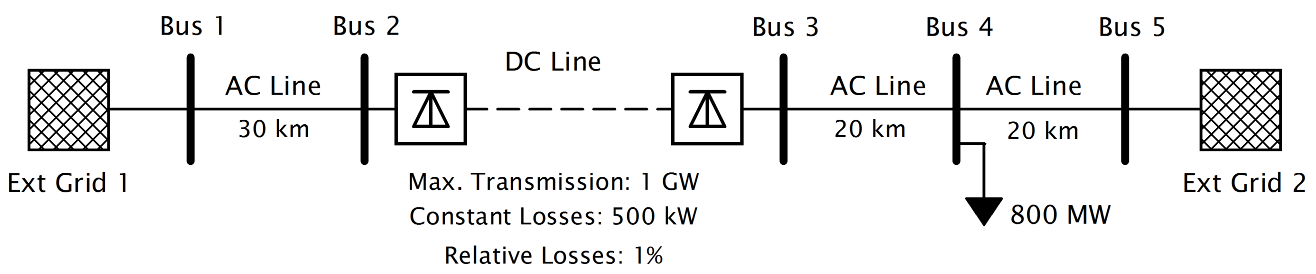
DC Line dispatch with `pandapower` Optimal Power Flow (OPF)

This is an introduction to the usage of the `pandapower` **optimal power flow** with DC lines. This example is adapted from the `pandapower` tutorial for OPF. The installation of `pandapower` is included at the bottom of this script.

Norway-Denmark Skagerrak Example Network

Let us imagine a load (industry) in Norway close to the Skagerrak DC line (Norway-Denmark)



Import and create a network in `pandapower`

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

In [2]:
```python
net = pp.create_empty_network()
```

First, create 5 buses. In `pandapower` these are just empty nodes that we have to fill with something, like a generator, a load, or an external grid.

In [3]:
```python
b1 = pp.create_bus(net, vn_kv=420) # High voltage grid at 420 kV
b2 = pp.create_bus(net, vn_kv=420)
b3 = pp.create_bus(net, vn_kv=420)
b4 = pp.create_bus(net, vn_kv=420)
b5 = pp.create_bus(net, vn_kv=420)
```

Now we connect the buses 1 to 2, 3 to 4, and 4 to 5 with AC lines of a standard type. Other types are found here.

In [4]:
```python
l1 = pp.create_line(net, from_bus=b1, to_bus=b2, length_km=30, std_type='490-AL1/64-ST1A 380.0', name='
l2 = pp.create_line(net, from_bus=b3, to_bus=b4, length_km=20, std_type='490-AL1/64-ST1A 380.0', name='
l3 = pp.create_line(net, from_bus=b4, to_bus=b5, length_km=20, std_type='490-AL1/64-ST1A 380.0', name='
```

Add a DC line between bus 2 and bus 3. This is a bit different since the length of the line is not needed, only the *transmission losses in percentange* (`loss_percent`) and the *conversion losses from AC to DC* (`loss_mw`).

In [5]:
```python
dcl1 = pp.create_dcline(net, from_bus=b2, to_bus=b3,     # between bus 2 and bus 3
                        p_mw=200, max_p_mw=1700,         # consider the maximum power 1700 MW
                        loss_percent=1.0, loss_mw=0.5,
                        vm_from_pu=1., vm_to_pu=1.,
#                        in_service=True, length_km=240,
                        name='DC Line Skagerrak')
```

Add the *external grids* (`ext_grid`) of Denmark and Norway at bus 1 and bus 5 respectively

In [6]:
```python
eg1 = pp.create_ext_grid(net, bus=b1, min_p_mw=0., name='Denmark') # min_p_mw=0. is important t prevent
eg2 = pp.create_ext_grid(net, bus=b5, min_p_mw=0., name='Norway')
```

In [7]:
```python
net.ext_grid
```

Out[7]:

	name	bus	vm_pu	va_degree	slack_weight	in_service	min_p_mw
0	Denmark	0	1.0	0.0	1.0	True	0.0
1	Norway	4	1.0	0.0	1.0	True	0.0

Create a load of **800 MW** in Norway.

In [8]:
```
load = pp.create_load(net, bus=b4, p_mw=800, controllable=False)
```

We now run a regular **load/power flow** to check out the DC line model (`runpp`):

In [9]:
```
pp.runpp(net)
```

The transmission power of the DC line is defined in the loadf low as given by the `p_kw` parameter, which was set to **200 MW**:

Let us inspect the results using `res_` * and then what we are interested, like `line`, `dcline`, `gen`, `ext_grid`, ...

In [10]:
```
net.res_line
```

Out[10]:

	p_from_mw	q_from_mvar	p_to_mw	q_to_mvar	pl_mw	ql_mvar	i_from_ka	i_to_ka	i_ka	vm_from_pu
0	200.424094	-54.924399	-200.000000	38.455133	0.424094	-16.469265	0.285670	0.279965	0.285670	1.000000
1	197.500000	38.778749	-197.225603	-49.762407	0.274397	-10.983658	0.276676	0.280339	0.280339	1.000000
2	-602.774397	49.762407	605.238497	-51.356319	2.464100	-1.593912	0.833581	0.834977	0.834977	0.997406

In [11]:
```
net.res_dcline
```

Out[11]:

	p_from_mw	q_from_mvar	p_to_mw	q_to_mvar	pl_mw	vm_from_pu	va_from_degree	vm_to_pu	va_to_degree
0	200.0	-38.455133	-197.5	-38.778749	2.5	1.0	-0.520428	1.0	-0.706556

The losses amount to 2.5 MW, which are made up of 0.5 MW conversion loss and 200 MW * 0.01 = 2 MW transmission losses.

Optimal Power Flow (OPF) problem

Now we define costs for the external grids to run an *Optimal Power Flow* problem. Check `create_poly_cost()` for further details.

In [12]:
```
cost_eg_Denmark = pp.create_poly_cost(net, element=0, et='ext_grid', cp1_eur_per_mw=10) # Higher cost o
cost_eg_Norway = pp.create_poly_cost(net, element=1, et='ext_grid', cp1_eur_per_mw=8)  # Lower cost of

# Here we do a little trick and increase the line limits
net.line['max_loading_percent'] = 1000
```

Run the OPF (`runopp(net, delta=1e-16)`)

In [13]:
```
pp.runopp(net, delta=1e-16)
```

This function runs an Optimal Power Flow using the PYPOWER OPF. To make sure that the PYPOWER OPF converges, we decrease the power tolerance `delta` (the default value is `delta=1e-10`). The power tolerance `delta` measures the extent to which exceeding minimum and maximum power limits are tolerated. In the above case, the limits considered by the OPF for the generators are `min_p_mw - delta` and `max_p_mw + delta` as lower and upper bounds respectively on the active power.

Since we defined lower costs for the external grid of Norway, it fully services the load:

In [14]:
```
net.res_ext_grid
```

Out[14]:

	p_mw	q_mvar
0	0.500047	-9.14392
1	804.327721	-2.83435

While the DC line does not transmit any power:

In [15]:
```
net.res_dcline
```

Out[15]:

	p_from_mw	q_from_mvar	p_to_mw	q_to_mvar	pl_mw	vm_from_pu	va_from_degree	vm_to_pu	va_to_degree
0	0.500045	9.143817	-0.000044	2.80257	0.5	0.999995	-0.001233	0.994886	-1.32896

If we set the costs of the left grid to a lower value than the right grid and run the loadflow again:

In [16]:
```
net.poly_cost.cp1_eur_per_mw.at[cost_eg_Denmark] = 8  # Lower cost of generating power in Denmark
net.poly_cost.cp1_eur_per_mw.at[cost_eg_Norway] = 10  # Higher cost of generating power in Norway

# Run OPF again
pp.runopp(net, delta=1e-16)
```

We can see that the power now comes from the left ext_grid:

In [17]:
```
net.res_ext_grid
```

Out[17]:

	p_mw	q_mvar
0	819.561312	-9.143921
1	0.000199	-21.281278

And is transmitted over the DC line:

In [18]:
```
net.res_dcline
```

Out[18]:

	p_from_mw	q_from_mvar	p_to_mw	q_to_mvar	pl_mw	vm_from_pu	va_from_degree	vm_to_pu	va_to_degree
0	812.821658	-19.895032	-804.278869	-15.161111	8.542789	0.992403	-2.036341	1.006132	1.299392

We can however see that the lines on the left hand side are now overloaded:

In [19]:
```
net.res_line.loading_percent
```

Out[19]:
```
0    117.361953
1    114.511333
2      3.047309
Name: loading_percent, dtype: float64
```

If we set the maximum line loading to 100% and run the OPF again:

In [20]:
```
net.line["max_loading_percent"] = 100

# Run OPF again
pp.runopp(net, delta=1e-16)
```

We can see that the lines are no longer overloaded:

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

Out[21]:
```
0    100.000081
1     97.843676
2     16.980049
Name: loading_percent, dtype: float64
```

Because the load is serviced from both grids:

In [22]:
```
net.res_ext_grid
```

Out[22]:

	p_mw	q_mvar
0	698.362533	-1.127654
1	117.108630	-18.637188

And the DC line transmits only part of the power needed to service the load:

In [23]:
```
net.res_dcline
```

Out[23]:

	p_from_mw	q_from_mvar	p_to_mw	q_to_mvar	pl_mw	vm_from_pu	va_from_degree	vm_to_pu	va_to_degree
0	693.468206	-3.953104	-686.107135	-7.999877	7.361071	0.9931	-1.729241	1.004366	0.920507

Finally, we can also define transmission costs for the DC line:

In [24]:
```python
costeg1 = pp.create_poly_cost(net, 0, 'dcline', cp1_eur_per_mw=3)

# Run OPF again
pp.runopp(net, delta=1e-16)
```

Because the sum of the costs for generating power in Denmark (`bus1`) and transmitting it to Norway via Skagerrak (`DC line`) is now larger than for generating it in Norway (`bus5`), the OPF draws *as much power from Norway as is possible* without violating line loading constraints:

In [25]:
```python
net.res_line.loading_percent
```

Out[25]:
```
0     15.325863
1     15.141506
2    100.000062
Name: loading_percent, dtype: float64
```

In [26]:
```python
net.res_dcline
```

Out[26]:

	p_from_mw	q_from_mvar	p_to_mw	q_to_mvar	pl_mw	vm_from_pu	va_from_degree	vm_to_pu	va_to_degree
0	106.524719	8.635256	-104.97497	8.543515	1.54975	0.998941	-0.263174	0.996022	-0.976356

Finally, let us check the **costs** of this operation:

In [27]:
```python
net.res_cost
```

Out[27]:
```
8156.304437166348
```

Installation and others

To use `pandapower` it is advised to have `anaconda` / `miniconda` installed. Then either:

```
pip install pandapower
```

or

```
conda install pandapower
```

is enough.

This script was adapted from a `pandapower` tutorial for OPF for the *FYS377 Digital Power Systems*, by *Heidi S. Nygård*, NMBU. Adapted by Leonardo Rydin Gorjão. 2023.