Working together for a green, competitive and inclusive Europe

This work was supported by a grant from the EEA Grants - Financial Mechanism 2014-2021, Projects of cooperation in university education" Education, Scholarships, Apprenticeship and Youth Entrepreneurship Program in Romania"

" Innovative Teaching methods for tomorrow's Renewable Energy Specialists"

19-COP-0025

The project is a cooperation between two partners: the Faculty of Physics at the University of Bucharest and the Iceland School of Energy at Reykjavik University

Static Power System Simulation using Python

Dr. Bogdan Dobrica

University of Bucharest, email: bdobrica@gmail.com

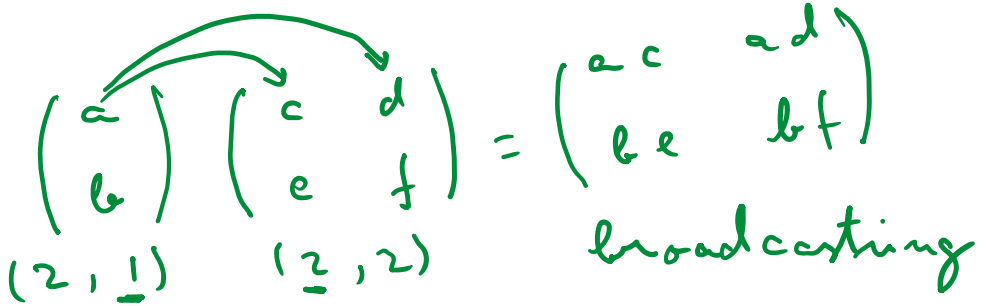
Why do we need Power Systems Simulations?

- More hybrid power systems are being connected to existing grids;
- Grids are legally required to be evaluated statically and dynamically to cover normal and abnormal operating conditions;
- To achieve maximum efficiency, each connected system needs to operate optimally.

Why use Python?

- The equations describing the grid behavior are non-linear and can be solved using numerical methods – hence, Python.

Python Tools



- NumPy – for working doing mathematical operations with matrices;
- Pandas – as we need also an Excel equivalent, right? DATAFRAME
- Jupyter Notebook – for ease of doing programming experiments;
- PyPSA – a readily available static power simulation system for Python;
  - https://pypsa.readthedocs.io/en/latest/ (documentation)
  - https://pypsa.org/ (official website)
- PyPSA-EUR – a collection of models covering European Network of Transmission System Operators.
  - https://pypsa-eur.readthedocs.io/en/latest/ (documentation)

Installing Required Software

- Download Python https://www.python.org/downloads/
  - Recommended version: 3.7
- Open a console and run the following commands:

```
$: python -m pip install –upgrade pip
$: python -m pip install numpy openpyxl pandas notebook pypsa
$: python -m jupyter notebook
```

Jupyter Notebook

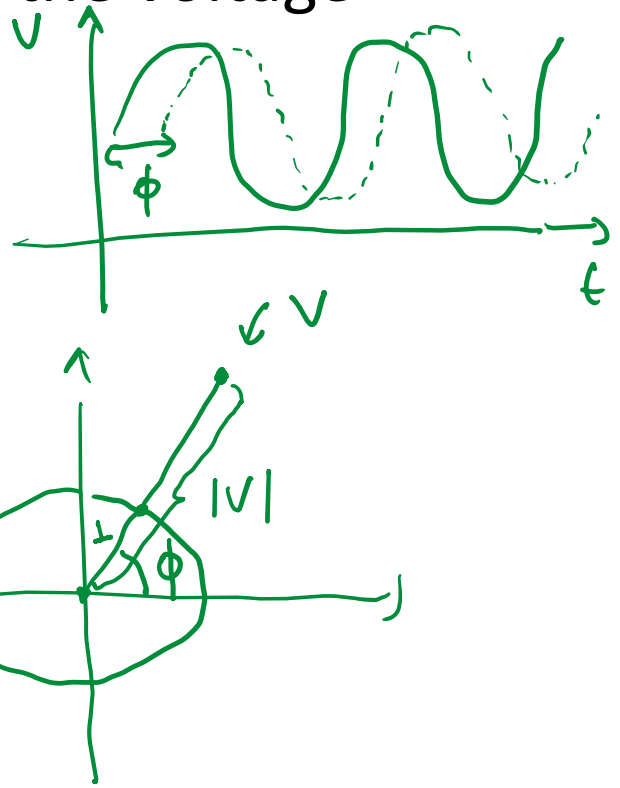
- A web-browser IDE for Python;
- Based on “running cells” – small pieces of code that share their environment:
  - For the current cell you’ll have access to all the objects defined in previously run cells.
- Useful shortcuts:
  - [a] creates a new cell before the current active cell; [b] creates one after
  - [m] on an active cell converts it to a “markup” cell; [y] to go back
  - [CTRL+ENTER] on an active cell runs the cell
  - [d][d] on an active cell, deletes the cell.

AC Power: A Few Considerations

- Considering a simple AC circuit, the general solution for the voltage will give:

$$V = |V|e^{j(\omega t + \phi)} = |V|e^{j\phi}e^{j\omega t} = \tilde{V}e^{jwt}$$

- Where:
  - V is the complex voltage solution to conservation of energy;
  - ω is the solution angular frequency;
  - ϕ is the solution phase, and;
  - t it the time.
- We call the term $\tilde{V}$ a phasor (**phas**e vect**or**)

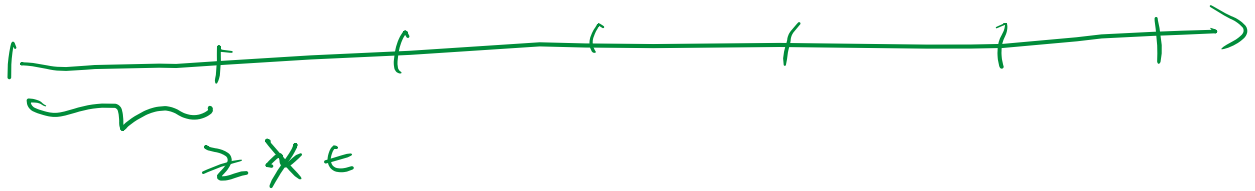


AC Power: Static Analysis

- In a normal power grid, the angular frequency is the same. The phasor becomes the only important factor.
- E.g. Ohm's Law:

$$\underline{V} = \underline{I}\underline{Z} \Leftrightarrow \tilde{V}e^{\omega t} = \tilde{I}\underline{Z}e^{\omega t} \Leftrightarrow \underline{\tilde{V}} = \underline{\tilde{I}}\underline{Z} \Leftrightarrow \underline{\tilde{V}}\underline{Y} = \underline{\tilde{I}}$$

- Where:
  - I is the current, and $\tilde{I}$ is the corresponding phasor;
  - V is the voltage, and $\tilde{V}$ is the corresponding phasor;
  - Z is the impedance / Y is the admittance seen by the generator.
- The equation is time-independent → steady state → static analysis.



AC Power: Phasor Notation

- Power in an AC circuit is given by:

$$\tilde{S} = P + jQ = \tilde{V}\tilde{I}^{\star}$$

- Where:
  - $\tilde{I}^{\star}$ is the complex conjugate of the current phasor;
  - $\tilde{S}$ is the complex power for which $|\tilde{S}|$ is the apparent power and for which
  - $P = Re(\tilde{S})$ is the real power;
  - $Q = Im(\tilde{S})$ is the reactive power.
- The power factor is $\cos(\theta) = \frac{P}{|\tilde{S}|}$

Power Grid

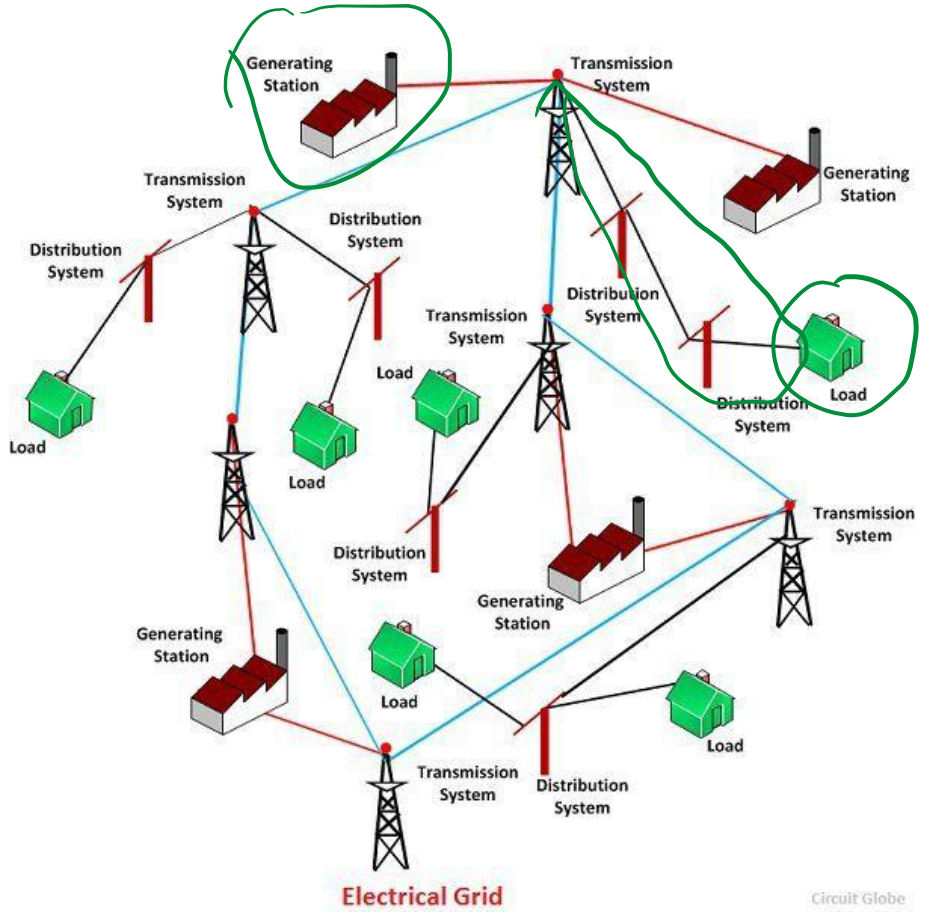
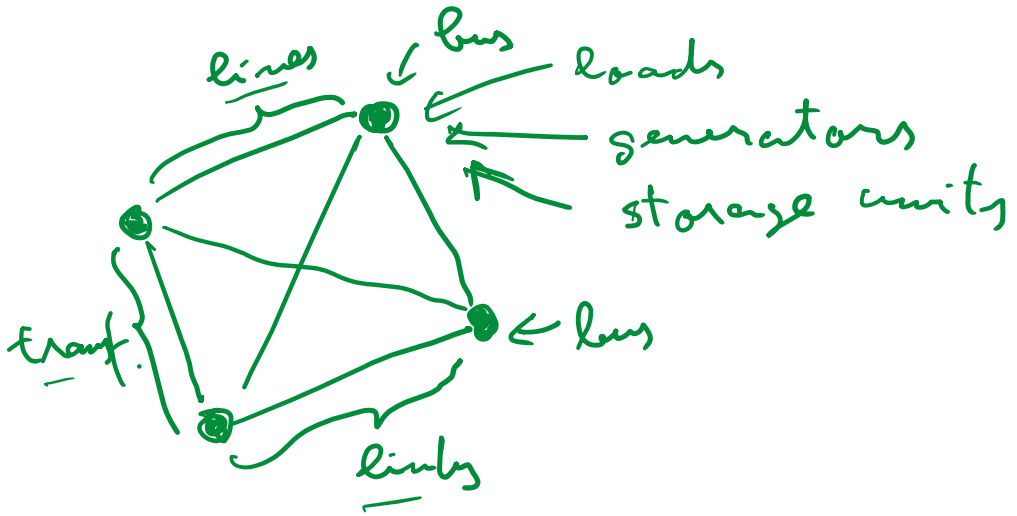


image source: circuitglobe.com

- A power grid is a network (i.e. graph)
- graph node = power grid bus
- edges = power grid lines / transformers



Power Grid in PyPSA

- The object is called pypsa.Network
- Parameters:

argument	type	unit	default	description
name	string	-	-	unique name
snapshots	list / pd.Index	-	[‘now’]	List of time “moments”
snapshot_weightings	pd.DataFrame	hours	1	How long is a time “moment”
now	any	-	‘now’	Current time “moment”
srid	integer	-	4326	Spatial Reference System Identifier

Power Grid in PyPSA

```
In [1]: import pypsa

        grid = pypsa.Network()

In [ ]:
```

```
In [1]: import pypsa

        grid = pypsa.Network(name = 'My Grid')

In [ ]:
```

AC Power Buses

- A bus is the node at which several components of the power system (e.g. generators, loads, transformers) are connected.
- A bus enforces energy and charge conservation laws.
- A bus has 4 parameters:
  - $|\tilde{V}|$ - the amplitude of the voltage;
  - ϕ – the phase angle of the voltage;
  - $P = Re(\tilde{S})$ – the real power;
  - $Q = Im(\tilde{S})$ – the reactive power.

AC Power Buses: Types

- Based on the availability of the parameters, there are 3 types of buses:

Bus Types	$\lvert\widetilde{V}\rvert$	θ	P	Q
Generator Bus (PV-bus)	known	unknown	known	unknown
Load Bus (PQ-bus)	unknown	unknown	known	known
Slack or Reference Bus	known	known	unknown	unknown

220 V ± 5%

Generator Bus

- Has at least one generator connected to it;
- The voltage amplitude is kept constant by controlling the reactive power.

Load Bus

- There are no generators connected to this bus;
- The load bus voltage can be permitted within a tolerable value (i.e. 5%);
- The phase angle is of little importance for the load.

Slack Bus

- Injects or absorbs active or reactive power from the system;
- Doesn’t have any loads connected to it;
- The phase angle is usually taken as 0;
- The slack bus is an artificial concept in load flow studies as the resistive losses of the system are not know accurately in advance;
- The slack bus is chosen arbitrarily from the busses that meet the conditions (no load, at least one generator);

Buses in PyPSA

- Use pypsa.Network.add(class_name='Bus') to add new buses.
- Parameters:

argument	type	unit	default	description
name	string	-	-	unique name
v_nom	float	kV	1	Nominal voltage
x	float	-	0	Position in the SRID system (longitude)
y	Float	-	0	Position in the SRID system (latitude)
carrier	string	-	AC	The carrier ('AC', 'DC', 'heat', 'gas' etc)
unit	string	-	None	Unit for the bus carrier (usually `MW`)
v_mag_pu_set	string / Series	/unit	1	Voltage magnitude set point.

Buses in PyPSA

In [1]:
```
import pypsa

grid = pypsa.Network(name = 'My Grid')
grid.add(
    class_name = 'Bus',
    name = 'Bus #1',
    v_nom = 0.22,
    x = 44.4268,
    y = 26.1025
)
```

In [2]:
```
grid.buses
```

Out[2]:

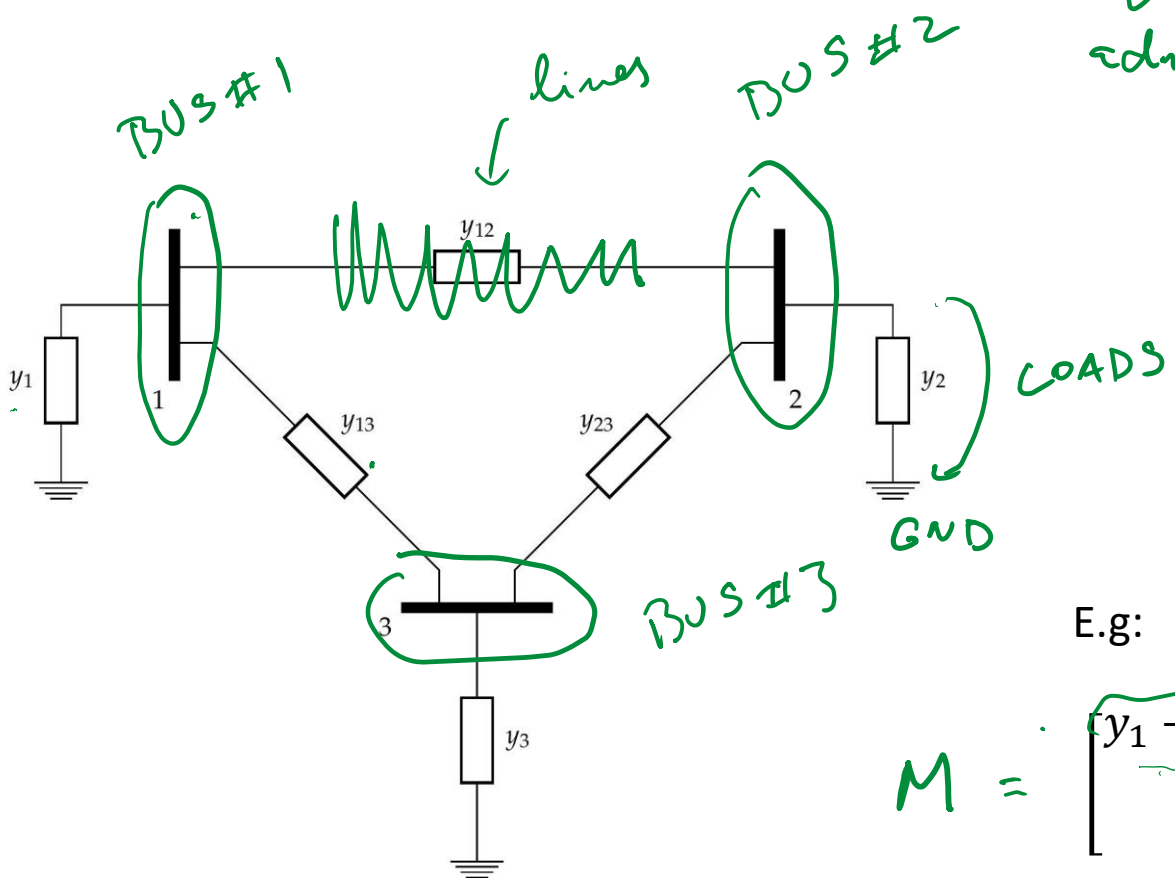
attribute	v_nom	type	x	y	carrier	unit	v_mag_pu_set	v_mag_pu_min	v_mag_pu_r
Bus #1	0.22		44.4268	26.1025	AC	None	1.0	0.0	

Bus Admittance Matrix

$Y = \frac{1}{Z}$ ← imp

↓ adm

$\tilde{I} = M \tilde{V}$



$$Y_{il} = \begin{cases} y_i + \sum_{\substack{k=1 \\ k \neq l}}^{N} y_{ik}, i = l \\ -y_{il}, i \neq l \end{cases}$$

$y_{il} = 0 \Leftrightarrow$ bus i is not connected to bus l.

E.g:

$$M = \begin{bmatrix} y_1 + y_{12} + y_{13} & -y_{12} & -y_{13} \\ -y_{12} & y_2 + y_{12} + y_{23} & -y_{23} \\ -y_{13} & -y_{23} & y_3 + y_{13} + y_{23} \end{bmatrix}$$

BUS #	1	2	3
1	-	0	-
2	0	-	-
3	-	-	-

Bus Equations

- In a system with N buses, for each bus $i \in \overline{1, N}$:

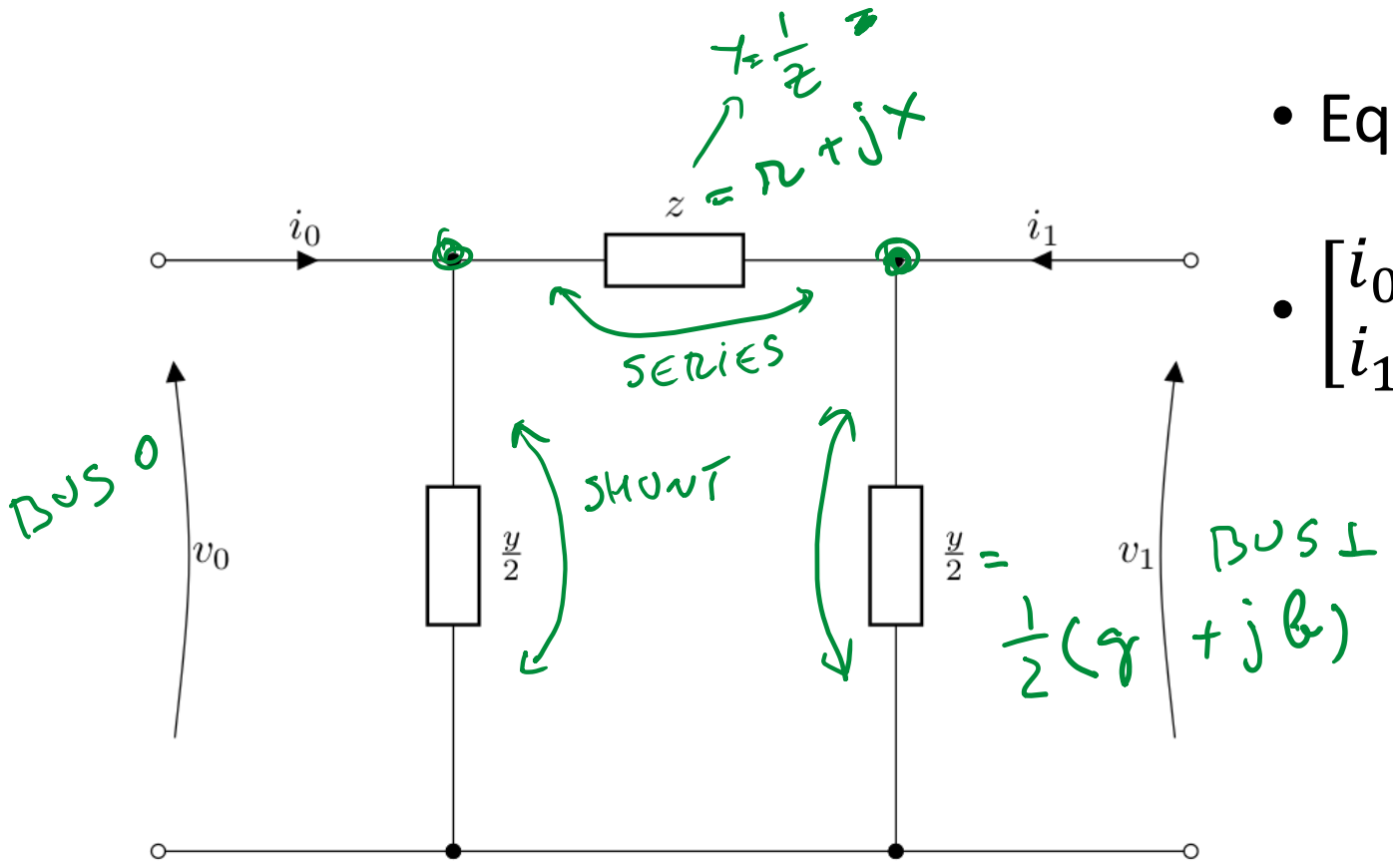
$$\begin{cases} P_i = \sum_{k=1}^{N} |\widetilde{V_i}||\widetilde{V_k}|(Re(Y_{ik})\cos(\phi_i - \theta_k) + Im(Y_{ik})\sin(\phi_i - \theta_k)) \\ Q_i = \sum_{k=1}^{N} |\widetilde{V_i}||\widetilde{V_k}|(Re(Y_{ik})\sin(\phi_i - \theta_k) - Im(Y_{ik})\cos(\phi_i - \theta_k)) \end{cases}$$

- Y is the bus admittance matrix, θ_i is the phase angle on bus i;
- P_i, Q_i are the active and reactive powers from bus i.

Lines

- Lines are transmission and distribution lines. In the network (graph) language they represent edges between the network's nodes.
- One line can connect two busses, both either AC or DC.
- Standard line properties can be found here:
  - https://pandapower.readthedocs.io/en/latest/std_types/basic.html

Line Model



- Equation:
- $$\begin{bmatrix} i_0 \\ i_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{z} + \frac{y}{2} & -\frac{1}{z} \\ -\frac{1}{z} & \frac{1}{z} + \frac{y}{2} \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}$$

Lines in PyPSA

- Use pypsa.Network.add(class_name='Line') to add new lines.
- Parameters:

argument	type	unit	default	Description
name	string	-	-	unique name
bus0	string	-	-	The name of the 1st connected bus
bus1	string	-	-	The name of the 2nd connected bus
type	string	-	-	e.g. '149-AL1/24-ST1A 10.0'
x	float	Ω	0	Series reactance: z = r + jx
r	float	Ω	0	Series resistance: z = r + jx
g	float	S	0	Shunt conductivity: y = g + jb
b	float	S	0	Shunt susceptance: y = g + jb

Lines in PyPSA

In [1]:
```
import pypsa

grid = pypsa.Network(name = 'My Grid')
grid.add(
    class_name = 'Bus',
    name = 'Bus #1',
    v_nom = 0.22,
    x = 44.4268,
    y = 26.1025
)
grid.add(
    class_name = 'Bus',
    name = 'Bus #2',
    v_nom = 0.22,
    x = 44.9367,
    y = 26.0129
)
```

In [2]:
```
grid.add(
    class_name = 'Line',
    name = 'Line #1',
    bus0 = 'Bus #1', bus1 = 'Bus #2',
    x = 0.1, r = 0.01
)
grid.lines
```

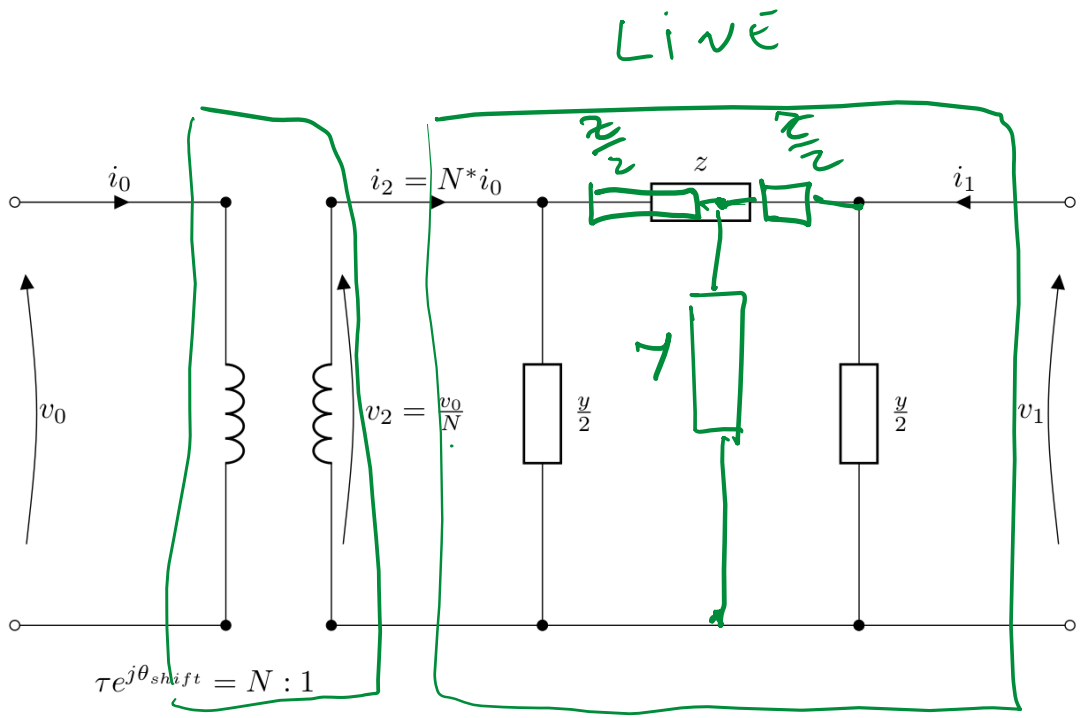
Out[2]:

attribute	bus0	bus1	type	x	r	g	b	s_nom	s_nom_extendable	s_nom_min	...	v_ang_min	v_ang_max
Line #1	Bus #1	Bus #2		0.1	0.01	0.0	0.0	0.0	False	0.0	...	-inf	inf

Transformers

- Transformers are edges In the network (graph) language that connect two different AC busses with different voltages.
- There are two physical models for a transformer:
  - “t” configuration, having two series impedances and one shunt admittance;
  - “pi” configuration, having two shunt admittances and one series impedance.
- Standard transformer types are available here:
  - https://pandapower.readthedocs.io/en/latest/std_types/basic.html#transformers

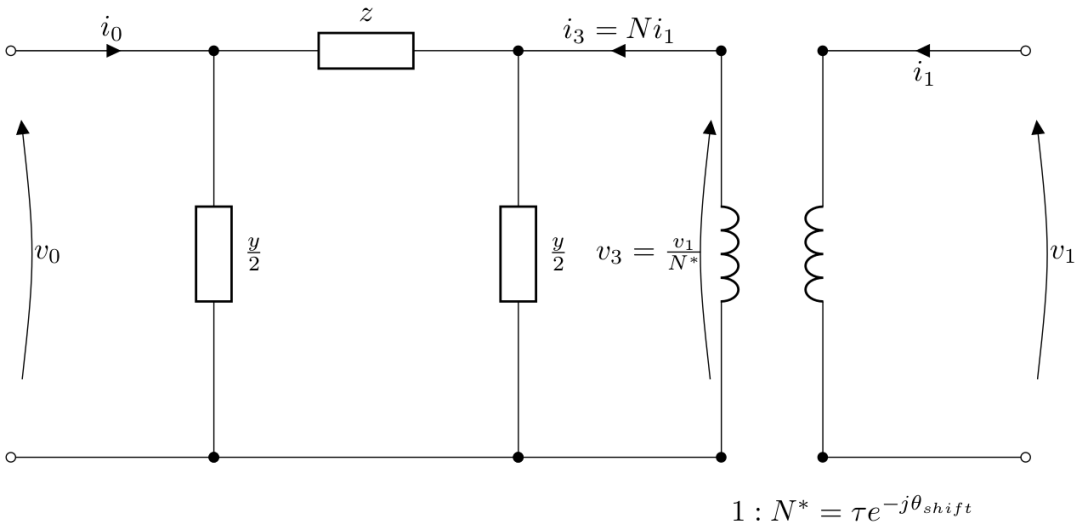
Transformer Models (I)



- Equation:
- $$\begin{bmatrix} i_0 \\ i_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{z} + \frac{y}{2} & -\frac{1}{z}\frac{1}{\tau e^{-j\theta}} \\ -\frac{1}{z}\frac{1}{\tau e^{j\theta}} & \left(\frac{1}{z} + \frac{y}{2}\right)\frac{1}{\tau^2} \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}$$

$pi = \pi$

Transformer Models (II)



- Equation:
- $$\begin{bmatrix} i_0 \\ i_1 \end{bmatrix} = \begin{bmatrix} \left(\frac{1}{z} + \frac{y}{2}\right)\frac{1}{\tau^2} & -\frac{1}{z}\frac{1}{\tau e^{-j\theta}} \\ -\frac{1}{z}\frac{1}{\tau e^{j\theta}} & \frac{1}{z} + \frac{y}{2} \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}$$

Transformers in PyPSA

- Use pypsa.Network.add(class_name='Transformer')

argument	type	unit	default	Description
name	string	-	-	unique name
bus0	string	-	-	The name of the 1st connected bus
bus1	string	-	-	The name of the 2nd connected bus
type	string	-	-	e.g. '0.63 MVA 20/0.4 kV'
x / r	float	Ω	0	Series: z = r + jx
g / b	float	S	0	Shunt: y = g + jb
tap_ratio	float	/unit	1	Ratio of per unit voltages at each bus
tap_side	int	-	0	Which tap is connected directly to a bus
phase_shift	float	degree	0	Voltage phase angle shift.

Links

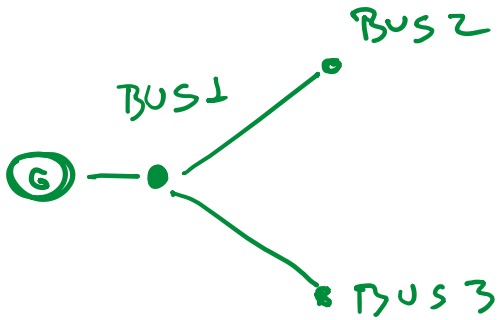
- Links are edges between arbitrary carrier buses.
- They are useful for defining converters between two different carriers (e.g. ‘AC’ to ‘DC’, ‘Wind’ to ‘AC’, etc.).
- The main parameter is the conversion efficiency between the connected buses.
- The efficiency can be set as a time dependent series.

Links in PyPSA

- Use pypsa.Network.add(class_name='Link') to add new lines.
- Parameters:

argument	type	unit	default	Description
name	string	-	-	unique name
bus0	string	-	-	The name of the 1st connected bus
bus1	string	-	-	The name of the 2nd connected bus
carrier	string	-	-	Energy carrier transported. E.g. 'DC'
efficiency	float / Series	-	1	Efficiency of converting from bus0 to bus1
p_nom	float	MVA	0	Limit of power which can pas through.
p_nom_min	float	MVA	0	If p_nom is expandable, its min value.
p_nom_max	float	MVA	inf	If p_nom is expandable, its max value.

Generators



- Generators attach to a single bus and can feed in power.
- It converts energy from its carrier to the carrier-type of the bus to which it is attached.
- Generators can be:
  - conventional generators, providing power anywhere withing a given range;
  - environment-dependent generators, providing power withing limits set as time series.
- For generators, if $P > 0$ the generator is supplying active power to the bus and if $Q > 0$ it is supplying reactive power (i.e. behaving like a capacitor).

Generators in PyPSA (1)

- Use pypsa.Network.add(class_name='Generator')
- Parameters:

argument	type	unit	default	Description
name	string	-	-	unique name
bus	string	-	-	The name of the connected bus
control	string	-	-	The control strategy 'PQ', 'PV', 'Slack'.
carrier	string	-	-	Energy carrier used. E.g. 'gas', 'coal'
p_nom_expandable	bool	-	False	Flag that defines if the power is expandable
p_nom	float	MVA	0	Nominal power.
p_nom_min	float	MVA	0	If p_nom is expandable, its min value.
p_nom_max	float	MVA	inf	If p_nom is expandable, its max value.

Generators in PyPSA (2)

- Use pypsa.Network.add(class_name='Generator')
- Parameters:

argument	type	unit	default	Description
p_min_pu	float / Series	/unit	-1	Min power output per unit of p_nom
p_max_pu	float / Series	/unit	1	Max power output per unit of p_nom
p_set	float	MW	0	Active power set point.
q_set	float	Mvar	0	Reactive power set point.
marginal_cost	float / Series	/MWh	0	Production cost of 1MWh
capital_cost	float	/MW	0	Cost of expanding capacity with 1MW
efficiency_store	float	/unit	1	Efficiency of storage energy.
efficiency_dispatch	float	/unit	1	Efficiency of stored energy dispatching.

percent

0

0

NO STORAGE

Generators in Python with PyPSA

In [3]:
```
grid.add('Generator', 'My gen',
            bus='My bus #0',
            p_set = 100,
            control = 'PQ')

grid.generators
```

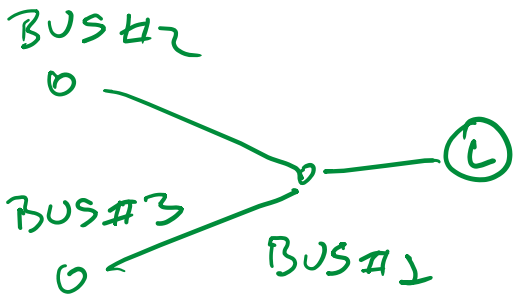
Out[3]:

attribute	bus	control	type	p_nom	p_nom_extendable	p_nom_min	p_nom_max	p_min_pu	p_max_pu	p_set	...	shut_down_cost	min_up_time	min_down_ti
My gen	My bus #0	PQ		0.0	False	0.0	inf	0.0	1.0	100.0	...	0.0	0	

1 rows × 30 columns

Loads

- The load attaches to a single bus and consumes power as a PQ load.
- If $P > 0$ the load is consuming active power from the bus and if $Q > 0$ it is consuming reactive power (i.e. behaving like an inductor).



Loads in PyPSA

- Use pypsa.Network.add(class_name='Generator')
- Parameters:

argument	type	unit	default	Description
name	string	-	-	unique name
bus	string	-	-	The name of the connected bus
carrier	string	-	-	Energy carrier used. E.g. 'AC', 'DC'
p_set	float / Series	MW	0	Active power consumption set point.
q_set	float / Series	Mvar	0	Reactive power consumption set point.
sign	float	-	-1	1 = consumer / -1 = generator

Loads in Python with PyPSA

In [4]:
```
grid.add('Load','My load',
             bus='My bus #1',
             p_set=100)

grid.loads
```

Out[4]:

attribute	bus	carrier	type	p_set	q_set	sign
My load	My bus #1			100.0	0.0	-1.0

Check for Model Convergence

```
In [5]: grid.pf()
```

```
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
INFO:pypsa.pf:Performing non-linear load-flow on AC sub-network SubNetwork 0 for snapshots Index(['now'], dtype='obje
ct')
INFO:pypsa.pf:Newton-Raphson solved in 2 iterations with error of 0.000000 in 0.016957 seconds
```

```
Out[5]: {'n_iter':      0
         now  2,
         'error':                 0
         now  1.164135e-10,
         'converged':         0
         now  True}
```

Model Solutions

In [6]:
```
grid.lines_t.p0
```

Out[6]:

	My line #0	My line #1	My line #2
now	66.667585	-33.333333	-33.333563

In [7]:
```
grid.buses_t.v_ang*180/np.pi
```

Out[7]:

	My bus #0	My bus #1	My bus #2
now	0.0	-0.007892	-0.003946

In [8]:
```
grid.buses_t.v_mag_pu
```

Out[8]:

	My bus #0	My bus #1	My bus #2
now	1.0	0.999986	0.999993

Thank you!