9bus-example-simulation

April 22, 2024

Security-constrained simulation example

In this notebook, we will run two power system optimization problems for the 9-bus network:

Problem 1. An optimization example without N-1 security constraints Problem 2. An optimization example with N-1 security constraints

The notebook will produce results for both problems. A companion notebook provides analyses of the results and demonstrates why solving problem 2 is more beneficial than problem 1.

[ ]:
```
# necessary imports
import os
from datetime import datetime, timedelta
import pandas as pd
from typing import NamedTuple
import pypsa
```

Network description

We will first start with network attributes. These define buses, transmission lines, transformers, generators, as well as buses on the network.

The static network looks as follows

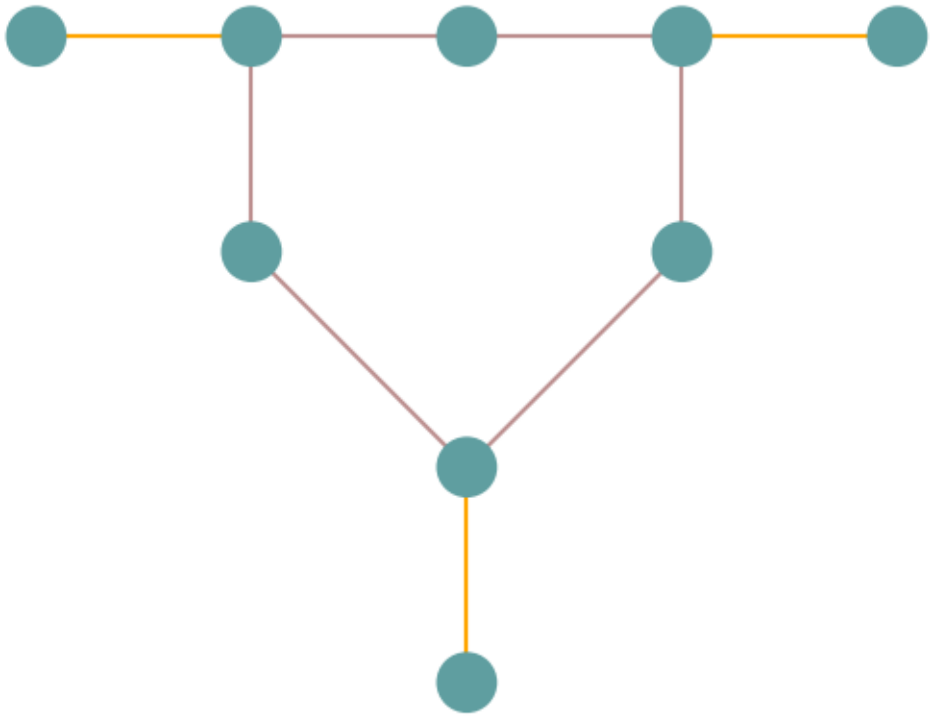
[ ]:
```
# Build network
network = pypsa.Network()

class Coordinate(NamedTuple):
    x: float
    y: float

coordinate_by_bus_label = {"1": Coordinate(x=0, y=0),
                           "2": Coordinate(x=-2, y=3),
                           "3": Coordinate(x=2, y=3),
                           "4": Coordinate(x=0, y=1),
                           "5": Coordinate(x=-1, y=2),
                           "6": Coordinate(x=1, y=2),
                           "7": Coordinate(x=-1, y=3),
                           "8": Coordinate(x=0, y=3),
                           "9": Coordinate(x=1, y=3)}
# Add buses
```

```python
for bus_label, coordinate in coordinate_by_bus_label.items():
    network.add("Bus", name=bus_label, x=coordinate.x, y=coordinate.y)


# Transformer data
lines_data = {
    "line_name": ["ln4-5", "ln6-4", "ln6-9", "ln5-7", "ln7-8", "ln8-9"],
    "bus0": ["4", "6", "6", "5", "7", "8"],
    "bus1": ["5", "4", "9", "7", "8", "9"],
    "r": [0.17, 0.039, 0.019, 0.039, 0.0085, 0.032],
    "x": [0.092, 0.17 , 0.1008 , 0.072  , 0.161, 0.085],
    "s_nom": [80, 45, 80, 80, 80, 80]
}

for i in range(len(lines_data["line_name"])):
    network.add("Line",
                lines_data["line_name"][i],
                bus0=lines_data["bus0"][i],
                bus1=lines_data["bus1"][i],
                r=lines_data["r"][i],
                x=lines_data["x"][i],
                s_nom=lines_data["s_nom"][i])


transformers_data = {
    "transformer_name": ["xf1-4", "xf2-7", "xf9-3"],
    "bus0": ["1", "2", "9"],
    "bus1": ["4", "7", "3"],
    "r": [0.001, 0.001, 0.001],
    "x": [0.0576, 0.0586 , 0.0625],
}

for i in range(len(transformers_data["transformer_name"])):
    network.add("Transformer",
                transformers_data["transformer_name"][i],
                bus0=transformers_data["bus0"][i],
                bus1=transformers_data["bus1"][i],
                r=transformers_data["r"][i],
                x=transformers_data["x"][i],
                model="pi",
                s_nom=1000)


# Add loads
network.add("Load", f"ld1", bus="5")
network.add("Load", f"ld2", bus="6")
```

```
# Add generators
network.add("Generator", f"Gn1", bus="1", control="slack", committable=True,␣
 ↪carrier="nuclear", p_nom=500.0, marginal_cost=10.0)
network.add("Generator", f"Gn2", bus="2", committable=True, carrier="gas", ␣
 ↪p_nom=200.0, marginal_cost=50.0)
network.add("Generator", f"Gn3", bus="3", committable=True, carrier="wind",␣
 ↪p_nom=200.0, marginal_cost=0.0)
network.plot()
```

```
WARNING:pypsa.plot:Cartopy needs to be installed to use `geomap=True`.
```

[ ]:
```
(<matplotlib.collections.PatchCollection at 0x1282cac10>,
 <matplotlib.collections.LineCollection at 0x1282cb490>,
 <matplotlib.collections.LineCollection at 0x1282c3ed0>)
```



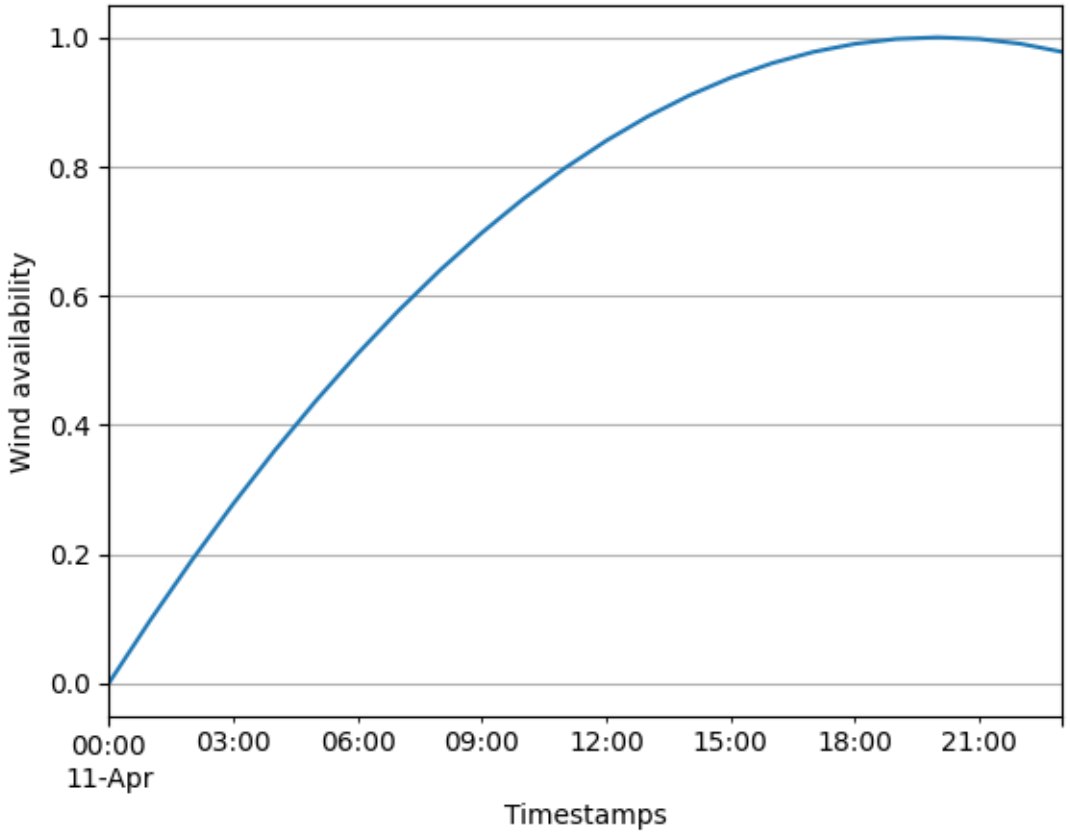
We will assume a wind unit is located on bus 3. There are two loads on buses 5 and 6.

The start timestamp of the simulation will be midnight April 11, 2024. We will consider 24-hour time period referred to as `snapshots`.

The wind profile provides the availability of wind over these 24-hour snapshots and looks as follows (assumes a peak wind power at hour 20)

```
[ ]: start_timestamp = datetime.fromisoformat("2024-04-11T00:00:00Z")
snapshots = [start_timestamp + timedelta(hours=h) for h in range(24)]
wind_profile= pd.Series({snapshot: 1 - ((snapshot.hour/20 - 1)**2) for snapshot␣
 ↪in snapshots})
ax = wind_profile.plot(grid=True)
ax.set_xlabel("Timestamps")
ax.set_ylabel("Wind availability")
```

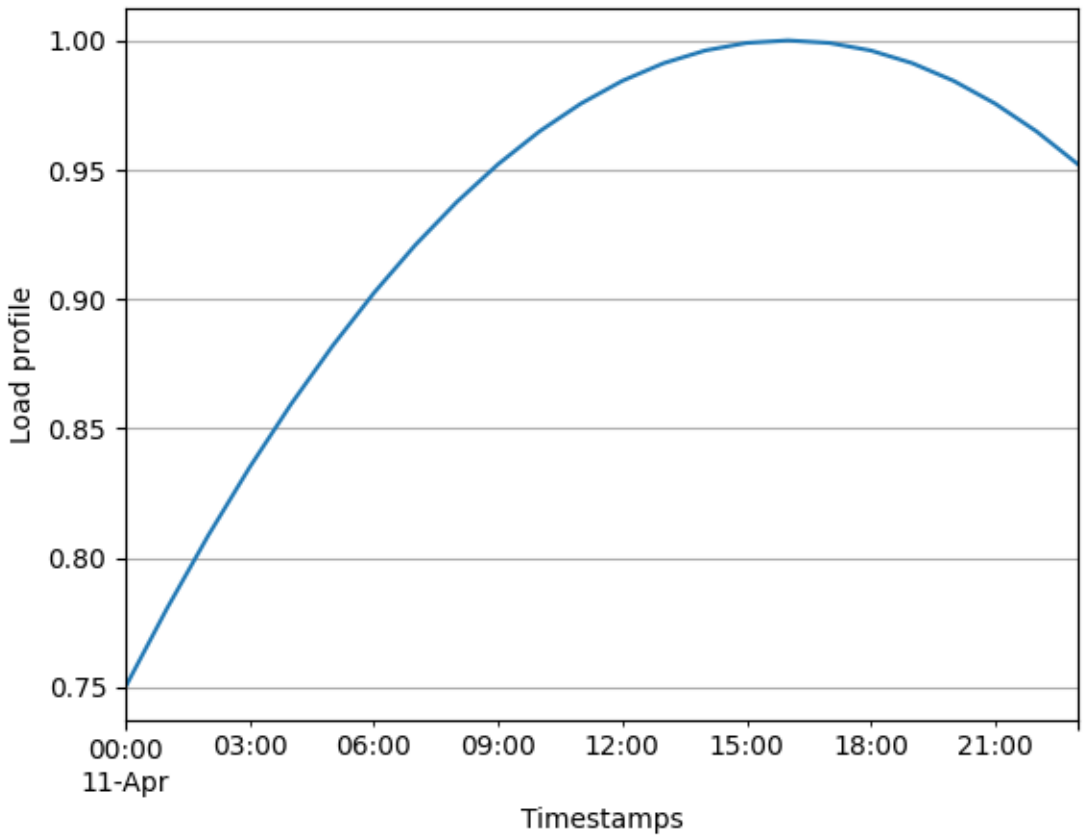
```
[ ]: Text(0, 0.5, 'Wind availability')
```



We will additionally assume a load profile placed on buses 5 and 6. The load profile is as follows (it assumes a peak load at hour 16):

```
[ ]: load_profile= pd.Series({snapshot: 1-(0.25 * (snapshot.hour/16 - 1)**2) for␣
 ↪snapshot in snapshots})
ax = load_profile.plot(grid=True)
ax.set_xlabel("Timestamps")
ax.set_ylabel("Load profile")
```

```
[ ]: Text(0, 0.5, 'Load profile')
```



[ ]:
```
# Update the network to accept wind and load profiles
network.set_snapshots(snapshots=snapshots)


# add load
dfm_load = pd.concat([50 * load_profile, 50 * load_profile], axis=1)
dfm_load.columns = ['ld1', 'ld2']
dfm_load.columns.name = 'Load'
dfm_load.index.name = 'snapshot'
network.loads_t['p_set'] = dfm_load


# add wind
dfm_wind = pd.DataFrame(wind_profile)
dfm_wind.columns = ['Gn3']
dfm_wind.columns.name = 'Generator'
dfm_wind.index.name = 'snapshot'
network.generators_t['p_max_pu'] = dfm_wind

network_gen_p_set = network.generators_t['p_set']
```

[ ]:
```
# simple optimization without considering outages due to contingency
network.optimize.optimize_security_constrained(snapshots=snapshots,␣
↪branch_outages=[], solver_name="cbc")
```

```
/opt/miniconda3/envs/re-demo/lib/python3.11/site-packages/linopy/common.py:133:
UserWarning:

coords for dimension(s) ['snapshot'] is not aligned with the pandas object.
Previously, the indexes of the pandas were ignored and overwritten in these
cases. Now, the pandas object's coordinates are taken considered for alignment.

/opt/miniconda3/envs/re-demo/lib/python3.11/site-packages/linopy/common.py:133:
UserWarning:

coords for dimension(s) ['snapshot'] is not aligned with the pandas object.
Previously, the indexes of the pandas were ignored and overwritten in these
cases. Now, the pandas object's coordinates are taken considered for alignment.

INFO:linopy.model: Solve problem using Cbc solver
INFO:linopy.io: Writing time: 0.27s
INFO:linopy.solvers:Welcome to the CBC MILP Solver
Version: 2.10.10
Build Date: Aug  1 2023

command line - cbc -printingOptions all -import
/var/folders/4j/bqbqmtcd37q6sn93_hr93cwr0000gn/T/linopy-problem-n7_5_mvi.lp
-solve -solu /var/folders/4j/bqbqmtcd37q6sn93_hr93cwr0000gn/T/linopy-solve-
bduuejg7.sol (default strategy 1)
Option for printingOptions changed from normal to all
Continuous objective value is 2190.04 - 0.00 seconds
Cgl0004I processed model has 59 rows, 82 columns (0 integer (0 of which binary))
and 154 elements
Cbc3007W No integer variables - nothing to do
Cuts at root node changed objective from 2190.04 to -1.79769e+308
Probing was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were
active after adding rounds of cuts (0.000 seconds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after
```

```
adding rounds of cuts (0.000 seconds)
ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)

Result - Optimal solution found

Objective value:                2190.03906250
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.01
Time (Wallclock seconds):       0.02

Total time (CPU seconds):       0.02   (Wallclock seconds):       0.03


INFO:linopy.constants: Optimization successful:
Status: ok
Termination condition: optimal
Solution: 504 primals, 960 duals
Objective: 2.19e+03
Solver model: not available
Solver message: Optimal - objective value 2190.03906250


INFO:pypsa.optimization.optimize:The shadow-prices of the constraints Generator-
com-p-lower, Generator-com-p-upper, Generator-com-transition-start-up,
Generator-com-transition-shut-down, Generator-com-status-
min_up_time_must_stay_up, Line-fix-s-lower, Line-fix-s-upper, Transformer-fix-s-
lower, Transformer-fix-s-upper, Kirchhoff-Voltage-Law were not assigned to the
network.
```

[ ]:
```python
# let's print all the solution

results_path = "opt_wo_security"
if not os.path.exists(results_path):
    os.makedirs(results_path)

# produce static data files
network.buses.to_csv(f"{results_path}/static_buses.csv")
network.lines.to_csv(f"{results_path}/static_lines.csv")
network.transformers.to_csv(f"{results_path}/static_transformers.csv")
network.generators.to_csv(f"{results_path}/static_generators.csv")
network.loads.to_csv(f"{results_path}/static_loads.csv")
# produce dynamic data files
network.buses_t['p'].to_csv(f"{results_path}/buses_injection_timeseries.csv")
network.loads_t['p'].to_csv(f"{results_path}/loads_timeseries.csv")
```

```
network.generators_t['p'].to_csv(f"{results_path}/
 ↪generation_production_timeseries.csv")
network.lines_t['p0'].to_csv(f"{results_path}/lines_from_timeseries.csv")
network.lines_t['p1'].to_csv(f"{results_path}/lines_to_timeseries.csv")
network.transformers_t['p0'].to_csv(f"{results_path}/
 ↪transformers_from_timeseries.csv")
network.transformers_t['p1'].to_csv(f"{results_path}/transformers_to_timeseries.
 ↪csv")



# produce post contingency flows for outage of line ln7-8
branch_outages = network.lines.index[4:5]
network.generators_t['p_set'] = network.generators_t['p']
dfm_post_contingency_flows = []
for snapshot in snapshots:
    dfm_post_contingency_flow = network.lpf_contingency(snapshot,␣
 ↪branch_outages=branch_outages)
    dfm_post_contingency_flow[['snapshot']] = snapshot
    dfm_post_contingency_flow = dfm_post_contingency_flow.reset_index().
 ↪rename(columns={"level_0": "branch", "level_1": "label"})
    dfm_post_contingency_flows.append(dfm_post_contingency_flow)


dfm_post_contingency_flows = pd.concat(dfm_post_contingency_flows)
dfm_post_contingency_flows.to_csv(f"{results_path}/post-ctg-flow.csv")
network.generators_t['p']
```

```
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 00:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 01:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 02:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 03:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 04:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
```

```
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 05:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 06:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 07:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 08:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 09:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 10:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 11:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 12:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 13:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 14:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 15:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 16:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
```

```
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 17:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 18:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 19:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 20:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 21:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 22:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 23:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
```

[ ]:

Generator	Gn1	Gn2	Gn3
snapshot			
2024-04-11 00:00:00+00:00	7.500000e+01	0.0	0.000000
2024-04-11 01:00:00+00:00	5.852734e+01	0.0	19.500000
2024-04-11 02:00:00+00:00	4.285938e+01	0.0	38.000000
2024-04-11 03:00:00+00:00	2.799609e+01	0.0	55.500000
2024-04-11 04:00:00+00:00	1.393750e+01	0.0	72.000000
2024-04-11 05:00:00+00:00	6.835938e-01	0.0	87.500000
2024-04-11 06:00:00+00:00	0.000000e+00	0.0	90.234375
2024-04-11 07:00:00+00:00	-2.500000e-07	0.0	92.089844
2024-04-11 08:00:00+00:00	0.000000e+00	0.0	93.750000
2024-04-11 09:00:00+00:00	-2.500000e-07	0.0	95.214844
2024-04-11 10:00:00+00:00	0.000000e+00	0.0	96.484375
2024-04-11 11:00:00+00:00	-2.500000e-07	0.0	97.558594
2024-04-11 12:00:00+00:00	0.000000e+00	0.0	98.437500
2024-04-11 13:00:00+00:00	-2.500000e-07	0.0	99.121094
2024-04-11 14:00:00+00:00	0.000000e+00	0.0	99.609375

```
2024-04-11 15:00:00+00:00 -2.500000e-07  0.0   99.902344
2024-04-11 16:00:00+00:00  0.000000e+00  0.0  100.000000
2024-04-11 17:00:00+00:00 -2.500000e-07  0.0   99.902344
2024-04-11 18:00:00+00:00  0.000000e+00  0.0   99.609375
2024-04-11 19:00:00+00:00 -2.500000e-07  0.0   99.121094
2024-04-11 20:00:00+00:00  0.000000e+00  0.0   98.437500
2024-04-11 21:00:00+00:00 -2.500000e-07  0.0   97.558594
2024-04-11 22:00:00+00:00  0.000000e+00  0.0   96.484375
2024-04-11 23:00:00+00:00 -2.500000e-07  0.0   95.214844
```

[ ]:
```
# simple optimization with considering outages due to contingency
branch_outages = network.lines.index[4:5]
network.generators_t['p_set'] = network_gen_p_set
network.optimize.optimize_security_constrained(snapshots=snapshots,␣
 ↪branch_outages=branch_outages, solver_name="cbc")
```

```
/opt/miniconda3/envs/re-demo/lib/python3.11/site-packages/linopy/common.py:133:
UserWarning:

coords for dimension(s) ['snapshot'] is not aligned with the pandas object.
Previously, the indexes of the pandas were ignored and overwritten in these
cases. Now, the pandas object's coordinates are taken considered for alignment.

/opt/miniconda3/envs/re-demo/lib/python3.11/site-packages/linopy/common.py:133:
UserWarning:

coords for dimension(s) ['snapshot'] is not aligned with the pandas object.
Previously, the indexes of the pandas were ignored and overwritten in these
cases. Now, the pandas object's coordinates are taken considered for alignment.

INFO:linopy.model: Solve problem using Cbc solver
INFO:linopy.io: Writing time: 0.09s
INFO:linopy.solvers:Welcome to the CBC MILP Solver
Version: 2.10.10
Build Date: Aug  1 2023

command line - cbc -printingOptions all -import
/var/folders/4j/bqbqmtcd37q6sn93_hr93cwr0000gn/T/linopy-problem-y89n8vfu.lp
-solve -solu /var/folders/4j/bqbqmtcd37q6sn93_hr93cwr0000gn/T/linopy-solve-
ctamws92.sol (default strategy 1)
Option for printingOptions changed from normal to all
Continuous objective value is 5352.34 - 0.00 seconds
Cgl0004I processed model has 138 rows, 115 columns (0 integer (0 of which
binary)) and 345 elements
Cbc3007W No integer variables - nothing to do
Cuts at root node changed objective from 5352.34 to -1.79769e+308
Probing was tried 0 times and created 0 cuts of which 0 were active after adding
```

```
rounds of cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were
active after adding rounds of cuts (0.000 seconds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)

Result - Optimal solution found

Objective value:                5352.34375000
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.03
Time (Wallclock seconds):       0.02

Total time (CPU seconds):       0.04   (Wallclock seconds):       0.02


INFO:linopy.constants: Optimization successful:
Status: ok
Termination condition: optimal
Solution: 504 primals, 1392 duals
Objective: 5.35e+03
Solver model: not available
Solver message: Optimal - objective value 5352.34375000


INFO:pypsa.optimization.optimize:The shadow-prices of the constraints Generator-
com-p-lower, Generator-com-p-upper, Generator-com-transition-start-up,
Generator-com-transition-shut-down, Generator-com-status-
min_up_time_must_stay_up, Line-fix-s-lower, Line-fix-s-upper, Transformer-fix-s-
lower, Transformer-fix-s-upper, Kirchhoff-Voltage-Law, Transformer-fix-s-lower-
security, Transformer-fix-s-upper-security, Line-fix-s-lower-security, Line-fix-
s-upper-security were not assigned to the network.
```

[ ]:
```python
# let's print all the solution

results_path = "opt_w_security"
```

```
if not os.path.exists(results_path):
    os.makedirs(results_path)

# produce static data files
network.buses.to_csv(f"{results_path}/static_buses.csv")
network.lines.to_csv(f"{results_path}/static_lines.csv")
network.transformers.to_csv(f"{results_path}/static_transformers.csv")
network.generators.to_csv(f"{results_path}/static_generators.csv")
network.loads.to_csv(f"{results_path}/static_loads.csv")
# produce dynamic data files
network.buses_t['p'].to_csv(f"{results_path}/buses_injection_timeseries.csv")
network.loads_t['p'].to_csv(f"{results_path}/loads_timeseries.csv")
network.generators_t['p'].to_csv(f"{results_path}/
 ↪generation_production_timeseries.csv")
network.lines_t['p0'].to_csv(f"{results_path}/lines_from_timeseries.csv")
network.lines_t['p1'].to_csv(f"{results_path}/lines_to_timeseries.csv")
network.transformers_t['p0'].to_csv(f"{results_path}/
 ↪transformers_from_timeseries.csv")
network.transformers_t['p1'].to_csv(f"{results_path}/transformers_to_timeseries.
 ↪csv")



# produce post contingency flows for outage of line ln7-8
branch_outages = network.lines.index[4:5]
network.generators_t['p_set'] = network.generators_t['p']
dfm_post_contingency_flows = []
for snapshot in snapshots:
    dfm_post_contingency_flow = network.lpf_contingency(snapshot,␣
 ↪branch_outages=branch_outages)
    dfm_post_contingency_flow[['snapshot']] = snapshot
    dfm_post_contingency_flow = dfm_post_contingency_flow.reset_index().
 ↪rename(columns={"level_0": "branch", "level_1": "label"})
    dfm_post_contingency_flows.append(dfm_post_contingency_flow)


dfm_post_contingency_flows = pd.concat(dfm_post_contingency_flows)
dfm_post_contingency_flows.to_csv(f"{results_path}/post-ctg-flow.csv")
```

```
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 00:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 01:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
```

```
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 02:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 03:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 04:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 05:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 06:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 07:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 08:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 09:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 10:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 11:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 12:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 13:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
```

```
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 14:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 15:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 16:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 17:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 18:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 19:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 20:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 21:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 22:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
INFO:pypsa.pf:Performing linear load-flow on AC sub-network SubNetwork 0 for
snapshot(s) DatetimeIndex(['2024-04-11 23:00:00+00:00'], dtype='datetime64[ns,
UTC]', name='snapshot', freq=None)
WARNING:pypsa.contingency:No type given for ln7-8, assuming it is a line
```