

SMART GRIDS: FROM TRADITIONAL TO MODERNIZED RESILIENT SYSTEMS

Víctor Escala García

Josep Fanals Batllori

Pol Heredia Julbe

Roger Izquierdo Toro

Palina Nicolas

SMART GRIDS

Master's degree in Energy Engineering

Master's degree in Electric Power Systems and Drives



**UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH**

**Escola Tècnica Superior d'Enginyeria
Industrial de Barcelona**

CONTENTS

1	Introduction	2
2	Phase 1	3
3	Code	5

1. INTRODUCTION

2. PHASE 1

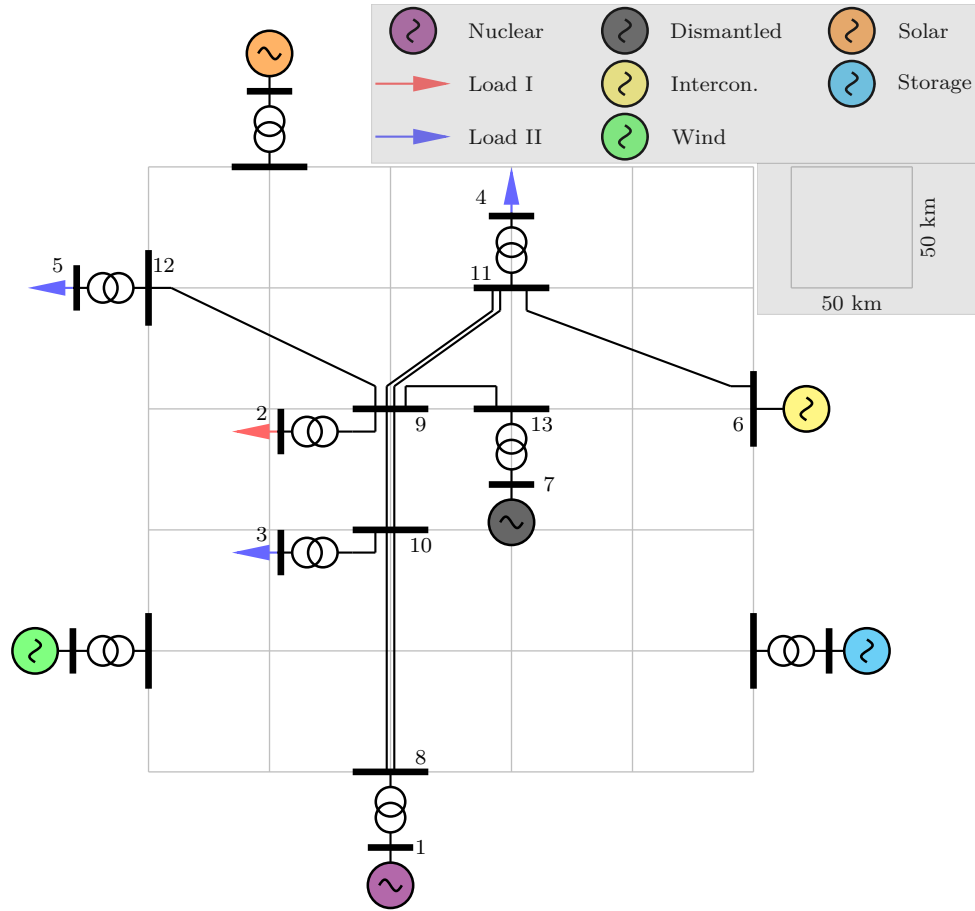


Figure 1. Overview of the network

Bus Hour	1	2	3	4	5	6	8	9	10	11	12
0	1.050	0.958	0.986	0.967	0.929	1.000	1.030	0.982	0.997	0.979	0.942
1	1.050	0.971	0.995	0.978	0.945	1.000	1.034	0.992	1.006	0.989	0.957
2	1.050	0.978	1.001	0.985	0.955	1.000	1.037	0.998	1.011	0.995	0.966
3	1.050	0.981	1.003	0.987	0.959	1.000	1.038	1.001	1.013	0.997	0.969
4	1.050	0.982	1.003	0.988	0.960	1.000	1.038	1.001	1.013	0.998	0.970
5	1.050	0.978	1.001	0.985	0.955	1.000	1.037	0.998	1.011	0.995	0.966
6	1.050	0.963	0.989	0.971	0.935	1.000	1.032	0.985	1.000	0.983	0.947
7	1.050	0.932	0.965	0.944	0.894	1.000	1.021	0.960	0.979	0.958	0.909
8	1.050	0.907	0.946	0.923	0.862	1.000	1.013	0.940	0.962	0.939	0.880
9	1.050	0.896	0.937	0.913	0.847	1.000	1.008	0.930	0.954	0.930	0.865
10	1.050	0.891	0.934	0.909	0.840	1.000	1.007	0.926	0.950	0.926	0.859
11	1.050	0.892	0.935	0.910	0.842	1.000	1.007	0.927	0.951	0.927	0.861
12	1.050	0.895	0.937	0.913	0.846	1.000	1.008	0.930	0.953	0.929	0.865
13	1.050	0.897	0.939	0.914	0.849	1.000	1.009	0.931	0.955	0.931	0.867
14	1.050	0.908	0.947	0.924	0.863	1.000	1.013	0.940	0.962	0.940	0.880
15	1.050	0.915	0.952	0.929	0.872	1.000	1.015	0.946	0.967	0.945	0.888
16	1.050	0.918	0.955	0.933	0.877	1.000	1.016	0.949	0.969	0.948	0.893
17	1.050	0.920	0.956	0.934	0.878	1.000	1.017	0.950	0.970	0.949	0.894
18	1.050	0.914	0.951	0.929	0.871	1.000	1.015	0.945	0.966	0.944	0.888
19	1.050	0.894	0.936	0.912	0.845	1.000	1.008	0.929	0.953	0.929	0.863
20	1.050	0.881	0.926	0.900	0.826	1.000	1.003	0.918	0.943	0.918	0.846
21	1.050	0.888	0.932	0.906	0.836	1.000	1.006	0.924	0.948	0.924	0.856
22	1.050	0.914	0.952	0.929	0.871	1.000	1.015	0.945	0.966	0.944	0.888
23	1.050	0.940	0.971	0.952	0.905	1.000	1.024	0.967	0.984	0.965	0.920

Table 1. Voltage profile, in pu, for 24 hours

Load Hour	8-10	10-9	9-11	9-12	11-6
0	26.455	19.983	9.608	31.041	64.359
1	24.390	18.423	9.046	28.520	59.872
2	23.081	17.437	8.701	26.936	57.069
3	22.536	17.027	8.559	26.280	55.912
4	22.445	16.958	8.536	26.171	55.719
5	23.074	17.431	8.699	26.928	57.053
6	25.751	19.450	9.415	30.178	62.821
7	30.722	23.232	10.829	36.361	73.837
8	34.303	25.990	11.904	40.967	82.183
9	35.951	27.271	12.414	43.140	86.055
10	36.582	27.764	12.611	43.983	87.541
11	36.445	27.657	12.568	43.800	87.220
12	36.002	27.311	12.430	43.209	86.176
13	35.732	27.101	12.346	42.850	85.540
14	34.196	25.907	11.872	40.827	81.932
15	33.261	25.184	11.587	39.611	79.743
16	32.725	24.771	11.425	38.918	78.490
17	32.543	24.631	11.370	38.685	78.066
18	33.360	25.261	11.617	39.739	79.975
19	36.132	27.413	12.470	43.382	86.482
20	37.988	28.867	13.054	45.886	90.862
21	36.992	28.085	12.739	44.534	88.508
22	33.316	25.226	11.604	39.681	79.870
23	29.399	22.221	10.443	34.693	70.872

Table 2. Percentual loading of the lines for a full day operation

3. CODE

```

1 import pandapower as pp
2 import pandas as pd
3 import numpy as np
4 import pandapower.control as control
5 import pandapower.networks as nw
6 import pandapower.timeseries as timeseries
7 from pandapower.timeseries.data_sources.frame_data import DFData
8 from pandapower.plotting import simple_plot
9
10 from line_param_calc import calc_line
11
12
13
14 pd.set_option('display.max_rows', 500)
15 pd.set_option('display.max_columns', 500)
16 pd.set_option('display.width', 1000)
17
18 def initialize_net(path_bus, path_geodata, path_line, path_demand, path_busload,
19                  path_generation, path_busgen, path_trafo):
20     """
21     initialize the grid from the .csv files
22
23     :param path_bus: path to the bus .csv file
24     :param geodata: path to the geodata .csv file
25     :param path_line: path to the line .csv file
26     :param path_demand: path to the normalized demand .csv file
27     :param busload: path to the bus-load look up table .csv file
28     :param path_generation: path to the normalized generation .csv file
29     :param busgen: path to the bus-generator look up table .csv file
30     :param trafo: path to the trafo .csv file
31     :return: the net class
32     """
33
34     def create_bus(path_bus, path_geodata):
35         """
36         adapts the data from the bus file (if needed)
37
38         :param path_bus:
39         :param path_geodata:
40         :return: the net with the buses added
41         """
42
43         df_bus = pd.read_csv(path_bus)
44         df_geodata = pd.read_csv(path_geodata)
45
46         net.bus = df_bus
47
48         # adapt geodata
49         for ll in range(len(df_geodata)):
50             indx_bus = pp.get_element_index(net, "bus", df_geodata['name'][ll])
51             df_geodata['name'][ll] = indx_bus
52
53         net.bus_geodata = df_geodata
54
55         return net
56
57     def create_line(path_line):
58         """
59         adapts the data from the line file
60
61         :param path_line:
62         :return: the net with the lines added
63         """
64
65         df_line = pd.read_csv(path_line)

```

```

66     for _, line in df_line.iterrows():
67         from_bus = pp.get_element_index(net, "bus", line.from_bus)
68         to_bus = pp.get_element_index(net, "bus", line.to_bus)
69
70         rr, xx, cc, imax = calc_line(line.a,
71                                     line.b,
72                                     line.c,
73                                     line.d,
74                                     line.e,
75                                     line.max_i,
76                                     int(line.parallel))
77
78         pp.create_line_from_parameters(net,
79                                     from_bus,
80                                     to_bus,
81                                     length_km=line.length,
82                                     r_ohm_per_km=rr,
83                                     x_ohm_per_km=xx,
84                                     c_nf_per_km=cc,
85                                     max_i_ka=imax,
86                                     name=line.name_l,
87                                     parallel=line.parallel)
88
89     return net
90
91
92 def create_load(path_demand, path_busload, path_bus):
93     """
94     adapts the load files
95
96     :param path_demand:
97     :param path_busload:
98     :param path_bus:
99     :return: the net with the loads added
100    """
101
102    df_demand = pd.read_csv(path_demand)
103    df_busload = pd.read_csv(path_busload)
104    df_bus = pd.read_csv(path_bus)
105
106    # create basic load dataframe
107    # find the bus index of each load
108    load_indx = []
109    for _, load in df_busload.iterrows():
110        bus_load = pp.get_element_index(net, "bus", load.bus)
111        load_indx.append(bus_load)
112
113    load_indx = pd.DataFrame(load_indx)
114    load_indx = load_indx.rename(columns={0: "bus"})
115
116    # load name and peak power
117    load_name = df_busload['bus']
118    load_pmw = df_busload['p_mw']
119    load_qmvar = df_busload['q_mvar']
120
121    # merge in a full dataframe
122    headers = ["name", "bus", "p_mw", "q_mvar"]
123    df_load = pd.concat([load_name, load_indx, load_pmw, load_qmvar], axis=1)
124    df_load.columns.values[0] = "name"
125
126    # create time series from the basic load df
127    Nt = len(df_demand)
128    Nl = len(df_load)
129    pmw_ts = np.zeros((Nt, Nl), dtype=float)
130    qmvar_ts = np.zeros((Nt, Nl), dtype=float)
131    for i in range(Nt): # number of time periods
132        pmw_ts[i,:] = df_load['p_mw'][:] * df_demand['norm'][i]

```

```

133         qmvar_ts[i,:] = df_load['q_mvar'][:] * df_demand['norm'][i]
134
135     # form loads as a static picture (initial time)
136     for ll in range(len(df_busload)):
137         pp.create_load(net, bus=load_indx['bus'][ll], p_mw=pmw_ts[0, ll], q_mvar=
qmvar_ts[0, ll], name=load_name[ll], index=int(ll))
138
139     # timeseries
140     df_pload_ts = pd.DataFrame(pmw_ts, index=list(range(Nt)), columns=net.load.index)
141     df_qload_ts = pd.DataFrame(qmvar_ts, index=list(range(Nt)), columns=net.load.index
)
142     ds_pload_ts = DFData(df_pload_ts)
143     ds_qload_ts = DFData(df_qload_ts)
144     const_load = control.ConstControl(net, element='load', element_index=net.load.
index, variable='p_mw', data_source=ds_pload_ts, profile_name=net.load.index)
145     const_load = control.ConstControl(net, element='load', element_index=net.load.
index, variable='q_mvar', data_source=ds_qload_ts, profile_name=net.load.index) # add
the reactive like this?
146
147     return net
148
149
150 def create_generator(path_generation, path_busgen, path_bus):
151     """
152     adapts the generation files
153
154     :param path_generation:
155     :param path_busgenerator:
156     :param path_bus:
157     :return: the net with the generators added
158     """
159
160     df_generation = pd.read_csv(path_generation)
161     df_busgen = pd.read_csv(path_busgen)
162     df_bus = pd.read_csv(path_bus)
163
164     # create basic generator dataframe
165     # find the bus index of each gen
166     gen_indx = []
167     for _, gen in df_busgen.iterrows():
168         bus_gen = pp.get_element_index(net, "bus", gen.bus)
169         gen_indx.append(bus_gen)
170
171     gen_indx = pd.DataFrame(gen_indx)
172     gen_indx = gen_indx.rename(columns={0: "bus"})
173
174     # load name and peak power
175     gen_name = df_busgen['bus']
176     gen_pmw = df_busgen['p_mw']
177     gen_vpu = df_busgen['vm_pu']
178
179     # merge in a full dataframe
180     headers = ["name", "bus", "p_mw", "vm_pu"]
181     df_gen = pd.concat([gen_name, gen_indx, gen_pmw, gen_vpu], axis=1)
182     df_gen.columns.values[0] = "name"
183
184     # create time series from the basic load df
185     Nt = len(df_generation)
186     Ng = len(df_gen)
187     pmw_ts = np.zeros((Nt, Ng), dtype=float)
188     for i in range(Nt): # number of time periods
189         pmw_ts[i,:] = df_gen['p_mw'][:] * df_generation['norm'][i]
190
191     # gen structure for 1 t
192     for ll in range(len(df_busgen)):
193         pp.create_gen(net, bus=gen_indx['bus'][ll], p_mw=pmw_ts[0, ll], vm_pu=gen_vpu[
ll], name=gen_name[ll], index=int(ll))

```



```

194
195
196     # timeseries
197     df_gen_ts = pd.DataFrame(pmw_ts, index=list(range(Nt)), columns=net.gen.index)
198     ds_gen_ts = DFData(df_gen_ts)
199     const_gen = control.ConstControl(net, element='gen', element_index=net.gen.index,
variable='p_mw', data_source=ds_gen_ts, profile_name=net.gen.index)
200
201     return net
202
203
204 def create_intercon(path_bus):
205     """
206     defines the interconnection (slack bus)
207
208     :param path_bus:
209     :return: the net with the interconnection added
210     """
211
212     df_bus = pd.read_csv(path_bus)
213
214     # find the slack index
215     slack_indx = 0
216     for ll in range(len(df_bus)):
217         # slack_indx = pp.get_element_index(net, "bus", bb.name)
218         if df_bus['name'][ll] == 'intercon':
219             slack_indx = pp.get_element_index(net, "bus", df_bus['name'][ll])
220
221     pp.create_ext_grid(net, slack_indx, vm_pu=1.0, va_degree=0)
222
223     return net
224
225
226 def create_trafo(path_trafo):
227     """
228     defines the transformers
229
230     :param path_trafo:
231     :return: the net with the transformers added
232     """
233
234     df_trafo = pd.read_csv(path_trafo)
235
236     # for trafo in df_trafo:
237     for _, trafo in df_trafo.iterrows():
238         hv_bus = pp.get_element_index(net, "bus", trafo.hv_bus)
239         lv_bus = pp.get_element_index(net, "bus", trafo.lv_bus)
240
241         pp.create_transformer_from_parameters(net,
242                                             hv_bus,
243                                             lv_bus,
244                                             trafo.sn_mva,
245                                             trafo.vn_hv_kv,
246                                             trafo.vn_lv_kv,
247                                             trafo.vkr_percent,
248                                             trafo.vk_percent,
249                                             trafo.pfe_kw,
250                                             trafo.io_percent)
251
252     return net
253
254
255
256
257
258
259 # create empty network

```

```

260     net = pp.create_empty_network()
261
262     # buses
263     net = create_bus(path_bus, path_geodata)
264
265     # lines
266     net = create_line(path_line)
267
268     # loads
269     net = create_load(path_demand, path_busload, path_bus)
270
271     # gens
272     net = create_generator(path_generation, path_busgen, path_bus)
273
274     # interconnection
275     net = create_intercon(path_bus)
276
277     # trafos
278     net = create_trafo(path_trafo)
279
280
281
282     return net
283
284
285 if __name__ == "__main__":
286     # load paths
287     path_bus = 'Datafiles/bus1.csv'
288     path_geodata = 'Datafiles/geodata1.csv'
289     path_line = 'Datafiles/line1.csv'
290     path_demand = 'Datafiles/demand1.csv'
291     path_busload = 'Datafiles/bus_load1.csv'
292     path_generation = 'Datafiles/generation1.csv'
293     path_busgen = 'Datafiles/bus_gen1.csv'
294     path_trafo = 'Datafiles/trafo1.csv'
295
296     # define net
297     net = initialize_net(path_bus, path_geodata, path_line, path_demand, path_busload,
298                          path_generation, path_busgen, path_trafo)
299
300     # run timeseries
301     ow = timeseries.OutputWriter(net, output_path="./Results/", output_file_type=".xlsx")
302     ow.log_variable('res_bus', 'vm_pu')
303     ow.log_variable('res_line', 'loading_percent')
304     timeseries.run_timeseries(net)
305
306     # run diagnostic
307     # pp.diagnostic(net)
308     print(net.bus)
309
310     # plot
311     # pp.plotting.simple_plot(net)
312     # simple_plot(net)

```

Listing 3.1. Main code in Python with the Pandapower library