04_pandapower

October 10, 2023

pandapower – A Short Introduction

This tutorial explains **panda**power library basis used in RHT laboratories. **panda**power is an easy to use tool for loadflow and short-circuit calculations in power systems. To go further, we recommend you take a look at these two - **panda**power's documentation. - **panda**power's tutorials. - **panda**power_heig_ui's documentation.

1 Create a small power network

We consider the following simple 3-bus example network from **panda**power's tutorial.

The above network can be created in pandapower as follows:

[1]:
```
import pandapower as pp
```

1.1 Create a pandapower empty power network object

In order to create an empty network object, we can run the following **panda**power command:

[2]:
```
# Create an empty network
net = pp.create_empty_network()
net
```

[2]:
```
This pandapower network is empty
```

The empty network object is composed by a dictionary of pandas DataFrame.

1.2 Create three buses with different voltage levels

We need three buses with different voltage levels and names, buses are the elements which connect equipments together:

[3]:
```
# Create buses
bus1 = pp.create_bus(net, vn_kv=20., name="Bus 1")
bus2 = pp.create_bus(net, vn_kv=0.4, name="Bus 2")
bus3 = pp.create_bus(net, vn_kv=0.4, name="Bus 3")
```

Remark: We need to pay attention that voltage levels are express in kV and correspond to line voltages.

1.3 Creating a transformer and connecting it to the network

For a full understanding of the parameters to be applied, please read the documentation carefully.

In order to create the transformer object that will be connection to the network previously created, we can proceed as follows:

[4]:
```
# Create transformer
trafo = pp.create_transformer_from_parameters(
    net, hv_bus=bus1, lv_bus=bus2, sn_mva=0.4, vn_hv_kv=20.0, vn_lv_kv=0.4,␣
 ↪vk_percent=6.0, vkr_percent=1.425, pfe_kw=1.35, i0_percent=0.3375,␣
 ↪name="Trafo")

# trafo = pp.create_transformer(net, hv_bus=bus1, lv_bus=bus2, std_type="0.4␣
 ↪MVA 20/0.4 kV", name="Trafo")
```

Remark: pay attention in parameters units and in voltage levels matchs.

1.3.1 Create a transmission line and connect it to the network

[5]:
```
line = pp.create_line_from_parameters(net, from_bus=bus2, to_bus=bus3,␣
 ↪length_km=0.1, r_ohm_per_km=0.642, x_ohm_per_km=0.083, c_nf_per_km=210,␣
 ↪max_i_ka=0.142, name="Line")
```

1.3.2 Create a load connect it to the network

[6]:
```
# Create bus elements
load = pp.create_load(net, bus=bus3, p_mw=0.100, q_mvar=0.05, name="Load")
```

1.3.3 Create an external grid connection

This element is mandatory to be able to perform powerflow simulations. It insures to keep powers balanced within the power network:

[7]:
```
ext_grid = pp.create_ext_grid(net, bus=bus1, vm_pu=1.20, name="Grid Connection")
```

2 Data structure and data access

A **panda**power network object is structured as a dictionary:

- Keys are the type names of power network equipments names such as line, load transformer, etc. (string).
- Values are tables which contains all the information needed about their corresponding equipments (pandas DataFrame).

By calling the network have a quick overview of it and the number of element for each equipment.

[8]:
```
net
```

[8]:
```
This pandapower network includes the following parameter tables:
   - bus (3 element)
   - load (1 elements)
   - ext_grid (1 elements)
   - line (1 elements)
   - trafo (1 elements)
```

There are two ways to get the one equipment type table:

- By using the dictionary way to call values.
- By using the pandapower object.

[9]:
```python
net["bus"]
```

[9]:
```
    name  vn_kv type  zone  in_service
0  Bus 1   20.0    b  None        True
1  Bus 2    0.4    b  None        True
2  Bus 3    0.4    b  None        True
```

[10]:
```python
type(net["bus"])
```

[10]:
```
pandas.core.frame.DataFrame
```

[11]:
```python
net.bus
```

[11]:
```
    name  vn_kv type  zone  in_service
0  Bus 1   20.0    b  None        True
1  Bus 2    0.4    b  None        True
2  Bus 3    0.4    b  None        True
```

[12]:
```python
net.trafo
```

[12]:
```
    name std_type  hv_bus  lv_bus  sn_mva  vn_hv_kv  vn_lv_kv  vk_percent  \
0  Trafo     None       0       1     0.4      20.0       0.4         6.0

   vkr_percent  pfe_kw  i0_percent  shift_degree tap_side  tap_neutral  \
0        1.425    1.35      0.3375           0.0     None          NaN

   tap_min  tap_max  tap_step_percent  tap_step_degree  tap_pos  \
0      NaN      NaN               NaN              NaN      NaN

   tap_phase_shifter  parallel   df  in_service
0              False         1  1.0        True
```

[13]:
```python
net.line
```

[13]:
```
   name std_type  from_bus  to_bus  length_km  r_ohm_per_km  x_ohm_per_km  \
0  Line     None         1       2        0.1         0.642         0.083
```

```
   c_nf_per_km  g_us_per_km  max_i_ka   df  parallel  type  in_service
0        210.0          0.0     0.142  1.0         1  None        True
```

[14]:
```
net.load
```

[14]:
```
   name  bus  p_mw  q_mvar  const_z_percent  const_i_percent  sn_mva  scaling  \
0  Load    2   0.1    0.05              0.0              0.0     NaN      1.0

   in_service type
0        True  wye
```

To have access to one specific element or value of a table, use Pandas functions:

[15]:
```
net.bus.loc[0, :]
```

[15]:
```
name          Bus 1
vn_kv          20.0
type              b
zone           None
in_service     True
Name: 0, dtype: object
```

[16]:
```
type(net.bus.loc[0, :])
```

[16]:
```
pandas.core.series.Series
```

[17]:
```
net.bus.at[0, "name"]
```

[17]:
```
'Bus 1'
```

We can also modify the data using Pandas function:

[18]:
```
net.bus.loc[0, "name"] = "hv_bus"
net.bus
```

[18]:
```
     name  vn_kv type  zone  in_service
0  hv_bus   20.0    b  None        True
1   Bus 2    0.4    b  None        True
2   Bus 3    0.4    b  None        True
```

3 Run power flow

Now we can run a balanced power flow calculation using the following command:

[19]:
```
pp.runpp(net)
net
```

[19]:
```
This pandapower network includes the following parameter tables:
   - bus (3 element)
   - load (1 elements)
   - ext_grid (1 elements)
   - line (1 elements)
   - trafo (1 elements)
 and the following results tables:
   - res_bus (3 element)
   - res_line (1 elements)
   - res_trafo (1 elements)
   - res_ext_grid (1 elements)
   - res_load (1 elements)
```

Then if you check you **panda**power object you will see that powerflow results tables have been added.

It may also be interesting to consult the results for buses, lines and transformers:

[20]:
```
net.res_bus
```

[20]:
```
      vm_pu  va_degree      p_mw    q_mvar
0  1.200000   0.000000 -0.106038 -0.051875
1  1.190635  -0.539841  0.000000  0.000000
2  1.153532   0.080701  0.100000  0.050000
```

[21]:
```
net.res_line
```

[21]:
```
   p_from_mw  q_from_mvar  p_to_mw  q_to_mvar     pl_mw   ql_mvar  i_from_ka  \
0   0.103769     0.050486     -0.1      -0.05  0.003769  0.000486   0.139895

    i_to_ka      i_ka  vm_from_pu  va_from_degree  vm_to_pu  va_to_degree  \
0  0.139896  0.139896    1.190635       -0.539841  1.153532      0.080701

   loading_percent
0        98.518141
```

[22]:
```
net.res_trafo
```

[22]:
```
    p_hv_mw  q_hv_mvar   p_lv_mw  q_lv_mvar     pl_mw   ql_mvar  i_hv_ka  \
0  0.106038   0.051875 -0.103769  -0.050486  0.002268  0.001389  0.00284

    i_lv_ka  vm_hv_pu  va_hv_degree  vm_lv_pu  va_lv_degree  loading_percent
0  0.139895       1.2           0.0  1.190635     -0.539841        24.593091
```

All other pandapower elements and power grid analysis functionality (e.g. optimal power flow, state estimation or short-circuit calculation) are also fully integrated into pandapower's tabular data structure. This concludes a short walkthrough of some pandapower features. More in-depth tutorials can be found under this link

4 Create small power network using pandapower_heig_ui package

A package has been created in order to simplify network generation, timeseries simulation and data visualisation. We can generate **panda**power object from data stored in excels files through the following function. We advice you to take a look at its documentation.

[23]:
```
import pp_heig_plot as pp_plot
import pp_heig_simulation as pp_sim
from datetime import time
```

[24]:
```
net_file_path = "data/3_bus_example.xlsx"
net = pp_sim.load_net_from_xlsx(file_path=net_file_path)
net
```

[24]:
```
This pandapower network includes the following parameter tables:
   - bus (3 element)
   - load (1 elements)
   - ext_grid (1 elements)
   - line (1 elements)
   - trafo (1 elements)
```

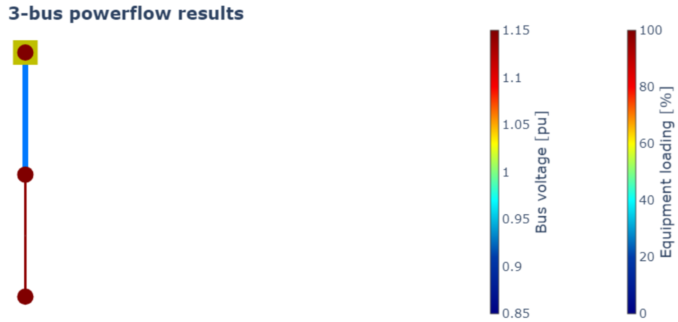
We can plot a simplified diagram of our network using the following function:

- By adding a filename, the plot will be saved in a png format in the default folder *plot*.
- We can change the folder name using the folder parameter.
- We can view the equipment parameters in the plot by moving the mouse over them.
- The network is well traced when it is tree-like. In the case of a mesh grid, a coordinate parameter must be added to the buses.

[25]:
```
pp_plot.plot_power_network(net=net, plot_title="3-bus example",␣
↪filename="3_bus_example")
```

We can run a simple power flow and visualise result using the following functions:

[26]:
```python
pp.runpp(net)
pp_plot.plot_powerflow_result(net=net, plot_title="3-bus powerflow results",␣
↪filename="3_bus_pp_result")
net.res_bus
```



[26]:

	vm_pu	va_degree	p_mw	q_mvar
0	1.200000	0.000000	-0.106038	-0.051875
1	1.190635	-0.539841	0.000000	0.000000
2	1.153532	0.080701	0.100000	0.050000

4.1 Timeseries powerflow simulation

We can create power profiles from excel files to perform timeseries powerflow simulations. After having been loaded, the resulting object is a dictionary of dataframe:

- Keys is the equipment name where profile are related to.
- Values can be active and reactive power profile table.

[27]:
```python
profile_file_path = "data/3_bus_power_profile.xlsx"
time_series = pp_sim.load_power_profile_form_xlsx(file_path=profile_file_path)
print(time_series.keys())
print(time_series["load"].keys())
time_series["load"]["p_mw"]
```

```
dict_keys(['load'])
dict_keys(['p_mw', 'q_mvar'])
```

[27]:

profile	0	1
00:00:00	0.02301	0.08414
01:00:00	0.01743	0.08866
02:00:00	0.01592	0.08950
03:00:00	0.02022	0.08509
04:00:00	0.03131	0.07463
05:00:00	0.03377	0.07337
06:00:00	0.03829	0.06886
07:00:00	0.05299	0.05567

```
08:00:00  0.07359  0.04019
09:00:00  0.08708  0.03242
10:00:00  0.08454  0.03552
11:00:00  0.08326  0.03750
12:00:00  0.07909  0.04142
13:00:00  0.06846  0.04953
14:00:00  0.06324  0.05306
15:00:00  0.06635  0.05080
16:00:00  0.05867  0.05915
17:00:00  0.05251  0.06481
18:00:00  0.04749  0.06822
19:00:00  0.04147  0.06954
20:00:00  0.03622  0.07252
21:00:00  0.03267  0.07635
22:00:00  0.02918  0.07906
23:00:00  0.02701  0.08083
```

In this example, the file loaded contains two different profiles for loads. If we take a look in the load **panda**power table we can see that the `profile_mapping` parameter of the load is set to 0. It means that power profiles applied to this load will be the 0.

[28]:
```
net.load
```

[28]:
```
     name  bus  p_mw  q_mvar  scaling  const_z_percent  const_i_percent  \
0  load_0    2   0.1    0.05      1.0              0.0              0.0

  sn_mva  in_service type  profile_mapping
0   None        True  wye                0
```

[29]:
```
pp_sim.apply_power_profile(net=net, equipment="load",␣
 ↪power_profiles=time_series["load"])
```

Then we need to create an output writer which will store simulation results:

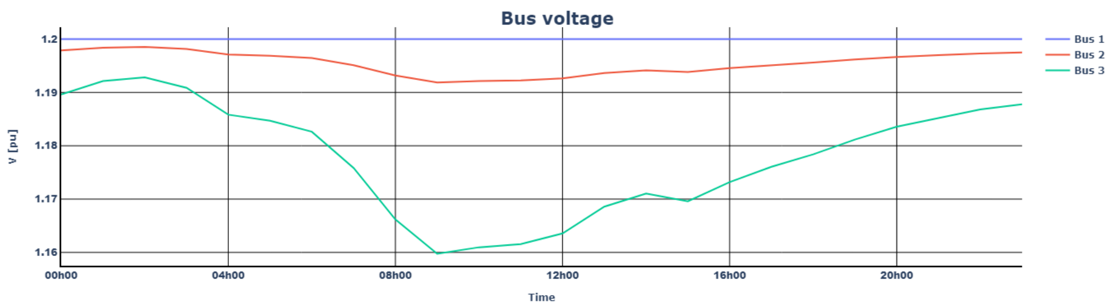
- Default results stored are `res_bus.vm_pu`, `res_line.loading_percent`, `res_trafo.loading_percent`.
- We can add other results using `add_results` parameters.

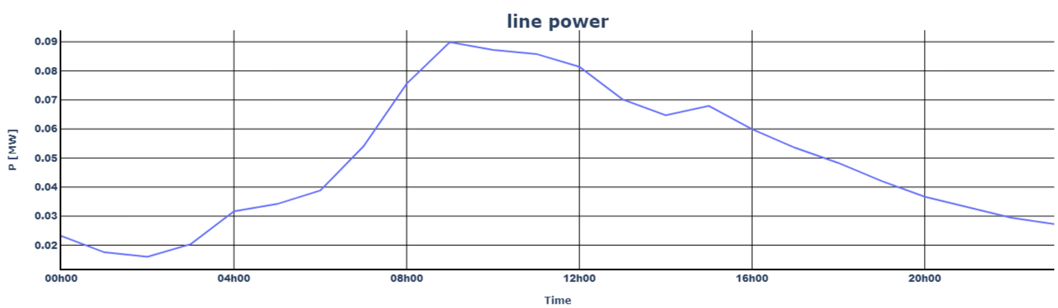
[30]:
```
pp_sim.create_output_writer(net=net, add_results= ["res_line.p_from_mw"])
```

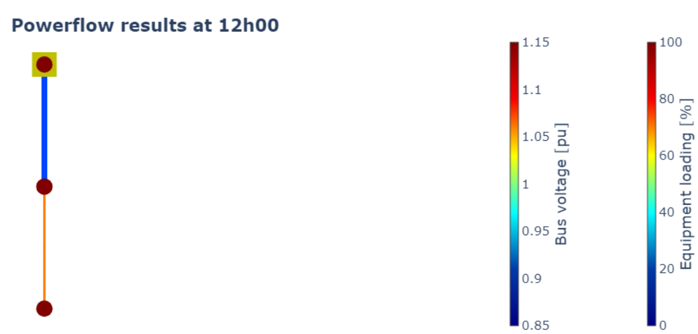
Finally, we can run times series simulation and plot results – as follows:

[31]:
```
result_df = pp_sim.run_time_simulation(net=net)
print()
pp_plot.plot_timeseries_result(data_df=result_df["res_bus.vm_pu"], ylabel="V␣
 ↪[pu]",
                               plot_title="Bus voltage", filename= "voltage_result")
print()
```

```
pp_plot.plot_timeseries_result(data_df=result_df["res_line.p_from_mw"],␣
 ↪ylabel="P [MW]",
                              plot_title="line power", filename= "line_result")
print()
pp_plot.plot_timestamps_powerflow_result(net=net, filename="net_result_12h",␣
 ↪plot_time=time(hour=12))
```

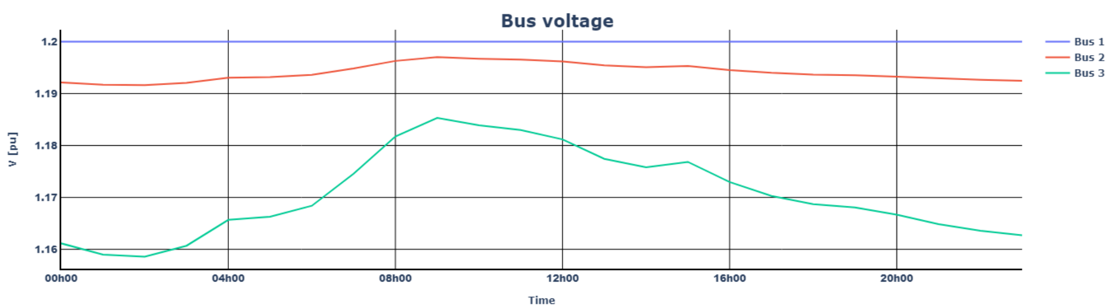


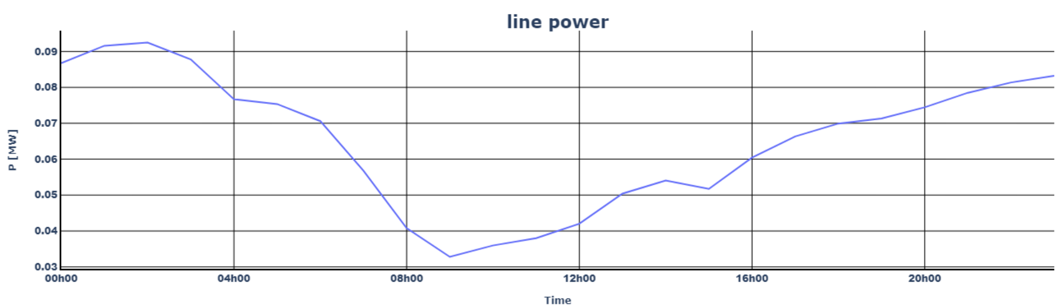




We can use the second power profile loaded for the excel file. To do this, we just need to modify the `profile_mapping` parameter before applying once again the power profile:

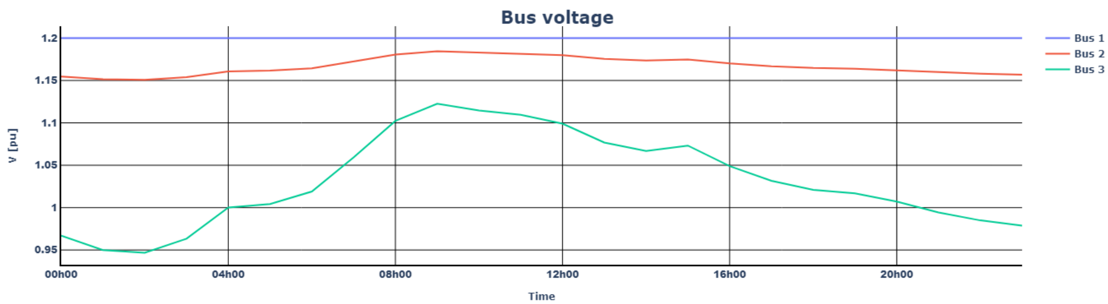
[32]:
```
net.load.loc[0, "profile_mapping"] = 1
pp_sim.apply_power_profile(net=net, equipment="load",␣
 ↪power_profiles=time_series["load"])
result_df = pp_sim.run_time_simulation(net=net)
print()
pp_plot.plot_timeseries_result(data_df=result_df["res_bus.vm_pu"], ylabel="V␣
 ↪[pu]",
                        plot_title="Bus voltage", filename= "voltage_result")
print()
pp_plot.plot_timeseries_result(data_df=result_df["res_line.p_from_mw"],␣
 ↪ylabel="P [MW]",
                        plot_title="line power", filename= "line_result")
```





We can also scale our power profiles modifying `scaling` parameters:

[33]:
```
net.load.loc[0, "scaling"] = 5
result_df = pp_sim.run_time_simulation(net=net)
pp_plot.plot_timeseries_result(data_df=result_df["res_bus.vm_pu"], ylabel="V␣
↪[pu]",
                            plot_title="Bus voltage", filename= "voltage_result")
print()
pp_plot.plot_timeseries_result(data_df=result_df["res_line.p_from_mw"],␣
↪ylabel="P [MW]",
                            plot_title="line power", filename= "line_result")
```



5 References

- Pandapower 'Getting started'
- Pandapower's documentation
- Pandapower's tutorials on GitHub

5.1 Citing pandapower

```
@article{pandapower.2018,
author={L. Thurner and A. Scheidler and F. Schafer and J. H. Menke and J. Dollichon and F. Meie
journal={IEEE Transactions on Power Systems},
title={pandapower - an Open Source Python Tool for Convenient Modeling, Analysis and Optimizati
year={2018},
doi={10.1109/TPWRS.2018.2829021},
url={https://arxiv.org/abs/1709.06743},
ISSN={0885-8950}
}
```