Carlos Colchero Leal

Transmission Network Optimization with PyPSA

Supervising Professor: Doctor Oliver Probst

Transmission Network Optimization with PyPSA

Power Flow in an AC Network

The models generated by PyPSA must satisfy the fundamental physical laws that govern the Power Flow in AC Networks. The basic node in every transmission network is the bus, which connects to other buses through links, lines and transformers. The Power Flow in a complicated AC Network is usually represented with a Non-Linear System of Equations in terms of the **voltages and admittances between the buses in a network.**

The information regarding the physical quantities in all the buses in a network can be stored inside vectors and matrices, synthetizing the power flow analysis to a matrix equation (Iowa State University, 2011).

$$S_n = V_n \cdot I_n^* = \mathrm{V}_n \sum_m Y_{nm}^* V_m^*$$

The total apparent power 'S' is the sum of both the **active** and the **reactive** power components. The labels 'n' and 'm' are linked to a bus within the network, so that 'V' and 'I' are N-dimensional complex vectors (or phasors) containing the voltage and current information, respectively. The N*M matrix 'Y' is known as the admittance matrix; the admittance between two buses is a complex number that measures how effortlessly will the current flow along a line (Zangwill, 2012). There are three different types of buses:

Slack Buses $n = 0$	**PQ Bus**	**PV Bus**
• The slack bus provides a reference point for the phase angles. • The Apparent Power is Unknown	• Total Apparent Power 'S' is known. • Both the Voltage magnitudes and angles are unknown.	• The Active Power 'P' is known • The voltage magnitudes are known • The Reactive Power and Voltage angles are unknown

Given that PV and PQ are the sets of buses, power flow is described by the non-linear system:

$$Re\left[V_n \sum_m Y_{nm}^* V_m^*\right] - P_n = 0 \quad \forall n \in PV \cup PQ$$

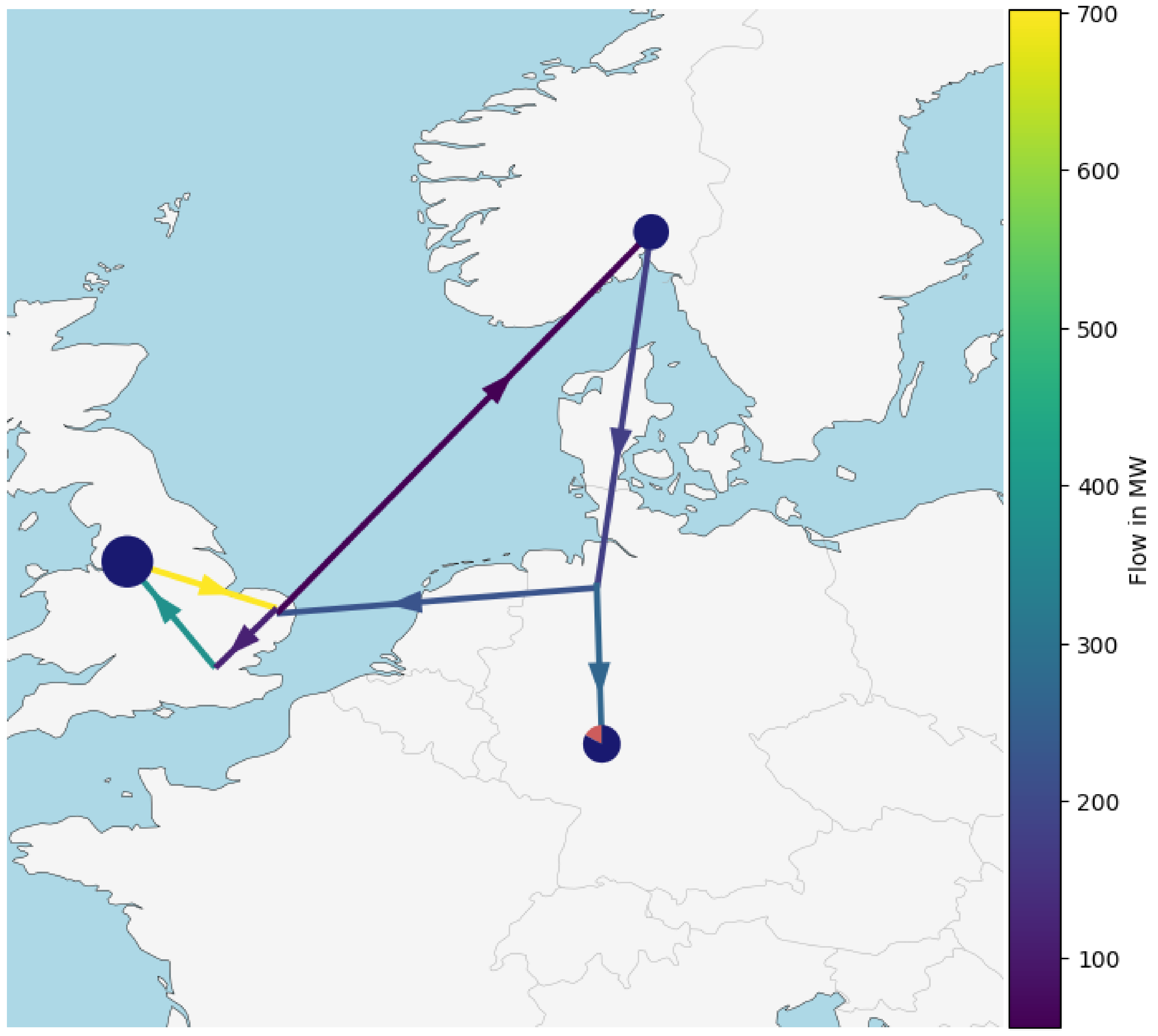
$$Im\left[V_n \sum_m Y_{nm}^* V_m^*\right] - Q_n = 0 \quad \forall n \in PQ$$

Linearized Power Flow Equations in an AC Network

It is possible to perform a linearization of the previously mentioned equations, providing a more efficient method to find approximate solutions to the power flow equations. The main assumptions of the linearization are that there are **no voltage magnitude variations, that the voltage angle differences are small** (small enough to apply the small angle approximation) **and that the branch resistances are negligible** so that the impedance of the branch is fully given by the reactance (Sereeter, 2021). Then, the active power is given by (Iowa State University, 2011). (Grainger, 1994):

$$P_n = \sum_m (KBK^T)_{nm}\theta_m - \sum_l K_{nl} b_l \theta_l^{shift}$$

Matrix 'B' is known as the susceptance matrix, the imaginary component of the admittance matrix: $Y_{nm} = G_{nm} + iB_{nm}$ which measures how easy it is for AC current to pass through a capacitance or inductance. 'K' is known as the incidence matrix which numerically describes the connections between nodes and edges in the network. The remaining term simply accounts for the possible phase shifts generated by the presence of transformers. **Given the active power P_n (except for the slack bus), PyPSA will find the voltage angles at all busses (except $\theta_0 = 0$).**



The branch active power rating, which is the maximum amount of active power that a branch can carry without exceeding thermal limits, can be found using:

$$F_l = \sum_i (BK^\tau)_{li}\,\theta_i - b_l \theta_l^{shift}$$

The network can then be plotted using PyPSA to visualize the flow within the buses at the found voltage angles.

The Objective Function

Every optimization procedure aims at minimizing or maximizing an objective function by adjusting the value of the decision variables, restricted by the given constraints. PyPSA can solve a variety of issues through the accurate selection and omission of constraints (Brown, et. al., 2024):

Economic Dispatch

- Finding the "best cost-effective point of operation for all controllable devices connected to the power system according to their economic efficiency in real time." (Sioshansi, 2022)

Linear Optimal Power Flow

- Using the linearized AC power flow equations, generator dispatch is optimized within a network, subject to the loading constraints of the network branches. (Brown, et. al., 2024), (Hörsch, 2018).

Security-Constrained Linear Optimal Power Flow

- The minimization function subject to contingencies and enforcing base case operating.

Capacity Expansion Planning

- Useful for long term, large investment projects, where an extension of the original capacity of the network takes place.

Modelling to Generate Alternatives

- In MGA, near cost minimum solution space is explored to study the near optimal solutions rather than directly discard them (Price, 2017).

The objective function introduced in PyPSA's documentation has several decision variables, applicable to any of the five previously described problems. Nevertheless, not all the listed variables will be relevant in the optimization problem since PyPSA allows for the omission of optional decision variables (such as storage, weather conditions…):

$$\min_{\substack{F_l, G_{n,r}, H_{n,s}, E_{n,s} \\ f_{l,t}, g_{n,r,t}, h_{n,s,t}, suc_{n,r,t}, sdc_{n,r,t}}} \left[\sum_l c_l \cdot F_l + \sum_{n,r} c_{n,r} \cdot G_{n,r} + \sum_{n,r,t} \left(\omega_t \cdot o_{n,r} \cdot g_{n,r,t} + suc_{n,r,t} + sdc_{n,r,t} \right) + \sum_{n,s} c_{n,s} \cdot H_{n,s} + \sum_{n,s} \hat{c}_{n,s} \cdot E_{n,s} + \sum_{n,r,t} \omega_t \cdot o_{n,s} \cdot \left[h_{n,s,t} \right]^+ \right]$$

(Brown, et. al., 2018):

Decision Variables

Each decision variable has an associated cost component, which assigns a weight to each variable according to its impact on the final cost. Finally, the linear constraints are applied to every decision variable, limiting the range of values each parameter can take during the optimization.

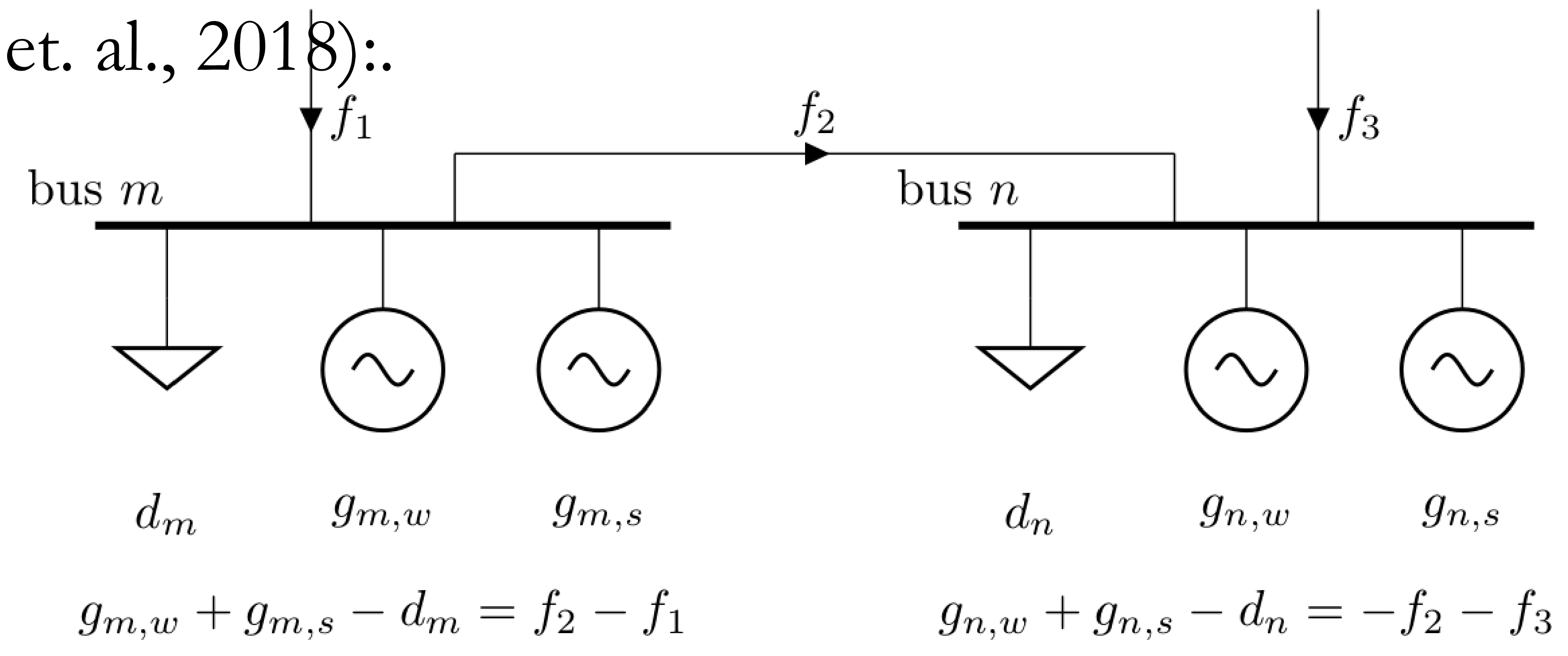
$$\min_{\substack{F_l, G_{n,r}, H_{n,s}, E_{n,s} \\ f_{l,t}, g_{n,r,t}, h_{n,s,t}, suc_{n,r,t}, sdc_{n,r,t}}} \left[\sum_l c_l \cdot F_l + \sum_{n,r} c_{n,r} \cdot G_{n,r} + \sum_{n,r,t} \left(\omega_t \cdot o_{n,r} \cdot g_{n,r,t} + suc_{n,r,t} + sdc_{n,r,t}\right) + \sum_{n,s} c_{n,s} \cdot H_{n,s} + \sum_{n,s} \hat{c}_{n,s} \cdot E_{n,s} + \sum_{n,r,t} \omega_t \cdot o_{n,s} \cdot \left[h_{n,s,t}\right]^+ \right]$$

(Brown, et. al., 2018):

Decision variable	Definition	Associated cost
F_l	The branch active power rating for each branch l.	c_l is the fixed cost per capacity.
$G_{n,r}$	The generator capacity at each bus 'n' for a type of technology 'r'	$c_{n,r}$ is the fixed cost per capacity
$g_{n,r,t}$	The electric dispatch of the generator at a time 't'	$o_{n,r}$ is the associated variable cost.
$suc_{n,r,t}, sdc_{n,r,t}$	The start-up and shut-down costs whenever unit commitment is activated.	NA
$H_{n,s}, E_{n,s}$	The storage unit power capacities and the store energy capacities respectively, each at bus 'n' for technology 's' .	There are fixed costs for the storage units $c_{n,s}$ as well as time-dependent costs for the stores $\hat{c}_{n,s}$.
$\left[h_{n,s,t}\right]^+$	The positive component of the storage dispatch	$o_{n,s}$is the associated variable cost
ω_t	Each period has a weighting ω_t which accounts for changes in the weather and demand conditions.	NA
$f_{l,t}$	The branch flow, which is an optimization variable that does not explicitly appear in the objective function.	NA

Universal Constraints

The listed decision variables are constrained based on their physical characteristics and capabilities. Depending on the selection of constraints, a different problem can be studied. The following diagram depicts the interconnection between two buses and their individual components (Brown, et. al., 2018):.



f_1
f_2
f_3
bus m
bus n
d_m
$g_{m,w}$
$g_{m,s}$
d_n
$g_{n,w}$
$g_{n,s}$
$g_{m,w} + g_{m,s} - d_m = f_2 - f_1$
$g_{n,w} + g_{n,s} - d_n = -f_2 - f_3$

Energy Flow Balance (Brown, et. al., 2018):

The system must satisfy the energy flow balances. The energy balance equations are the more important constraints, as they guarantee that the network follows the conservation of energy:

$$\sum_s g_{n,s,t} + \sum_s h_{n,s,t} - \sum_s f_{n,s,t} - \sum_l K_{nl} f_{l,t} = \sum_s d_{n,s,t} \quad for \;\; \omega_t \lambda_{n,t},$$

Kirchhoff's Current Law (Brown, et. al., 2018):

All the active power at a node should be injected by the neighboring nodes.

$$P_n = \sum_{l \in incident(n)} f_{l,t} \;\; such\ that\ |f_{l,t}| \leq F_l$$

Kirchhoff's Voltage Law (Brown, et. al., 2018):

KVL is an optional constraint that can be manually overridden for active branches. However, in the case of a passive branch, the KVL is given by the cycle matrix and the reactance along a cycle:

$$\sum_l C_{cl} \cdot x_l \cdot f_{l,t} = 0 \quad \forall c, t \; such\ that\ |f_{l,t}| \leq F_l$$

Constraints for Generators and Storage Units

The optimization problem can be extended through the introduction of generators, stores and storage units.

Generator Constraints (Brown, et. al., 2018):

The dispatch of generators $g_{n,r,t}$ is constrained by the generator capacity $G_{n,r}$ and the time-dependent availabilities (how much power can be dispatched at a given time). In the case of wind and solar energies, $\bar{g}_{n,r,t}$ **represents the weather-dependent power availability,** while $\tilde{g}_{n,r,t}$ represents the curtailment.

$$\tilde{g}_{n,r,t} \cdot G_{n,r} \leq \mathrm{g}_{\mathrm{n,r,t}} \leq \bar{g}_{n,r,t} \cdot G_{n,r} \quad \forall \mathrm{n, r, t}$$

The generator's power capacity itself is subject to constraint conditions given by the minimum $\tilde{G}_{n,r}$ and the maximum $\bar{G}_{n,r}$ installable potentials:

$$\tilde{G}_{n,r} \leq G_{n,r} \leq \bar{G}_{n,r} \quad \forall \mathrm{n, r}$$

Storage Unit Constraints (Brown, et. al., 2018):

The dispatch of storage units $h_{n,s,t}$ has a similar constraint to the dispatch of the generators. However, $\tilde{h}_{n,s,t}$ is now negative, since the dispatch of storage units can be positive when discharging and negative when absorbing power from the grid.

$$\tilde{h}_{n,s,t} \cdot H_{n,s} \leq h_{n,s,t} \leq \bar{h}_{n,s,t} \cdot H_{n,s} \quad \forall \mathrm{n, s, t}$$

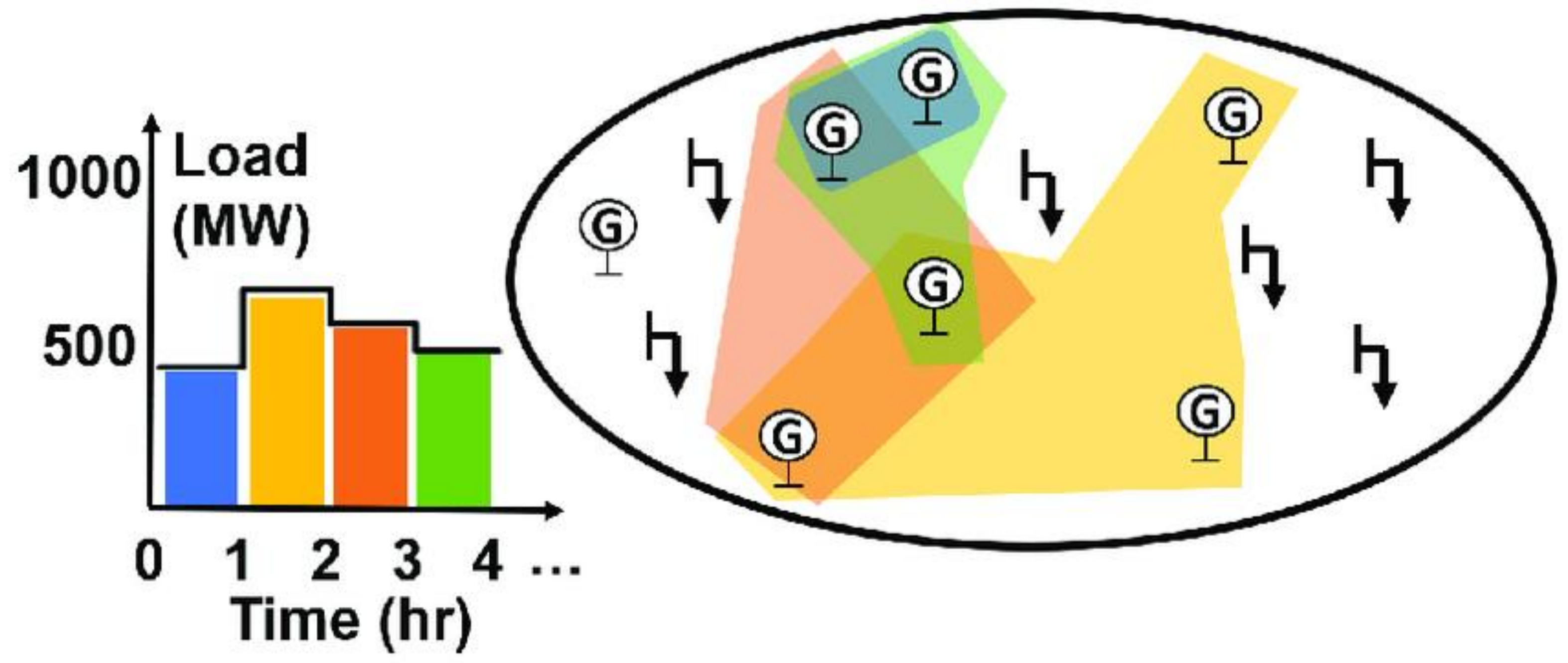
Due to the conservation of energy, the energy levels $e_{n,s,t}$ of all storage units must be consistent between all hours, only being limited by the storage energy capacity $E_{n,s}$.

$$e_{n,s,t} = \eta_{\mathrm{n,s,0}}^{\mathrm{w_t}} e_{n,s,t-1} + \eta_{n,s,+} \cdot \omega_t \left[h_{n,s,t}\right]^{+} - \eta_{n,s,-}^{-1} \cdot \omega_t \left[h_{n,s,t}\right]^{-} + \omega_t \cdot h_{n,s,t,inflow} - \omega_t \cdot h_{n,s,t,spillage}$$

$$\text{Such that } \tilde{e}_{n,s,t} \cdot E_{n,s} \leq e_{n,s,t} \leq \bar{e}_{n,s,t} \cdot E_{n,s} \quad \forall \mathrm{n, s, t}$$

Constraints for Unit Commitment

PyPSA allows for the implementation of Unit Commitment. NREL defines Unit Commitment as the scheduled operation of the grid participants to balance the supply and demand for electricity throughout a particular time period (Knueven, 2023). PyPSA aims at solving the complicated problem through the power flow linearization, as well as the introduction of time series. In this case each time step has a different "weight" in the simulation process, where each one of the weights is assigned to each time step using a time series. The constraints in this case are time-dependent, as well as the decision variables.



(Feliachi, 2022)

Binary Constraint

If unit commitment is activated, a variable that indicates if a generator is running at a specific time step is defined: $u_{n,r,t}$ can only take the values 0 or 1, indicating if generator is either not running or running in a specific period.

$$u_{n,r,t} \cdot \tilde{g}_{n,r,t} \cdot G_{n,r} \leq g_{n,r,t} \leq u_{n,r,t} \cdot \bar{g}_{n,r,t} \cdot G_{n,r} \quad \forall n,r,t$$

(Brown, et. al., 2018):

Up Time Constraint

"If the generator has just started up ($u_{n,r,t} - u_{n,r,t-1} = 1$) then it must run for at least $T_{n,r}^{min-up}$ periods"

$$\sum_{t\prime=t}^{t+T_{n,r}^{min-up}} u_{n,r,t\prime} \geq T_{n,r}^{min-up}(u_{n,r,t} - u_{n,r,t-1})$$

(Brown, et. al., 2018):

Down Time Constraint

The generator must cease operations for $T_{n,r}^{min-down}$ after having stopped running.

$$\sum_{t\prime=t}^{t+T_{n,r}^{min-up}} (1 - u_{n,r,t\prime}) \geq T_{n,r}^{min-up}(u_{n,r,t-1} - u_{n,r,t})$$

(Brown, et. al., 2018):

Associated startup cost

The cost associated with the start-up of the generators, is introduced into the objective function at each time step. The inequality is only non-zero if the generator has just started:

$$suc_{n,r,t} \geq suc_{n,r}(u_{n,r,t} - u_{n,r,t-1})$$

(Brown, et. al., 2018):

Associated shutdown cost

There is also an associated shut down cost, where the very act of shutting down a generator will lead to additional cost as the simulation advances:

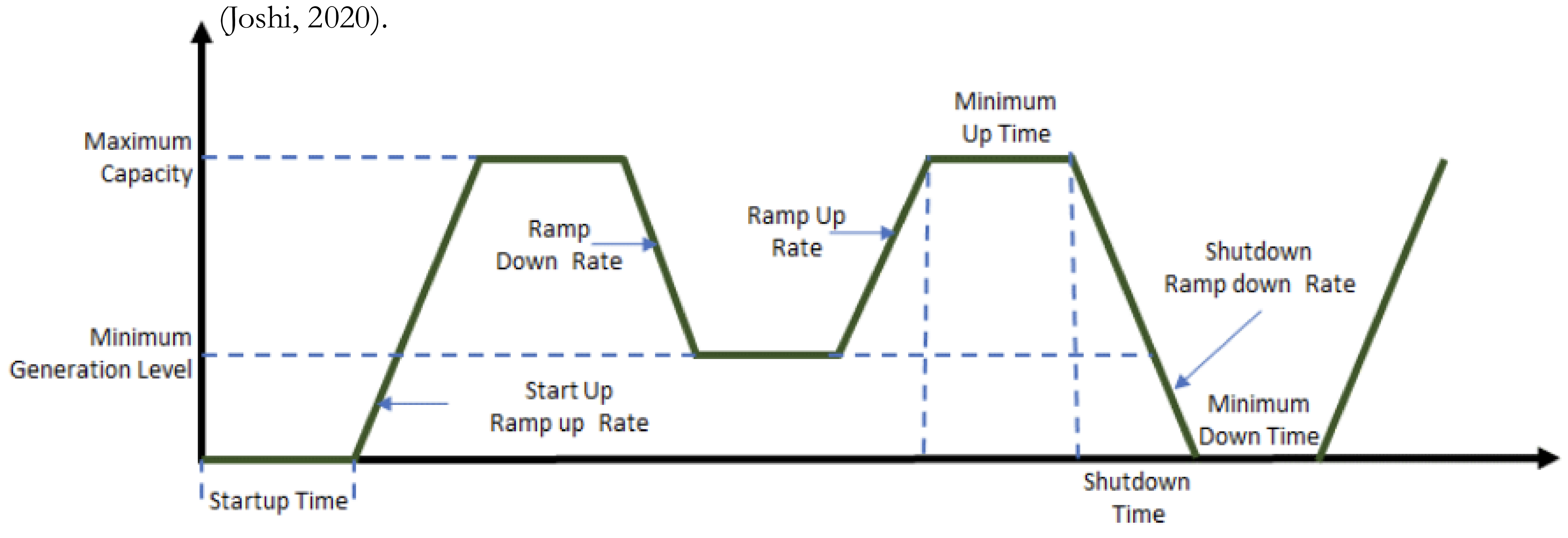
$$sdc_{n,r,t} \geq suc_{n,r}(u_{n,r,t} - u_{n,r,t-1})$$

(Brown, et. al., 2018):

Ramping Constraints

PyPSA allows for the implementation of ramp limits for generators and links. The introduction of ramping restricts the rate at which a generator or link can increase or decrease the power output at each time step. As explained by NREL, "power systems (…) are required to ensure a balance between load and generation to maintain the constant frequency of the grid, and, therefore, ensure reliable delivery of power" (Knueven, 2023).

(Joshi, 2020).



Ramping constraint

The ramp-up limit constricts the maximum rate at which power output can increase at each time step. In contrast, the ramp-down limit restricts the rate at which power output can decrease. The constraint is applied to the difference in dispatch between two consecutive time steps:

$$-\mathrm{rd}_{\mathrm{n,s}} * \overline{g_{n,s}} \leq (g_{n,s,t} - g_{n,s,t-1}) \leq \mathrm{ru}_{\mathrm{n,s}} * \overline{g_{n,s}}$$

(Brown, et. al., 2018):

Ramping constraint with Unit Commitment

For unit commitment, it is possible to also specify the ramp limits at start-up and shut-down: rdsd (shut-down) and rusu (start-up):

$$[-\mathrm{rd}_{\mathrm{n,s}} * \mathrm{u}_{\mathrm{n,s,t}} - \mathrm{rdsd}_{\mathrm{n,s}}(\mathrm{u}_{\mathrm{n,s,t-1}} - \mathrm{u}_{\mathrm{n,s,t}})]\overline{g_{n,s}} \leq (g_{n,s,t} - g_{n,s,t-1})$$
$$\leq [ru_{\mathrm{n,s}} * \mathrm{u}_{\mathrm{n,s,t-1}} + rusu_{\mathrm{n,s}}(\mathrm{u}_{\mathrm{n,s,t}} - \mathrm{u}_{\mathrm{n,s,t-1}})]\overline{g_{n,s}}$$

(Brown, et. al., 2018):

Initial Values for Example Implementation

With the objective of showcasing the application of the fundamental constraints, an example model was optimized. The following step-by-step guide displays the simplicity of PyPSA implementation. Some of PyPSA's dependencies are: **Numpy, Pandas, Matplotlib…**

The Network data is stored in .N file, which contains a collection of data frames for each of the elements inside the network (Brown, et. al., 2024). The first table contains the information of the buses (a combination of AC and DC buses with given nominal voltage. The second table introduces the generators connected to the network.

Bus Name	Nominal Voltage	Carrier	x	y
London	380	AC	-0.13	51.5
Norwich	380	AC	1.3	52.6
Norwich DC	200	DC	1.3	52.5
Manchester	380	AC	-2.2	53.47
Bremen	380	AC	8.8	53.08
Bremen DC	200	DC	8.8	52.98
Frankfurt	380	AC	8.7	50.12
Norway	380	AC	10.75	60
Norway DC	200	DC	10.75	60

Name	Bus	Capital Cost	Eff	Marginal cost	Nominal Power	Min Nominal Power	Carrier
Manchester Wind	Manchester	2793.6	1	0.11	80	80	wind
Manchester Gas	Manchester	196.6	0.35	4.532	50000	0	gas
Norway Wind	Norway	2184.3	1	0.09	100	100	wind
Norway Gas	Norway	158.3	0.356	5.892	20000	0	gas
Frankfurt Wind	Frankfurt	2129.5	1	0.1	110	100	wind
Frankfurt Gas	Frankfurt	102.7	0.355	4.086	80000	0	gas

In order to obtain a realistic simulation, the different time weightings for each of the carriers is introduced.

Snapshots	Manchester Wind	Frankfurt Wind	Norway Wind
01/01/15 0:00	0.93001988	0.5590784	0.9745832
01/01/15 1:00	0.48574758	0.75291037	0.48129038
01/01/15 2:00	0.23369174	0.12346509	0.4072258
01/01/15 3:00	0.25760422	0.96667665	0.59996496
01/01/15 4:00	0.62690557	0.8590078	0.52446822
01/01/15 5:00	0.60359841	0.52615379	0.00969271
01/01/15 6:00	0.67890755	0.07789301	0.22045336
01/01/15 7:00	0.36130261	0.05902347	0.8239185
01/01/15 8:00	0.62160405	0.2485545	0.55622973
01/01/15 9:00	0.52151837	0.10806017	0.43941604

Initial Values for Example Implementation

Furthermore, the table for the line information is introduced (Brown, et. al., 2024). The total apparent power and the reactance at each line must be given:

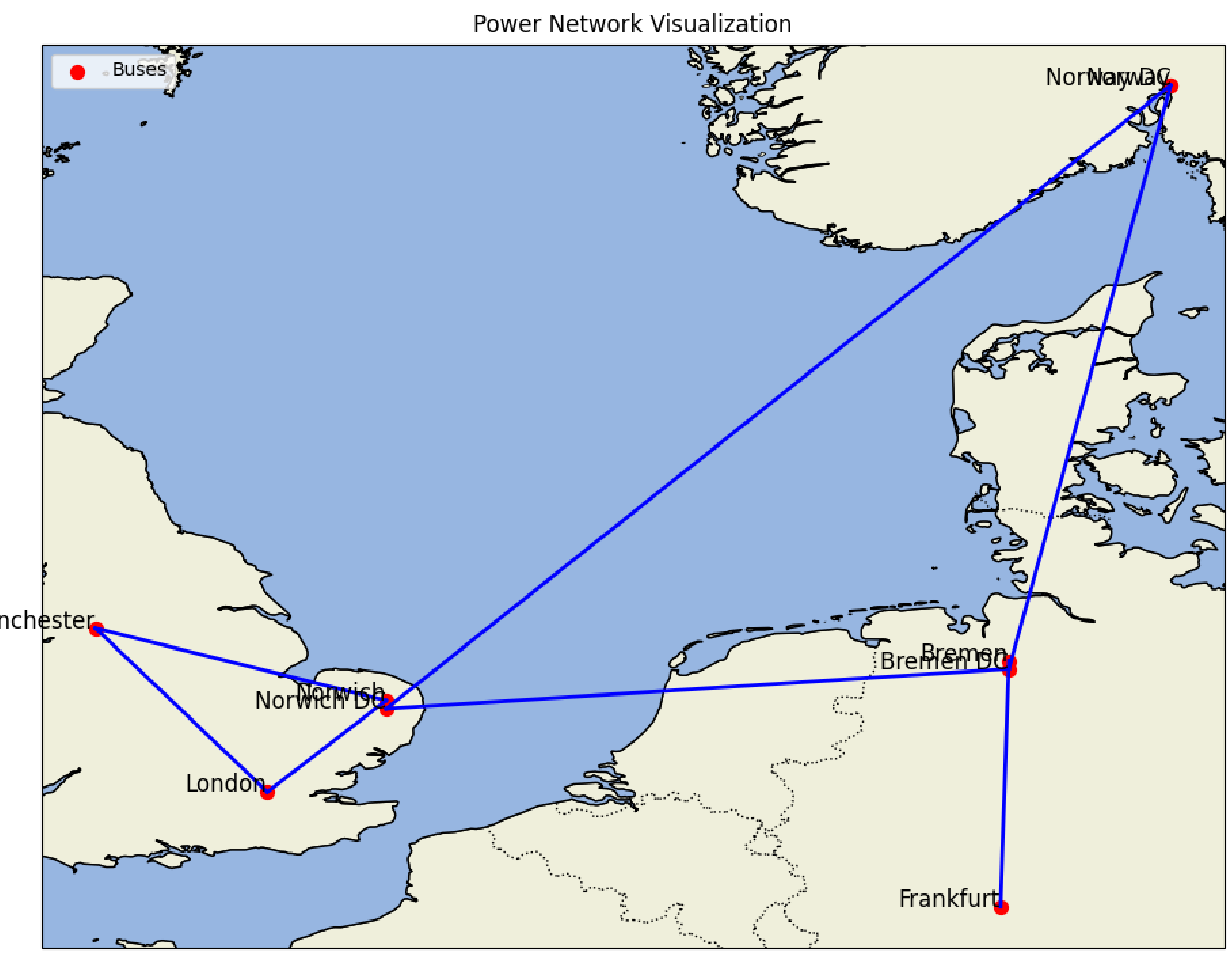
Line Name	Susceptance	Bus0	Bus1	Capital Cost	Apparent Power	Reactance	Resistance
0	0	London	Manchester	0.137	40000	0.79687828	0
1	0	Manchester	Norwich	0.133	40000	0.39155992	0
2	0	Bremen DC	Norwich DC	0.0087	40000	0	0.21260419
3	0	Norwich DC	Norway DC	0.129	40000	0	0.48616375
4	0	Norway DC	Bremen DC	0.062	40000	0	0.42872665
5	0	Norwich	London	0.0218	40000	0.23880035	0
6	0	Bremen	Frankfurt	0.2	40000	0.4	0

Additionally, the demand at each Snapshot for each location must be specified:

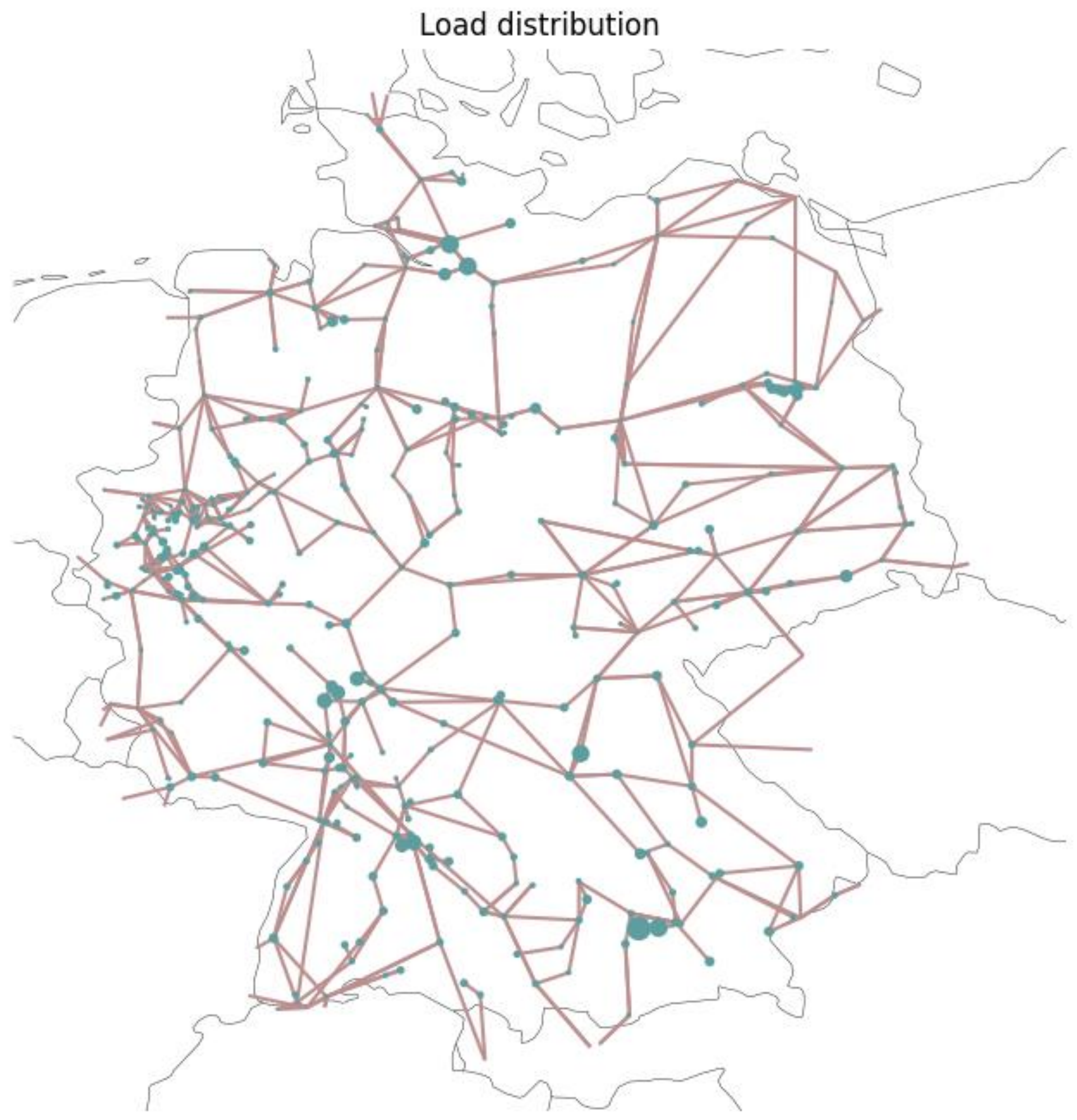
Snapshots	London	Norwich	Frankfurt	Bremen	Norway	Manchester
01/01/15 0:00	35.7962441	415.462564	398.047847	640.086378	820.035836	857.55144
01/01/15 1:00	976.824561	262.606146	432.436106	703.554334	854.834047	750.599624
01/01/15 2:00	250.587312	418.476353	379.803928	440.83613	42.5507444	156.564876
01/01/15 3:00	130.753145	552.959539	868.361764	612.576306	647.548233	527.870822
01/01/15 4:00	151.100169	218.159858	548.770755	803.436781	884.073873	83.897759
01/01/15 5:00	931.857052	791.976266	828.665243	605.400687	509.062449	676.623319
01/01/15 6:00	289.848287	531.870681	449.290752	641.09059	595.607965	731.1371
01/01/15 7:00	864.343322	23.5134667	699.163766	408.008541	291.64245	553.344889
01/01/15 8:00	689.577264	970.059068	915.86678	912.247776	2.15349255	298.338082
01/01/15 9:00	627.878986	0.92483369	414.887646	898.053092	760.740177	768.290586

Visualization of the Network

With the given information, it is possible to plot the Network using PyPSA's graphing tools:



```
## LOAD THE NETWORK ##
n = pypsa.Network("/Users/carloscolchero/Desktop/Probst/Proyecto 2. DLR/ac-dc-
data.nc")

## CREATE THE FIGURE ##
fig, ax = plt.subplots(figsize=(10, 8), subplot_kw={'projection': ccrs.PlateCarree()})

## ADD VISUAL ELEMENTS USING CARTOPY ##
ax.coastlines()
ax.add_feature(cfeature.BORDERS, linestyle=':')
ax.add_feature(cfeature.LAND, edgecolor='black')
ax.add_feature(cfeature.OCEAN)

## EXTRACT THE BUSES' LOCATIONS ##
bus_x = n.buses['x']
bus_y = n.buses['y']

## PLOT THE BUSES WITH THE GIVEN LOCATIONS ##
ax.scatter(bus_x, bus_y, color='red', s=50, label='Buses',
transform=ccrs.PlateCarree())

## PLOT THE LINES ##
for i, line in n.lines.iterrows():
bus0 = n.buses.loc[line['bus0']]
bus1 = n.buses.loc[line['bus1']]
ax.plot([bus0['x'], bus1['x']], [bus0['y'], bus1['y']], color='blue', linewidth=2,
transform=ccrs.PlateCarree())

## ADD THE NAME OF EACH BUS ##
for i, bus in n.buses.iterrows():
ax.text(bus['x'], bus['y'], bus.name, transform=ccrs.PlateCarree(), fontsize=12,
ha='right')

plt.title("Power Network Visualization")
plt.legend()
plt.show()
```



The information presented in previous slides is contained by the 'n' object. The canopy and matplotlib libraries are used to build a map representation of the Network. Other plots can contain the information on the reactance of lines or the voltage of buses.

Constraint Implementation

The first step to set up the optimization problem is to load the information contained in the network file:

```
n = pypsa.Network("/Users/carloscolchero/Desktop/Probst/Proyecto 2. DLR/ac-dc-data.nc")
```

In this example, the operational constraints such as the minimum nominal power, the reactance and resistance in the network, the efficiency of the generators… are already specified in the Network file. Additionally, ramping constraints are implemented to the generators as follows:

```
n.generators.loc[n.generators.carrier == "gas", "p_nom_extendable"] = False
n.generators.loc[n.generators.carrier == "gas", "ramp_limit_down"] = 0.2
n.generators.loc[n.generators.carrier == "gas", "ramp_limit_up"] = 0.2
```

Furthermore, it is possible to add additional elements to the network without modifying the source data frames. In this case, storage units are added to de model:

```
n.add("StorageUnit","su",bus="Manchester",marginal_cost=10,inflow=50,p_nom_extendable=True,capital_cost=100,p_nom=2000,effici
ency_dispatch=0.5,cyclic_state_of_charge=True,state_of_charge_initial=1000,)

n.add("StorageUnit", "su2", bus="Manchester", marginal_cost=10, p_nom_extendable=True, capital_cost=50, p_nom=2000,
efficiency_dispatch=0.5, carrier="gas", cyclic_state_of_charge=False, state_of_charge_initial=1000,)
```

PyPSA also allows for **State of Charge** implementation, so that the available storage capacity of a unit varies along snapshots.

```
n.storage_units_t.state_of_charge_set.loc[n.snapshots[7], "su"] = 100
```

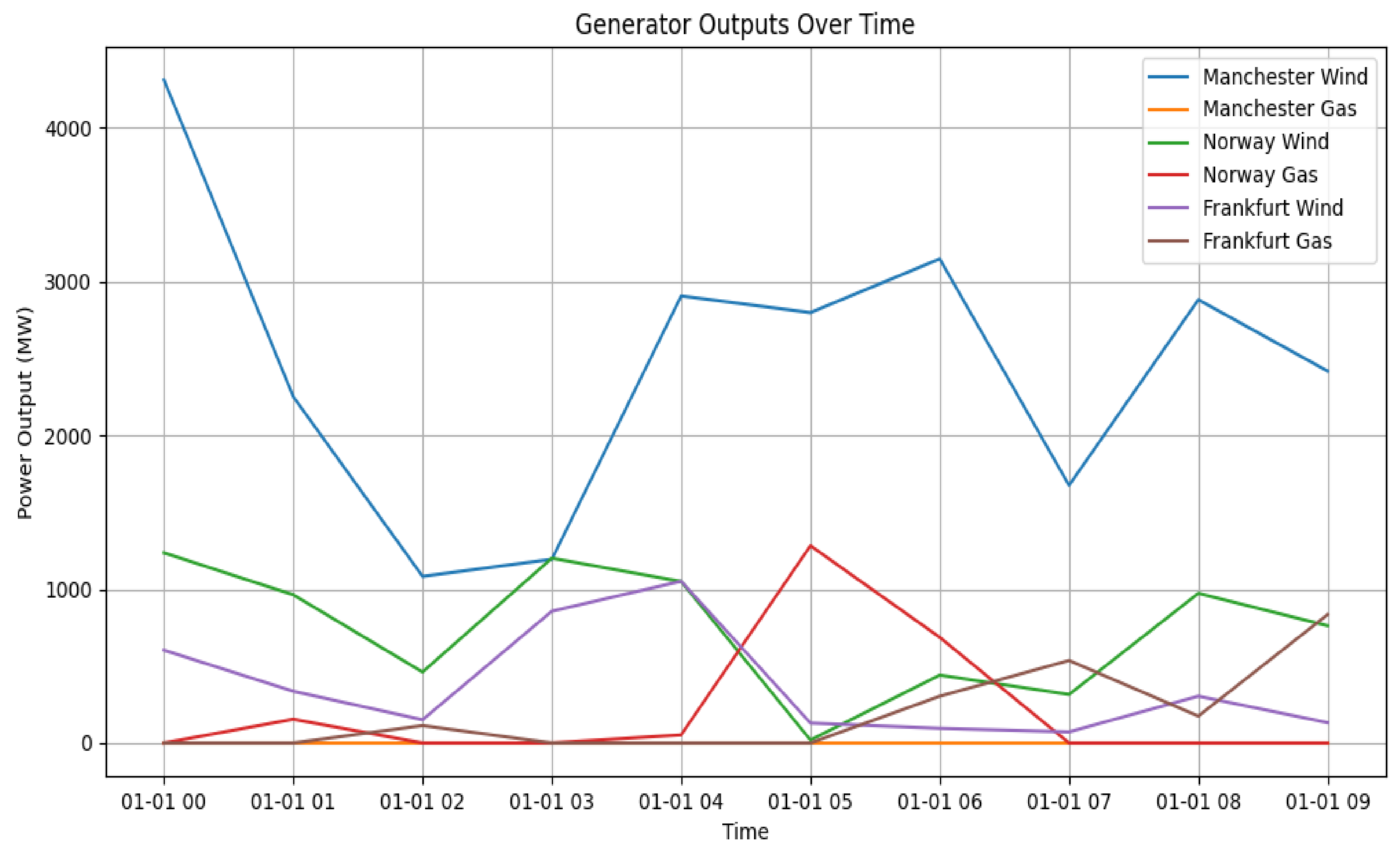
New buses and lines can be added and connected to the new component as follows:

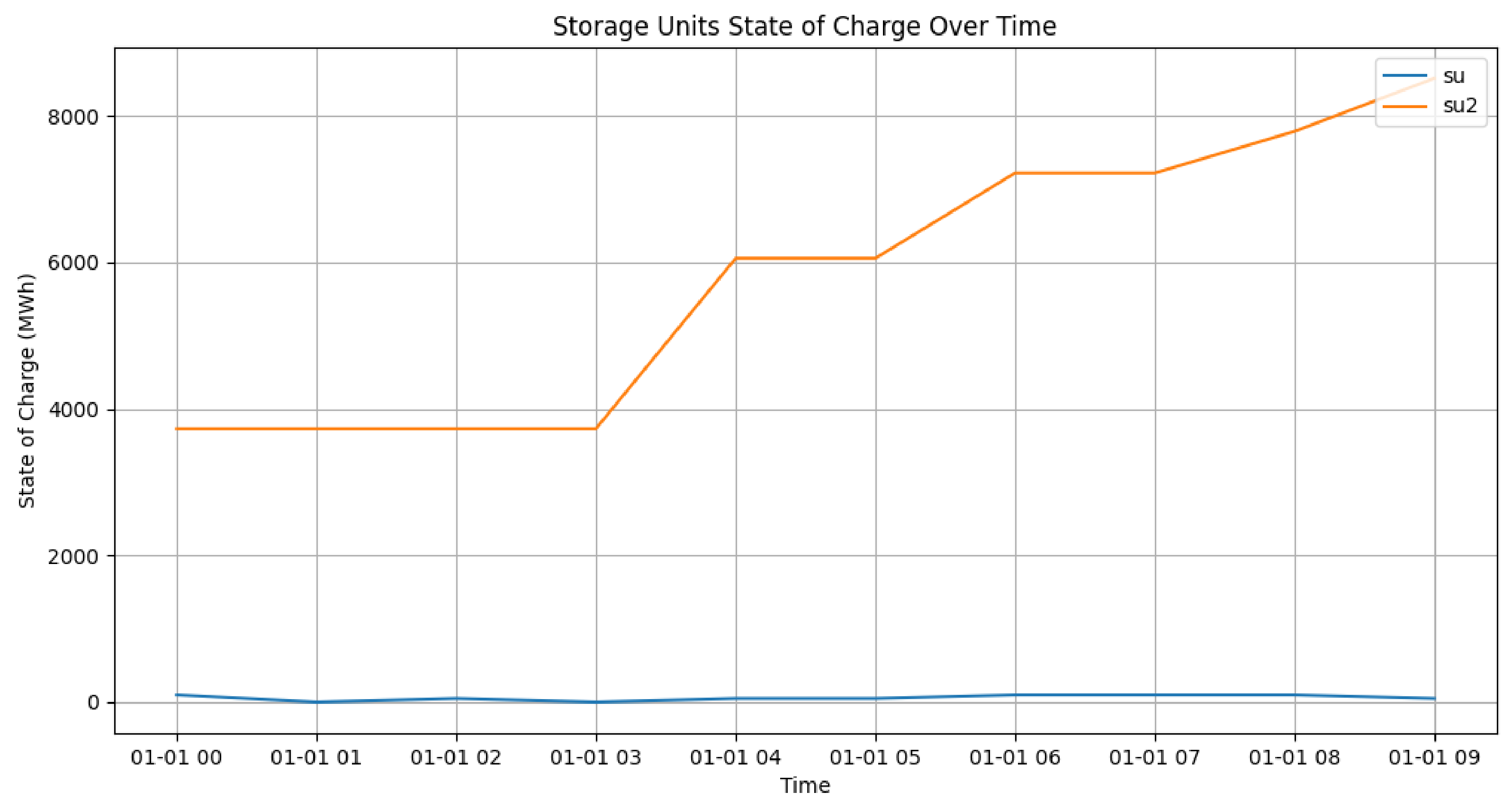
```
n.add("Bus", "storebus", carrier="hydro", x=-5, y=55)
n.madd("Link",["battery_power", "battery_discharge"],"",bus0=["Manchester", "storebus"],bus1=["storebus",
"Manchester"],p_nom=100,efficiency=0.9,p_nom_extendable=True,p_nom_max=1000,)

n.madd("Store",["store"],bus="storebus",e_nom=2000,e_nom_extendable=True,marginal_cost=10,capital_cost=10,e_nom_max=5000,e_initial=100,e_cyclic=True,)
```

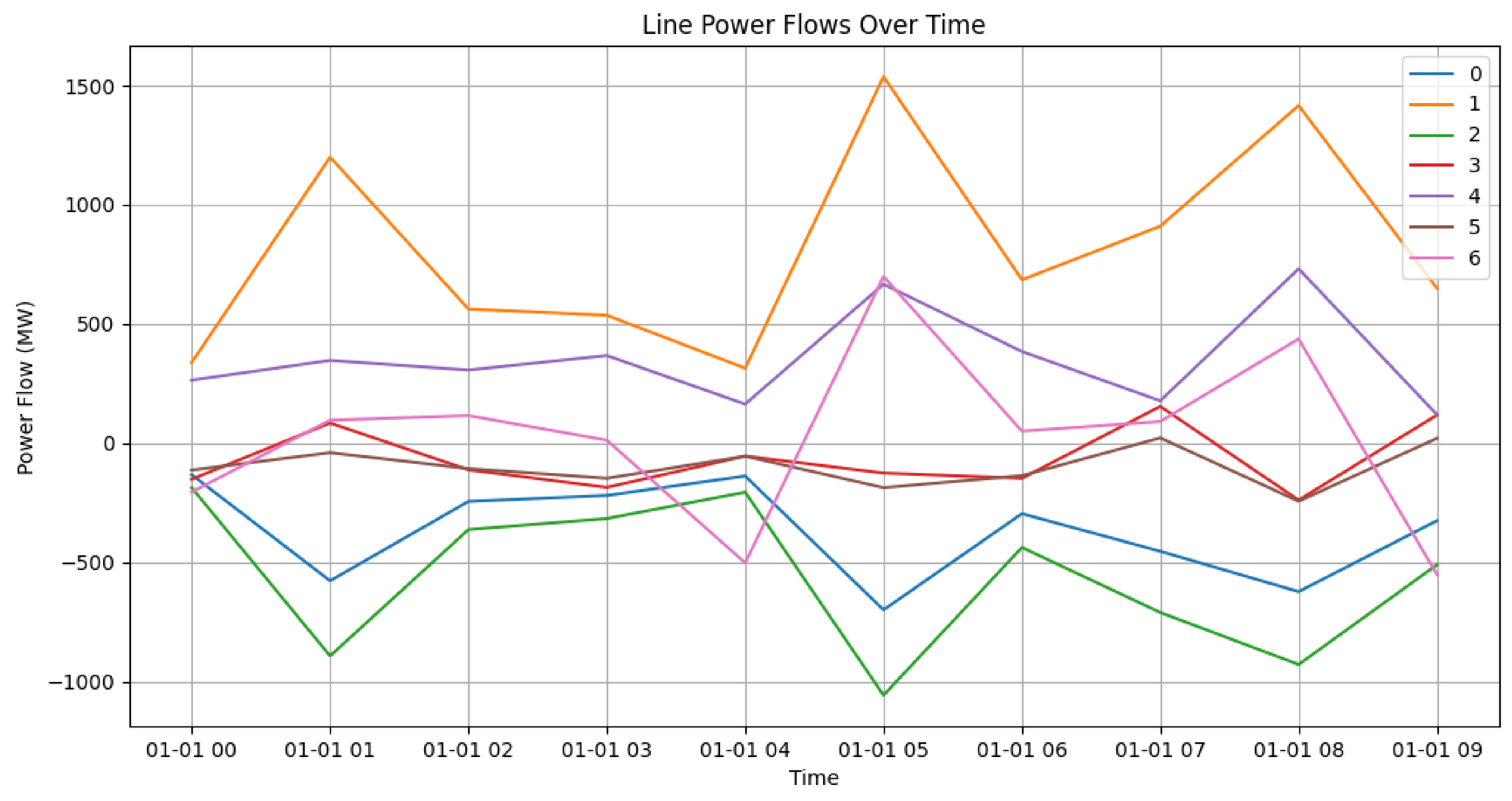
Network Analysis

The final step is to run the optimization command and analyze the given results. To analyze the results, libraries such as Pandas and Matplotlib can be imported. The generator output, state of charge of storage units and power flow in the lines will be studied:

```
generator_output = n.generators_t.p
storage_soc = n.storage_units_t.state_of_charge
line_flows = n.lines_t.p0
```







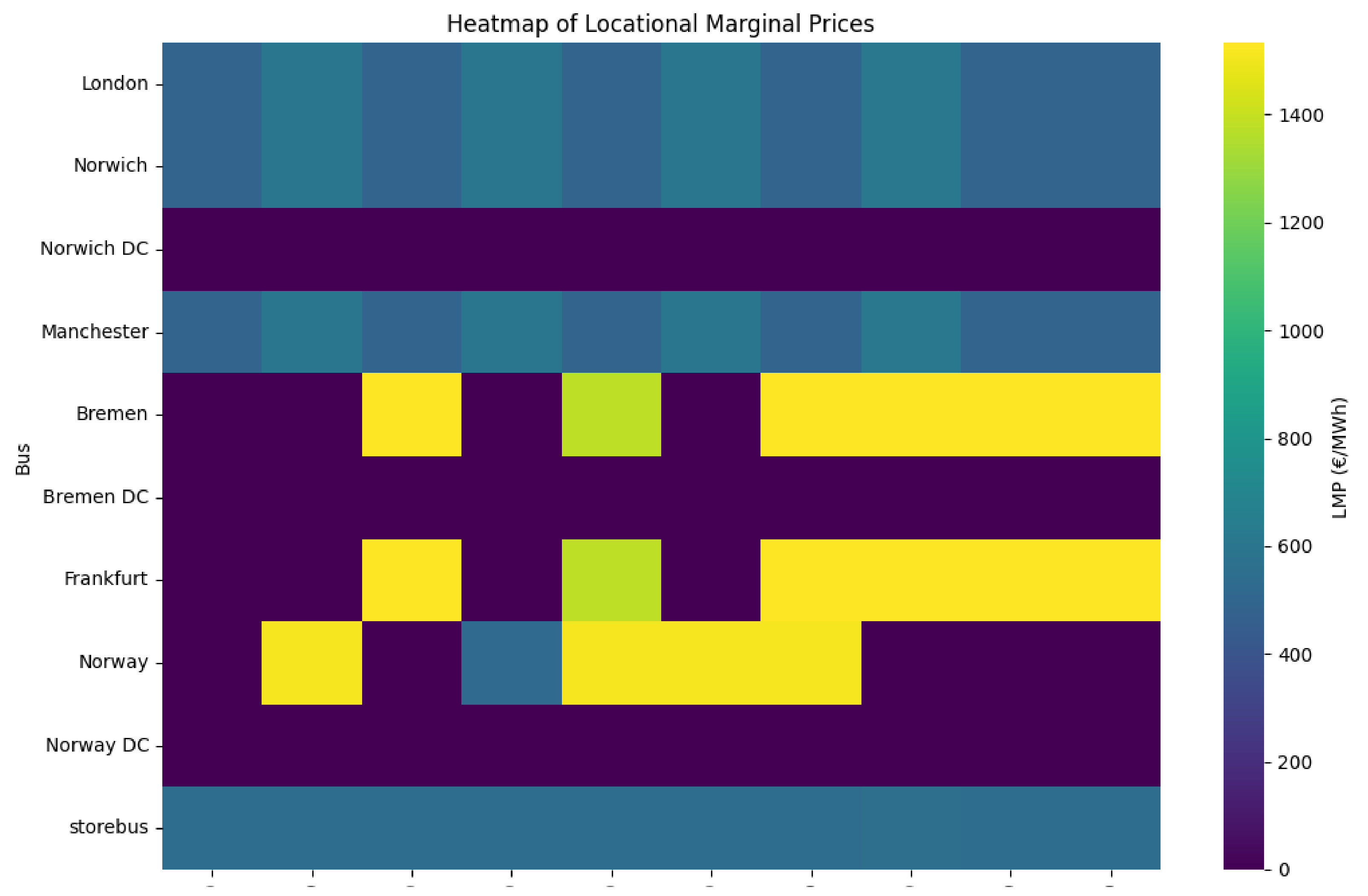
The final step is to run the optimization command and analyze the given results. To analyze the results, libraries such as Pandas and Matplotlib can be imported. The generator output, state of charge of storage units and power flow in the lines will be studied:

Finally, the minimization of the objective function will generate the **Total system cost: 19359124.844394356**

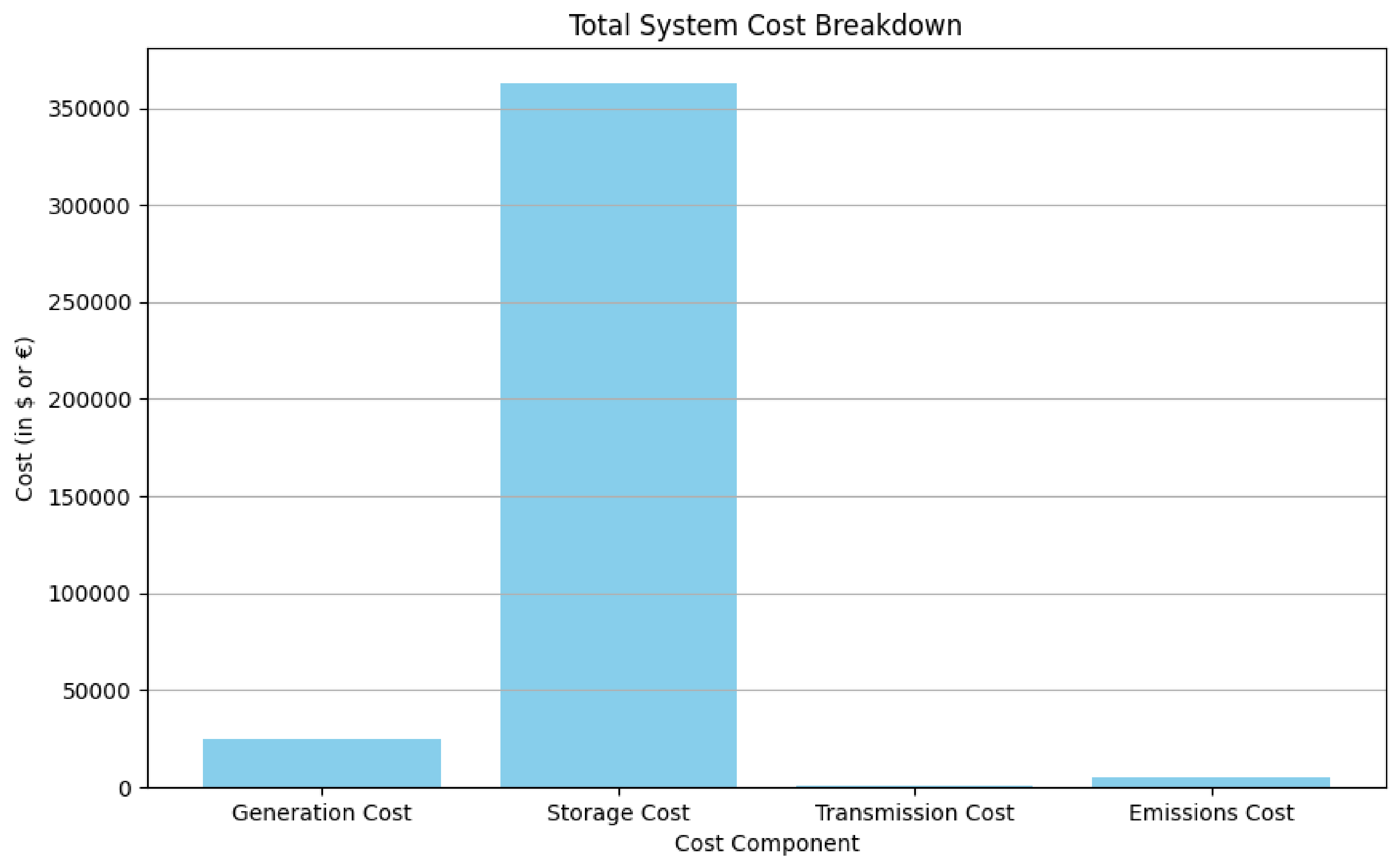
Cost Analysis

The final step is to run the optimization command and analyze the given results. To analyze the results, libraries such as Pandas and Matplotlib can be imported. The generator output, state of charge of storage units and power flow in the lines will be studied:

```
generator_output = n.generators_t.p
storage_soc = n.storage_units_t.state_of_charge
line_flows = n.lines_t.p0
```



Using the Seaborn library, it is possible to create a heat plot that shows the changes in the Marginal Price at each of the bus's locations as a function of time. This analysis is useful in determining the scope of the solution, and how it affects each of the involved nodes in the network.



Using the Matplotlib library and PyPSA's methods, it is possible to extract the information regarding **generation, storage, transmission and emissions cost.** The components of the solution to the optimization problem add up to the total cost showcased in the last slide.

Finally, it is possible to input information regarding investments, capacity expansions and custom cost components through the use of PyPSA's methods.

Bibliography

Brown, T., Hörsch, F., Hofmann, F., Neumann, F., Zeyen, L., Frysztacki, M., Glaum, P., Parzen, M. (2024). *Optimization with linopy*. PyPSA Documentation. https://pypsa.readthedocs.io/en/latest/examples/optimization-with-linopy.html

Brown, T., Hörsch, F., Hofmann, F., Neumann, F., Zeyen, L., Frysztacki, M., Glaum, P., Parzen, M. (n.d.). *AC-DC meshed examples* [GitHub repository]. GitHub. https://github.com/PyPSA/PyPSA/tree/master/examples/ac-dc-meshed

Brown, T., Hörsch, J., & Schlachtberger, D. (2018). *PyPSA: Python for power system analysis*. *Journal of Open Research Software*, *6*(1), 4. https://doi.org/10.5334/jors.188

Feliachi, A. (2022). A Multi-Layer Data-Driven Security Constrained Unit Commitment Approach with Feasibility Compliance. Energies. https://www.researchgate.net/figure/Unit-Commitment-Problem-Illustration-Colored-boxes-indicate-the-committed-generators-in_fig1_364591709

Grainger, J., Stevenson, W. (1994). *Basic Concepts*. Power System Analysis. McGraw-Hill.

Hörsch, H., Ronell, H., Witthaut, D., Brown, T. (2018). *Linear optimal power flow using cycle flows*. Electric Power Systems Research. https://doi.org/10.1016/j.epsr.2017.12.034

IAState. (2011). *Power flow equations* [PDF file]. Iowa State University. Retrieved from https://home.engineering.iastate.edu/~jdm/ee458_2011/PowerFlowEquations.pdf

Joshi, M., Palchak, D. (2020). *Ramping Up the Ramping Capability.* NREL. https://www.nrel.gov/docs/fy20osti/77639.pdf

Knueven, B. (2023). *Solving the Unit Commitment Problem*. NREL. https://www.nrel.gov/docs/fy23osti/85006.pdf

Price, J., Keppo, I. (2017). *Modelling to generate alternatives: A technique to explore uncertainty in energy-environment-economy models*. Applied Energy. https://www.sciencedirect.com/science/article/abs/pii/S0306261917302957

Sereeter, B., Markensteijn, A., Koote, M. (2021) *A novel linearized power flow approach for transmission and distribution networks*. Journal of Computational and Applied Mechanics. https://doi.org/10.1016/j.cam.2021.113572

Sioshansi, F. (2022). *Energy transformation and decentralization in future power systems*. Operation, Planning and Control Perspectives. https://doi.org/10.1016/B978-0-323-91698-1.00009-1

Zangwill, A. (2012). *Modern electrodynamics*. Cambridge University Press.