MATPOWER / PANDAPOWER

Principal differences:

Pandapower	Matpower
Loading percentage	☒
☒	Differenciates P/Q buses Generation/Load
Branch data power (from/to) one node (sum of everything in one node)	Branch data power indicates from bus #/to bus #
☒	Min/Max Voltage magnitude, angle, P losses, Q losses
Impedance magnitudes/unit lenght	Impedances in p.u.
kV, kW, kVAr,	kV, MW, MVAr

In Pandapower the SLACK bus needs to be implemented as the external grid.

"External grids represent the higher level power grid connection and are modelled as the slack bus in the power flow calculation." – *Pandapower documentation*

Also in the external grid we can add the 0 degree reference, whereas in the bus design (if we try to implement a slack bus as a regular kind of bus) it's not possible to do so.

From matpower to pandapower:

$$Y = G + jwC = G + jB$$
$$Z = R + jwL = R + jX$$

Matpower (p.u.) ⟶ **Pandapower (units/km)**

Base values

$$\boldsymbol{Z_b} = \frac{V_b^2}{S_b} = \frac{230[kV]^2}{100[MVA]} = 529\ \Omega$$

$$r = r_{p.u.} \cdot Z_b$$

$$x = x_{p.u.} \cdot jZ_b$$

$$b = \frac{b_{p.u.}}{Z_b} = 2 \cdot \pi \cdot f \cdot C$$

Input values pandapower

$$r\left[\frac{\Omega}{\text{km}}\right], \text{x}\left[\frac{\Omega}{\text{km}}\right], C\left[\frac{nF}{km}\right]$$

ejemplo1.m / ejemplo1.py

	r	x	b
line 1-2	5.33232	26.6616	616.76
line 1-3	3.93576	19.6788	466.63
line 2-4	3.93576	19.6788	466.63
line 3-4	6.7288	33.6444	767.19

Matpower:

```
>> makeYbus(ejemplo1)
ans =

  (1,1)    8.9852 -44.8359i
  (2,1)   -3.8156 +19.0781i
  (3,1)   -5.1696 +25.8478i
  (1,2)   -3.8156 +19.0781i
  (2,2)    8.9852 -44.8359i
  (4,2)   -5.1696 +25.8478i
  (1,3)   -5.1696 +25.8478i
  (3,3)    8.1933 -40.8638i
  (4,3)   -3.0237 +15.1185i
  (2,4)   -5.1696 +25.8478i
  (3,4)   -3.0237 +15.1185i
  (4,4)    8.1933 -40.8638i
```

```
Ybus =

   8.9852 -44.7459i  -3.8156 +19.0781i  -5.1696 +25.8478i   0.0000 + 0.0000i
  -3.8156 +19.0781i   8.9852 -44.7459i   0.0000 + 0.0000i  -5.1696 +25.8478i
  -5.1696 +25.8478i   0.0000 + 0.0000i   8.1933 -40.7613i  -3.0237 +15.1185i
   0.0000 + 0.0000i  -5.1696 +25.8478i  -3.0237 +15.1185i   8.1933 -40.7613i

>> runpf('ejemplo1')
```

Pandapower:

```
>>>net._ppc

   Ybus= net._ppc["internal"]["Ybus"].todense()

   Ybus/100
```

	0	1	2	3
0	(8.98519043680334-44.835927755798174j)	(-3.815628815628816+19.07814407814408j)	(-5.169561621174524+25.847808105872623j)	0j
1	(-3.815628815628816+19.07814407814408j)	(8.98519043680334-44.835927755798174j)	0j	(-5.169561621174524+25.84780810587262
2	(-5.169561621174524+25.847808105872623j)	0j	(8.193234291335287-40.863826773836294j)	(-3.0236726701607632+15.118543095939362
3	0j	(-5.169561621174524+25.847808105872623j)	(-3.0236726701607632+15.118543095939362j)	(8.193234291335287-40.863826773836294

METHODOLOGY / STEPS TO RUN A GRID

Import libraries:

```
import pandas as pd
import os  from pathlib import Path
import pandapower as pp  from pandapower
from pandapower import plotting                                    #not clear if necessary
```

1. **Import the excel** file and make it readable for any working directory. **Import json file** for the loads

```
file = 'Sitel_Invade_MV_Topology.xlsx'
xl = pd.ExcelFile(Path(str(os.getcwd()) + '/'+ file))
dfload = pd.read_json(Path(str(os.getcwd()) +'/ofpfs_sent.json', orient='rows'))
```

2. Create an **empty network**.

```
net = pp.create_empty_network()
```

3. Generate a **Data Frame** with all the **line data** available in the excel file.

```
MVNetwork = xl.parse('Linies')
```

4. **Assign** the appropriate data to the variables **X, C, R**.

```
x_ohm_per_km = MVNetwork['Reactancia_ohm_km'][0]
c_nf_per_km = MVNetwork['Capacitat_uF_km'][0]
r_ohm_per_km = MVNetwork['Resistencia_ohm_km'][0]
```

5. If the **line type** is **not defined** as a standard type, **create a new standard type** for your own line data.

```
line_data = {"c_nf_per_km": c_nf_per_km , "r_ohm_per_km": r_ohm_per_km, "x_ohm_per_km": x_ohm_per_km,
"max_i_ka":0.415}
pp.create_std_type(net, line_data, "line_ESTABANELL", element='line')
```

6. Generate a **Data Frame** with all the **busses** in the excel file (also those at the LV part of the TRAFOS, which can be found in the excel TRAFOS page.

```
MVNetworkbusses = xl.parse('Busses')
MVNetworkbussesTrafos = xl.parse('Trafos')
```

7. **Create the busses** (stored in the previous data frames) with its **voltage**, **origin-end**, **name** and indicating the **network where they belong** to. We can do this by iterating in the data frame ['name'] created with the excel file.

```
for i in MVNetworkbusses['name']:
        pp.create_bus(net, vn_kv=20.5,name=i, max_vm_pu=1.1, min_vm_pu=0.9)
for i in range(len(MVNetworkbussesTrafos['name'])):
        pp.create_bus(net, vn_kv= MVNetworkbussesTrafos['vn_kv'][i], name=MVNetworkbussesTrafos['name'][i],
max_vm_pu=1.1, min_vm_pu=0.9)
```

8. The **busses data frame can be edited** and we could include any other columns we may need, for example an identifier for the busses (all, including LV).

Previously, in order to get the IDs both from the joints at MV and the knots at the LV part of the transformers, we'll create a common data frame that will later become a list to be able to add it in the data bus data frame as an additional column:

```
l=MVNetworkbusses['Id Bus']                          #Data Frame
r=l.append(MVNetworkbussesTrafos['Id Bus'])          #r still a Data Frame
r=r.values.tolist()                                  #r list
```

We add in the net.bus (a pandas.core.frame.DataFrame type) a column for the Id Bus. Then, we can set the Id Bus column as the index of the net.bus and decide whether or not we keep the Id Bus column to be able to access to it*.

*In a data frame we cannot access the index as a number. We first need to indicate which column we want to pick and then access to the [i] item of that column.

```
net.bus.insert(5, 'Id Bus', r)
net.bus = net.bus.set_index('Id Bus', drop = False)
```

9. The **identifiers** used for the **busses** are:
   a. Transformer nodes: the one given
   b. Other nodes: 200-226 (own generated code to identify the non-transformers busses)
   c. Both identifiers in the "Busses" shit in the excel file.
10. **Create trafos** between the MV and LV busses with the parameters provided by Estabanell and also the standard type most similar to our voltages and power.

    ```
    pp.create_transformer_from_parameters(net, hv_bus=202, lv_bus=75, sn_mva=0.4, vn_hv_kv=20.5, vn_lv_kv=0.23, vk_percent=4., vkr_percent=0.04, pfe_kw=0.75, i0_percent=1.8, shift_degree=150, in_service=True, parallel=1, name='E.T. NODE 1', tap_side='hv', tap_neutral=0, tap_min=-2, tap_max=2, tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=False, df=1, index=75)
    ```

    Pay attention to the units, as may change depending on the program version.

11. **Create the lines** by selecting the **start and end node** and also its **standard type** already defined. Moreover, at this point, the length and the name of the line can be set.
    a. **Length:** MVNetwork['Length_km'][i]
    b. **Name:** bus origin
    c. **Line identifier:** Id TedisNet

    ```
    for i in range(len(MVNetwork['Id TedisNet'])):
        pp.create_line(net, from_bus=MVNetworkbusses['Id Bus'][i], to_bus=MVNetworkbusse ['Id Bus'][i+1],
        length_km=MVNetwork['Length_km'][i], std_type="line_ESTABANELL", name= MVNetwork['origen'] [i])
    ```

Add the line identifier but without making it become the line index:

```
net.line.insert(14, 'Id linia', MVNetwork['Id TedisNet'])
```

12. We need to define an **external grid**, which will take the place of **one of the existing busses**, making it **become the Slack bus**.

    ```
    pp.create_ext_grid(net, bus=201, vm_pu=1.0, va_degree=0.0, in_service=True, name="Grid Connection")
    ```

13. **Create loads** from the json file first iteration:

To understand the structure of the json file and where the inputs, outputs and simulation are, we will create different data frames.

```
dfload_output0= dfload['output'][0]
dfload_output0_input= dfload['output'][0]['input']
dfload_output0_simulation= dfload['output'][0]['simulation']
dfload_output0_timestamp= dfload['output'][0]['timestamp']
```

We will create an empty list where we will append the IDs from the busses that have a load [1]. The loads then will be created from the data in the first iteration input [2]. Also, free busses will be assigned a 0 power load [3].

```
[1]     l=[]
[2]     for i in range(len(dfload['output'][0]['input'])):
                pp.create_load(net, bus=dfload['output'][0]['input'][i]['IdTedisNet'], p_mw = (dfload['output'][0]
                ['input'][i]['ActivePower'])/1000,  q_mvar = (dfload['output'][0]['input'][i]['ReactivePower'])/1000)
                l.append(dfload['output'][0]['input'][i]['IdTedisNet'])
[3]     for i in net.bus['Id Bus']:
                if i not in l:
                        pp.create_load(net, bus=i, p_mw=0)
```

Pay attention to units again, as you may be given loads in “kilo” and pandapower request them in “Mega”.

14. **Running the net:**

There are many algorithms to solve power flow problems. The following algorithms are available:

“nr” Newton-Raphson (pypower implementation with numba accelerations)

“iwamoto_nr” Newton-Raphson with Iwamoto multiplier (maybe slower than NR but more robust)

“bfsw” backward/forward sweep (specially suited for radial and weakly-meshed networks)

“gs” gauss-seidel (pypower implementation)

“fdbx” fast-decoupled (pypower implementation)

“fdxb” fast-decoupled (pypower implementation)

The algorithm chosen is **bfsw**, out grid is radial and weakly meshed.

```
pp.runpp(net, algorithm='bfsw', calculate_voltage_angles=(False), init="flat", tolerance_mva=1e-8, trafo_model='t',
trafo_loading="current")  #not necessary
```

15. **Plotting the net:**

To plot the network built, the next steps need to be followed:

```
[1] conda install -c conda-forge python-igraph
        import igraph                                   #not sure if necessary
[3] pp.plotting.simple_plot(net)   necessary!
```

16. **Congestion in the grid:**

We will add the bus origin to the data frame of line results and make it its index column for both line results data frame and line data frame.

```
net.res_line.insert (14, 'from_bus', net.line['from_bus'])
net.res_line = net.res_line.set_index('from_bus', drop = False)

net.line = net.line.set_index("from_bus", drop = True)
```

Moreover, a dictionary will be initialized to store the overloaded lines, which will be outlined by a maximum load percentage (modified as wished). The parameters kept will be the loading percentage of the line, the origin bus name and the bus ID:

```
lines_overloaded={}
for i in net.res_line['from_bus']:
    if net.res_line['loading_percent'][i]>2:        → [2 it’s just an example]
        a=net.bus['name'][i]
        b=net.bus['Id Bus'][i]
        c=net.res_line['loading_percent'][i]
        lines_overloaded[(net.line['Id linia'][i])]=(a,b,c)
```

Convert the dictionary in a data frame:

```
df_lines_overloaded = pd.DataFrame.from_dict(lines_overloaded, orient='index')
```

We need to overwrite the variable when renaming it. If we don't do so, we won't be able to refer / access it.

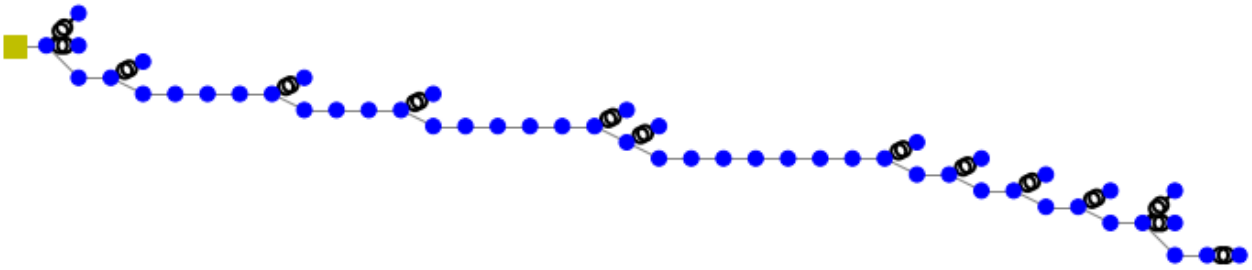
```
df_lines_overloaded=df_lines_overloaded.rename({0:'From Bus', 1:'Id Bus', 2:'overload_percentatge'}, axis='columns')
```

17. **Create excel file:** from the data frame obtained we will provide Estabanell with an excel file only containing the origin bus and the ID of the overloaded line. Thus, we will exclude the Bus ID.

```
df_lines_overloaded_ESTABANELL=df_lines_overloaded

del(df_lines_overloaded_ESTABANELL['Id Bus'])

df_lines_overloaded.to_excel(Path(str(os.getcwd()) +'/PowerFlow.xlsx'))
```



file:///Users/mariagris/CITCEA/ofpf_invade_citcea/ofpfs_sent.json

GRID FUNCTION

```
def grid():
    net = pp.create_empty_network()
    file = 'SiteI_Invade_MV_Topology.xlsx'
    xl = pd.ExcelFile(Path(str(os.getcwd()) + '/'+ file))
    MVNetwork = xl.parse('Linies')
    x_ohm_per_km = MVNetwork['Reactancia_ohm_km'][0]
    c_nf_per_km = MVNetwork['Capacitat_nF_km'][0]
    r_ohm_per_km = MVNetwork['Resistencia_ohm_km'][0]
    line_data = {"c_nf_per_km": c_nf_per_km , "r_ohm_per_km": r_ohm_per_km, "x_ohm_per_km": x_ohm_per_km, "max_i_ka":0.415}
    pp.create_std_type(net, line_data, "line_ESTABANELL", element='line')
    MVNetworkbusses = xl.parse('Busses')

    for i in MVNetworkbusses['name']:
        pp.create_bus(net, vn_kv=20.5,name=i, max_vm_pu=1.1, min_vm_pu=0.9)

    MVNetworkbussesTrafos = xl.parse('Trafos')

    for i in range(len(MVNetworkbussesTrafos['name'])):
            pp.create_bus(net, vn_kv= MVNetworkbussesTrafos['vn_lv_kv'][i], name=MVNetworkbussesTrafos['name'][i], max_vm_pu=1.1,
            min_vm_pu=0.9)
    l=MVNetworkbusses['Id Bus']
    r=l.append(MVNetworkbussesTrafos['Id Bus'])
    r=r.values.tolist()
    net.bus.insert(5, 'Id Bus', r)
    net.bus = net.bus.set_index("Id Bus", drop = False)

    pp.create_transformer_from_parameters(net, hv_bus=202, lv_bus=75, sn_mva=0.4, vn_hv_kv=20.5, vn_lv_kv=0.23, vk_percent=4., vkr_percent=0.04, pfe_kw=0.75,
    i0_percent=1.8, shift_degree=150, in_service=True, parallel=1, name='E.T. NODE 1', tap_side='hv', tap_pos=0, tap_neutral=0, tap_min=-2, tap_max=2,
    tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=False, df=1, index=75)
    pp.create_transformer_from_parameters(net, hv_bus=202, lv_bus=76, sn_mva=0.63, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=4.09, vkr_percent=0.0409,
    pfe_kw=0.548, i0_percent=0.9, shift_degree=150, in_service=True, parallel=1, name='E.T. NODE 2', tap_side='hv',tap_pos=0, tap_neutral=0, tap_min=-2, tap_max=2,
    tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=True, df=1, index=76)
    pp.create_transformer_from_parameters(net, hv_bus=204, lv_bus=22, sn_mva=1, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=6., vkr_percent=0.08, pfe_kw=1.4,
    i0_percent=1.3, shift_degree=150, in_service=True, parallel=1, name='E.T.CUARTEL', tap_side='hv', tap_pos=0,tap_neutral=0, tap_min=-2, tap_max=2,
    tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=True,df=1, index=22)
    pp.create_transformer_from_parameters(net, hv_bus=209, lv_bus=26, sn_mva=0.63, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=4., vkr_percent=0.053, pfe_kw=1.03,
    i0_percent=1.6, shift_degree=150, in_service=True, parallel=1, name='E.T.PEDRALS', tap_side='hv',tap_pos=0, tap_neutral=0, tap_min=-2, tap_max=2,
    tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=True,df=1, index=26)
    pp.create_transformer_from_parameters(net, hv_bus=213, lv_bus=30, sn_mva=0.8, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=6., vkr_percent=0.06, pfe_kw=1.4,
    i0_percent=1.3, shift_degree=150, in_service=True, parallel=1, name='E.T. LA MUTUA',tap_side='hv', tap_pos=0,tap_neutral=0, tap_min=-2, tap_max=2,
    tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=True, df=1,index=30)
    pp.create_transformer_from_parameters(net, hv_bus=219, lv_bus=34, sn_mva=0.63, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=4.28, vkr_percent=0.0428,
    pfe_kw=1.3, i0_percent=1.05, shift_degree=150, in_service=True, parallel=1, name='E.T. LA LLEÓ', tap_side='hv',tap_pos=0, tap_neutral=0, tap_min=-2, tap_max=2,
    tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=True,df=1, index=34)
    pp.create_transformer_from_parameters(net, hv_bus=220, lv_bus=64, sn_mva=0.63, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=4., vkr_percent=0.04, pfe_kw=1.3,
    i0_percent=1.6, shift_degree=150, in_service=True, parallel=1, name='E.T. PRADES 1',tap_side='hv',tap_pos=0, tap_neutral=0, tap_min=-2, tap_max=2,
    tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=True,df=1, index=64)
    pp.create_transformer_from_parameters(net, hv_bus=228, lv_bus=42, sn_mva=0.4, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=4.05, vkr_percent=0.054,
    pfe_kw=0.768, i0_percent=1.72, shift_degree=150, in_service=True, parallel=1, name='E.T. ECUADOR', tap_side='hv', tap_pos=0, tap_neutral=0, tap_min=-2,
    tap_max=2, tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=True,df=1, index=42)
    pp.create_transformer_from_parameters(net, hv_bus=230, lv_bus=46, sn_mva=0.4, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=5., vkr_percent=0.066, pfe_kw=1.193,
    i0_percent=0.45, shift_degree=150, in_service=True, parallel=1, name='E.T. URUGUAI', tap_side='hv',tap_pos=0, tap_neutral=0, tap_min=-2, tap_max=2,
    tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=True, df=1, index=46)
    pp.create_transformer_from_parameters(net, hv_bus=232, lv_bus=50, sn_mva=0.63, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=4., vkr_percent=0.053, pfe_kw=1.3,
    i0_percent=1.6, shift_degree=150, in_service=True, parallel=1, name='E.T. LA TORRETA', tap_side='hv',tap_pos=0, tap_neutral=0, tap_min=-2,
    tap_max=2,tap_step_percent=2.5,tap_step_degree=0,tap_phase_shifter=True,df=1, index=50)
    pp.create_transformer_from_parameters(net, hv_bus=234, lv_bus=54, sn_mva=0.4, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=4., vkr_percent=0.04, pfe_kw=0.93,
    i0_percent=1.8, shift_degree=150, in_service=True, parallel=1,name='E.T. COSTA BRAVA 1', tap_side='hv',tap_pos=0, tap_neutral=0, tap_min=-
    2,tap_max=2,tap_step_percent=2.5,tap_step_degree=0,tap_phase_shifter=True, df=1, index=54)
    pp.create_transformer_from_parameters(net, hv_bus=236, lv_bus=58, sn_mva=0.4, vn_hv_kv=20.5, vn_lv_kv=0.23, vk_percent=4., vkr_percent=0.04, pfe_kw=0.93,
    i0_percent=1.8, shift_degree=150,in_service=True,parallel=1,name='E.T. NOVA VERDAGUER 1',tap_side='hv',tap_pos=0, tap_neutral=0,tap_min=-
    2,tap_max=2,tap_step_percent=2.5,tap_step_degree=0,tap_phase_shifter=True, df=1, index=58)
    pp.create_transformer_from_parameters(net, hv_bus=236, lv_bus=66, sn_mva=0.4, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=4., vkr_percent=0.04, pfe_kw=0.75,
    i0_percent=1.8, shift_degree=150, in_service=True, parallel=1,name='E.T. NOVA VERDAGUER 2',tap_side='hv',tap_pos=0, tap_neutral=0,tap_min=-
    2,tap_max=2,tap_step_percent=2.5,tap_step_degree=0,tap_phase_shifter=True, df=1,index=66)
    pp.create_transformer_from_parameters(net, hv_bus=238, lv_bus=61, sn_mva=0.25, vn_hv_kv=20.5, vn_lv_kv=0.4, vk_percent=4., vkr_percent=0.04, pfe_kw=0.65,
    i0_percent=2, shift_degree=150, in_service=True, parallel=1, name='E.T. GRANADA', tap_side='hv', tap_pos=0, tap_neutral=0, tap_min=-2, tap_max=2,
    tap_step_percent=2.5, tap_step_degree=0, tap_phase_shifter=True, df=1, index=61)

    for i in range(len(MVNetwork['Id TedisNet'])):
            pp.create_line(net, from_bus=MVNetworkbusses['Id Bus'][i], to_bus=MVNetworkbusses['Id Bus'][i+1],
            length_km=MVNetwork['Length_km'][i], std_type="line_ESTABANELL", name=MVNetwork['origen'][i])

    net.line.insert(14, 'Id linia', MVNetwork['Id TedisNet'])
    pp.create_ext_grid(net, bus=201, vm_pu=1.0, va_degree=0.0, in_service=True, name="Grid Connection")
    return(net)
```

ITERATIONS

```
for i in dfload['output']:
        power=i['input']
        net=grid()
        l=[]
        for e in power:
                pp.create_load(net, bus=e['IdTedisNet'], p_mw = (e['ActivePower'])/1000000, q_mvar = (e['ReactivePower'])/1000000)
                l.append(e['IdTedisNet'])
        for n in net.bus['Id Bus']:
                if n not in l:
                        pp.create_load(net, bus=n, p_mw=0, q_mvar=0)
        pp.runpp(net, algorithm='bfsw')
        pp.plotting.simple_plot(net)
        net.res_line.insert(14, 'from_bus', net.line['from_bus'])
        net.res_line = net.res_line.set_index('from_bus', drop = False)
        net.line = net.line.set_index("from_bus", drop = True)
        lines_overloaded={}
        for t in net.res_line['from_bus']:
                if net.res_line['loading_percent'][t]>2.85:
                        a=net.bus['name'][t]
                        b=net.bus['Id Bus'][t]
                        c=net.res_line['loading_percent'][t]
                        lines_overloaded[(net.line['Id linia'][t])]=(a,b,c)
        df_lines_overloaded=pd.DataFrame.from_dict(lines_overloaded, orient='index')
        df_lines_overloaded=df_lines_overloaded.rename({0:'From Bus', 1:'Id Bus', 2:'overload_percentatge'}, axis='columns')
        df_lines_overloaded_ESTABANELL=df_lines_overloaded

        del(df_lines_overloaded_ESTABANELL['Id Bus'])
        print('Data i hora actuals:',pd.datetime.now())
        print('Timestamp:',i['timestamp'])
        print(df_lines_overloaded_ESTABANELL)
        df_lines_overloaded.to_excel(Path(str(os.getcwd()) +'/PowerFlow.xlsx'))
```

JSON STRUCTURE


```
{
 _updated: "1970-01-01T00:00:00Z",
 timestamp: "2019-03-27T17:41:29Z",
 output:
 [
      {
       timestamp: "2019-03-27T13:00:00Z",
       input:
           [
                {
                   IdTedisNet: 34,
                   ActivePower: 234001.772057958,
                   ReactivePower: 0
                },
     simulation:
     [
          {
               Current: 12068.5584330606,
               Percentage: 34.9813287914801,
               Id: 5
```


(e) in the iteration: for e in i['input']

(i) in the iteration: for i in dfload ['output']