540432    0.000000  -0.000000

[75 rows x 5 columns]
```

```
In [ ]:
```

5.5 Short circuits in MV line

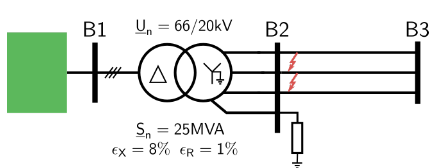
```
In [12]: import numpy as np
         from pydgrid import grid
         from pydgrid.plot_bokeh import plot_results
```



5.5.1 System

```
In [2]: data = {
        "transformers":[
                {"bus_j": "B0",  "bus_k": "B1",  "S_n_kVA": 250000.0, "U_j_kV":66.0, "U_k_kV":20.0,
                "R_cc_pu": 0.01, "X_cc_pu":0.08, "connection": "Dyn1",
                "conductors_1": 3, "conductors_2": 4}
                ],
        "lines":[
                {"bus_j": "B1",  "bus_k": "B2",  "code": "mv_al_120", "m": 200.0}
                ],
        "buses":[
                {"bus": "B0",  "pos_x": 0,  "pos_y": 0, "units": "m", "U_kV":66.0},
                {"bus": "B1",  "pos_x":30,  "pos_y": 0, "units": "m", "U_kV":20.0},
                {"bus": "B2",  "pos_x":200, "pos_y": 0, "units": "m", "U_kV":20.0}
                ],
        "grid_formers":[
                {"bus": "B0",
                "bus_nodes": [1, 2, 3], "deg": [0, -120, -240],
                "kV": [38.105, 38.105, 38.105]}
                ],
        "grid_feeders":[{"bus": "B2","bus_nodes": [1, 2, 3],
                        "kW": [0, 0, 0], "kvar": [0,0,0],
                        "kA": [0,0,0], "phi_deg":[30, 30, 30]}
                        ],
            "line_codes":
                {"mv_al_50":  {"R1":0.8,    "X1":   0.148, "R0":0.8,   "X0":        0.148},
                 "mv_al_95":  {"R1":0.403,  "X1":   0.129, "R0":0.403, "X0":        0.129},
                 "mv_al_120": {"R1":0.321,  "X1":   0.123, "R0":0.321, "X0":        0.321},
                 "mv_al_185": {"R1":0.209,  "X1":   0.113, "R0":0.209, "X0":        0.209},
                 "mv_al_300": {"R1":0.128,  "X1":   0.105, "R0":0.128, "X0":        0.128}
                 },
        "shunts":[
                {"bus": "B1" , "R": 1e12, "X": 0.0, "bus_nodes": [1,0]},
                {"bus": "B1" , "R": 1e-8, "X": 0.0, "bus_nodes": [1,2]}, # applied fault
                {"bus": "B1" , "R": 1e-8, "X": 0.0, "bus_nodes": [2,3]}, # applied fault
                ]
        }
```

5.5.2 Three phase short circuit

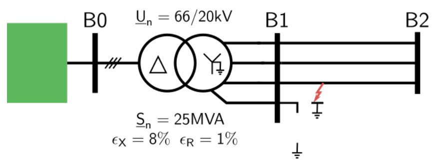



```
In [3]: grid_1 = grid()
        grid_1.read(data)   # Load data
```

```
        grid_1.pf()  # solve power flow
        p=plot_results(grid_1)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

5.5.3 Phase ground fault (isolated neutral)

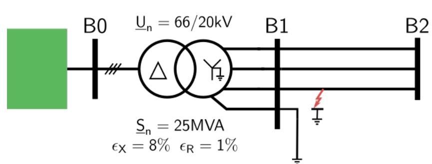



```
In [4]: data['shunts'] = [
                {"bus": "B1" , "R": 1e12, "X": 0.0, "bus_nodes": [1,0]},
                {"bus": "B1" , "R": 1e-8, "X": 0.0, "bus_nodes": [1,0]}, # applied fault
        ]
```

```
In [5]: grid_ph_g = grid()
        grid_ph_g.read(data)  # Load data
        grid_ph_g.pf()  # solve power flow
        p=plot_results(grid_ph_g)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

5.5.4 Phase ground fault (grounded neutral)



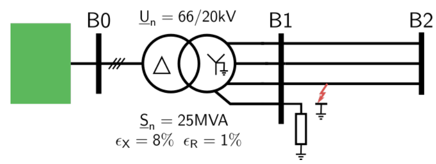

```
In [6]: data['shunts'] = [
                {"bus": "B1" , "R": 1e-12, "X": 0.0, "bus_nodes": [4,0]},
                {"bus": "B1" , "R": 1e-12, "X": 0.0, "bus_nodes": [1,0]}, # applied fault
        ]
```

```
In [7]: grid_ph_n = grid()
        grid_ph_n.read(data)  # Load data
        grid_ph_n.pf()  # solve power flow
        p=plot_results(grid_ph_n)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

5.5.5 Phase ground fault (grounded neutral with impedance)

```
In [10]: data['shunts'] = [
                 {"bus": "B1" , "R": 12.0, "X": 0.0, "bus_nodes": [4,0]},
```




```
            {"bus": "B1" , "R": 1e-12, "X": 0.0, "bus_nodes": [1,0]}, # applied fault
    ]
```

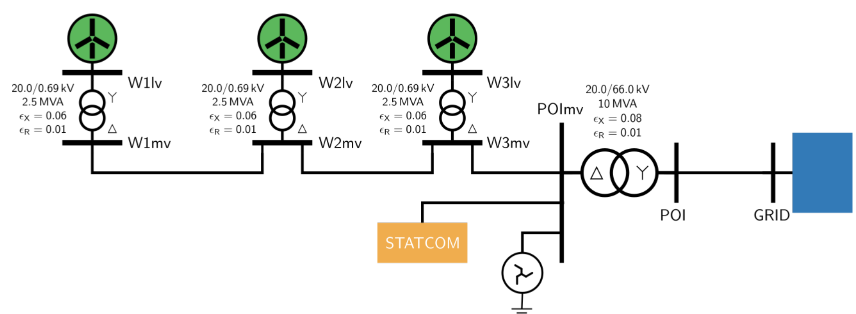
```
In [11]: grid_ph_n = grid()
         grid_ph_n.read(data)  # Load data
         grid_ph_n.pf()  # solve power flow
         p=plot_results(grid_ph_n)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

Run this notebook in binder:

5.6 Wind farm

```
In [207]: import numpy as np
          from pydgrid import grid
          from pydgrid.plot_bokeh import plot_results
```



5.6.1 System

```
In [210]: p_gen = 2000.0   # kW
          q_gen = 0        # kvar

          p_statcom = 0.0  # kW
          q_statcom = 0.0     # kvar

          data = {
              "lines":[
                  {"bus_j": "W1mv",  "bus_k": "W2mv",   "code": "mv_al_300", "m":500},
                  {"bus_j": "W2mv",  "bus_k": "W3mv",   "code": "mv_al_300", "m":500},
                  {"bus_j": "W3mv",  "bus_k": "POImv",  "code": "mv_al_300", "m":500},
                  {"bus_j": "POI",  "bus_k": "GRID",  "code": "hv_line", "m":50.0e3},
                      ],
              "buses":[
```

```
                {"bus": "W1lv",  "pos_x": -1500.0, "pos_y":  200.0, "units": "m", "U_kV":0.69},
                {"bus": "W2lv",  "pos_x": -1000.0, "pos_y":  200.0, "units": "m", "U_kV":0.69},
                {"bus": "W3lv",  "pos_x":  -500.0, "pos_y":  200.0, "units": "m", "U_kV":0.69},
                {"bus": "W1mv",  "pos_x": -1500.0, "pos_y":  180.0, "units": "m", "U_kV":20.0},
                {"bus": "W2mv",  "pos_x": -1000.0, "pos_y":  180.0, "units": "m", "U_kV":20.0},
                {"bus": "W3mv",  "pos_x":  -500.0, "pos_y":  180.0, "units": "m", "U_kV":20.0},
                {"bus": "POImv", "pos_x":     0.0, "pos_y":    0.0, "units": "m", "U_kV":20.0},
                {"bus": "POI",   "pos_x":   100.0, "pos_y":    0.0, "units": "m", "U_kV":66.0},
                {"bus": "GRID",  "pos_x":   500.0, "pos_y":    0.0, "units": "m", "U_kV":66.0},
                        ],
        "transformers":[
                        {"bus_j": "POImv",  "bus_k": "POI",  "S_n_kVA": 10000.0, "U_1_kV":20.0,
                        "R_cc_pu": 0.01, "X_cc_pu":0.08, "connection": "Dyg11_3w",   "conductor
                        {"bus_j": "W1mv",  "bus_k": "W1lv",  "S_n_kVA": 2500.0, "U_1_kV":20, "U
                        "R_cc_pu": 0.01, "X_cc_pu":0.06, "connection": "Dyg11_3w",   "conductor
                        {"bus_j": "W2mv",  "bus_k": "W2lv",  "S_n_kVA": 2500.0, "U_1_kV":20, "U
                        "R_cc_pu": 0.01, "X_cc_pu":0.06, "connection": "Dyg11_3w",   "conductor
                        {"bus_j": "W3mv",  "bus_k": "W3lv",  "S_n_kVA": 2500.0, "U_1_kV":20, "U
                        "R_cc_pu": 0.01, "X_cc_pu":0.06, "connection": "Dyg11_3w",   "conductor
                        ],
        "grid_formers":[
                        {"bus": "GRID","bus_nodes": [1, 2, 3],
                                "kV": [38.105, 38.105, 38.105], "deg": [30, 150, 270.0]
                        }
                        ],
        "grid_feeders":[{"bus": "W1lv","bus_nodes": [1, 2, 3],"kW": p_gen, "kvar": q_gen},
                        {"bus": "W2lv","bus_nodes": [1, 2, 3],"kW": p_gen, "kvar": q_gen},
                        {"bus": "W3lv","bus_nodes": [1, 2, 3],"kW": p_gen, "kvar": q_gen},
                        {"bus": "POImv","bus_nodes": [1, 2, 3],"kW": p_statcom, "kvar": q_statc
        ],
        "groundings":[
            {"bus": "POImv" , "R_gnd":32.0, "X_gnd":0.0, "conductors": 3}
          ],
        "line_codes":
            {
            "mv_al_150":   {"R1":0.262,  "X1":0.118, "C_1_muF":0.250 },
            "mv_al_185":   {"R1":0.209,  "X1":0.113, "C_1_muF":0.281 },
            "mv_al_240":   {"R1":0.161,  "X1":0.109, "C_1_muF":0.301 },
            "mv_al_300":   {"R1":0.128,  "X1":0.105, "C_1_muF":0.340 },
             "hv_line":  {"R1":0.219, "X1":0.365, "R0":0.219,   "X0":0.365}
            }
        }
```

```
In [211]: grid_1 = grid()
          grid_1.read(data)  # Load data
          grid_1.pf()  # solve power flow
          p=plot_results(grid_1)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
In [214]: mon = grid_1.monitor(bus_from='POI',bus_to='GRID')
          mon.P
```

```
Out[214]: 5910895.785662318
```

```
In [215]: mon.Q
```

```
Out[215]: -490329.45241958584
```

5.6.2 Short circuits

Three phase at MV side POI

```
In [4]: data['shunts'] = [ # three phase fault to ground
                         {"bus": "POImv" , "R":1.0e-8, "X": 0.0, "bus_nodes": [1,2]},
                         {"bus": "POImv" , "R":1.0e-8, "X": 0.0, "bus_nodes": [2,3]},
                         {"bus": "POImv" , "R":1.0e-8, "X": 0.0, "bus_nodes": [3,0]},
                         ]

        # powers to zero:
        data['grid_feeders'] = [{"bus": "W1lv","bus_nodes": [1, 2, 3],"kW": 0, "kvar": 0},
                                {"bus": "W2lv","bus_nodes": [1, 2, 3],"kW": 0, "kvar": 0},
                                {"bus": "W3lv","bus_nodes": [1, 2, 3],"kW": 0, "kvar": 0},
                                {"bus": "POImv","bus_nodes": [1, 2, 3],"kW":0, "kvar": 0}] # STATCOM

        grid_1 = grid()
        grid_1.read(data)
        grid_1.pf()

        p=plot_results(grid_1)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
In [ ]: I_cc = grid_1.transformers[0]['i_1a_m']
        print('Three phase short circuit current at POImv = {:0.2f} kA'.format(I_cc/1000))
```

```
Three phase short circuit current at POImv = 2.28 kA
```

Phase-ground W1mv bus

```
In [ ]: data['shunts'] = [
                         {"bus": "W1mv" , "R": 1.0e-8, "X": 0.0, "bus_nodes": [1,0]},
                         ]
        grid_1 = grid()
        grid_1.read(data)  # Load data
        grid_1.pf()

        p=plot_results(grid_1)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
In [ ]: I_cc = grid_1.transformers[0]['i_1a_m']
        print('Phase-ground short circuit current at W1mv = {:0.2f} kA'.format(I_cc/1000))
```

```
Phase-ground short circuit current at W1mv = 0.62 kA
```

Get POI voltage with generators reactive powers

```
In [ ]: from scipy import optimize as sopt
```

```
data['shunts'] = []

V_ref = 1.0
p_gen = 2.0e3
q_gen = 0

data['grid_feeders'][0]['kW'] = p_gen
data['grid_feeders'][1]['kW'] = p_gen
data['grid_feeders'][2]['kW'] = p_gen

grid_1 = grid()
grid_1.read(data)  # Load data
grid_1.pf()

def residual(x):

    q_gen = x
    data['grid_feeders'][0]['kvar'] = q_gen
    data['grid_feeders'][1]['kvar'] = q_gen
    data['grid_feeders'][2]['kvar'] = q_gen
    grid_1.read(data)  # Load data
    grid_1.pf()  # solve power flow

    V = abs(grid_1.res['POI'].v_ag)/66.0e3*np.sqrt(3)

    return V_ref - V


res = sopt.bisect(residual,-3000.0,3000.0)
res

p=plot_results(grid_1)
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

Optimization with reactive powers (without STATCOM)

```
In [ ]: V_ref = 1.0
        p_gen = 3.0e3
        q_gen = 0

        data['grid_feeders'][0]['kW'] = p_gen
        data['grid_feeders'][1]['kW'] = p_gen
        data['grid_feeders'][2]['kW'] = p_gen

        grid_1 = grid()
        grid_1.read(data)  # Load data
        grid_1.pf()


        def obj(x):

            data['grid_feeders'][0]['kvar'] = x[0]
            data['grid_feeders'][1]['kvar'] = x[1]
            data['grid_feeders'][2]['kvar'] = x[2]
            grid_1.read(data)  # Load data
```

```
        grid_1.pf()  # solve power flow
        mon = grid_1.monitor(bus_from='POI', bus_to='GRID')

        P_loss = p_gen*3*1000 - mon.P

        return P_loss


    res = sopt.minimize(obj,[0,0,0], method='SLSQP')
    print(res)

    p=plot_results(grid_1)
```

```
     fun: 192203.64692081884
     jac: array([0.875, 0.875, 1.375])
 message: 'Optimization terminated successfully.'
    nfev: 61
     nit: 8
    njev: 8
  status: 0
 success: True
       x: array([496.46556547, 496.46556547, 466.3287252 ])
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
In [ ]: mon = grid_1.monitor(bus_from='POI', bus_to='GRID')
        print('POI active power = {:0.3f} MW'.format(mon.P/1e6))
        print('POI reactive power = {:0.2f} Mvar'.format(mon.Q/1e6))
```

```
POI active power = 8.808 MW
POI reactive power = 0.32 Mvar
```

Optimization with reactive powers (with STATCOM)

```
In [ ]: V_ref = 1.0
        p_gen = 3.0e3
        q_gen = 0

        data['grid_feeders'][0]['kW'] = p_gen
        data['grid_feeders'][1]['kW'] = p_gen
        data['grid_feeders'][2]['kW'] = p_gen

        grid_1 = grid()
        grid_1.read(data)  # Load data
        grid_1.pf()


        def obj(x):


            data['grid_feeders'][0]['kvar'] = x[0]
            data['grid_feeders'][1]['kvar'] = x[1]
            data['grid_feeders'][2]['kvar'] = x[2]
            data['grid_feeders'][3]['kvar'] = x[3]
            grid_1.read(data)  # Load data
            grid_1.pf()  # solve power flow
            mon = grid_1.monitor(bus_from='POI', bus_to='GRID')
```

```
        P_loss = p_gen*3*1000 - mon.P

        return P_loss

    def const1(x):
        return 1.02-abs(grid_1.monitor(bus_from='POI', bus_to='GRID').V_a)/38.105/1000

    def const2(x):
        return -(0.98-abs(grid_1.monitor(bus_from='POI', bus_to='GRID').V_a)/38.105/1000)


    res = sopt.minimize(obj,[0,0,0,0], method='COBYLA',
                        constraints=[{'type':'ineq','fun':const1},{'type':'ineq','fun':const2}]
                       )
    print(res)

    p=plot_results(grid_1)
```

```
     fun: 192924.1092131082
   maxcv: 0.0
 message: 'Optimization terminated successfully.'
    nfev: 644
  status: 1
 success: True
       x: array([201.34719703, 200.97303112, 200.48882721, 286.99544409])
```

Data type cannot be displayed: application/vnd.bokehjs_exec.v0+json, application/javascript

```
In [ ]: mon = grid_1.monitor(bus_from='POI', bus_to='GRID')
        print('POI active power = {:0.3f} MW'.format(mon.P/1e6))
        print('POI reactive power = {:0.2f} Mvar'.format(mon.Q/1e6))
```

```
POI active power = 8.807 MW
POI reactive power = -0.25 Mvar
```

Run this notebook in binder:

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Bibliography

[T1] Dugan, R. C., & Santoso, S. (2003). An example of 3-phase transformer modeling for distribution system analysis. 2003 IEEE PES Transmission and Distribution Conference and Exposition (IEEE Cat. No.03CH37495), 3, 1028–1032. https://doi.org/10.1109/TDC.2003.1335084