Full Name:

Smart Grids Technologies
Module 2, Lab 2 - 31/03/2025
Load Flow Analysis

1 Organization

1.1 Objectives

This lab session covers the basics of the load flow analysis, numerical solution of the load flow problem and its approximations. It is assumed that you are familiar with the fundamentals of the load flow problem, and that you have completed lab 2.1 about the admittance matrix calculus.

First, we recall the *Newton-Raphson algorithm*, which is commonly used for solving load flow problems. Then, we review several approximation schemes *Ward-Hale*, *Carpentier* and *Stott* approximations.

You are provided with a MATLAB toolbox, in which the main functions are already implemented, but a few blocks of code are still missing. You will be asked to code the missing pieces, perform some load flow simulations, and interpret the obtained results.

1.2 Report

This report will not be graded; however, its submission is mandatory. The purpose of the questions within this document is to enhance your comprehension of the subject matter. Your acquired knowledge from all three laboratories of **Module 2** will be evaluated in Quiz 2, scheduled for Monday, 28 April 2025. The deadline for submission of the reports is Sunday, 6 April 2025, at 23:55. Do not forget to write your full name in the corresponding box at the top of this page.

2 Theory

This section of the lab introduces the load flow problem, the Newton-Raphson method and its approximations for the solution of load flow problem. If you are already familiar with the topics discussed here, feel free to skip, and proceed directly to Sec. 3.

2.1 The Load Flow Problem

Consider an electrical grid whose buses are labeled as $n \in \mathcal{N}$. Let $u \in \mathcal{U} \subsetneq \mathcal{N}$ be buses where *active* and *reactive power* are regulated (typically loads), and $g \in \mathcal{G} \subsetneq N$ buses where *active power* and *voltage magnitude* are regulated (typically generators), where $\mathcal{U} \cap \mathcal{G} = \varnothing$. In addition, there is one *slack bus*. By convention, it is assumed that bus 1 is the slack (i.e., $\mathcal{N} = \{1\} \cup \mathcal{U} \cup \mathcal{G}$). The *load flow* (LF) problem consists of determining the *magnitude* V_n and *angle* θ_n of the *(phase-to-ground) voltage phasors* $\bar{E}_n = E_n \angle \theta_n = E'_n + jE''_n$ in all buses (i.e., the state of the electrical network) for given active and reactive power of the loads $u \in \mathcal{U}$, and active power and voltage magnitude of the generators $g \in \mathcal{G}$. Let P_n/Q_n denote the *active/reactive power* entering the electrical network at bus n. The LF equations link the voltage phasors with the active/reactive power injections. Namely, for $n \in \mathcal{N}$

$$\bar{S}_n = P_n + jQ_n = \bar{E}_n \underline{I}_n = \bar{E}_n \sum_{h \in \mathcal{N}} \underline{Y}_{nh} \underline{E}_h \tag{1}$$

$$= (E'_n + jE''_n) \sum_{h \in \mathcal{N}} (G_{nh} - jB_{nh})(E'_h - jE''_h) \tag{2}$$

where $\bar{Y}_{nh} = Y_{nh} \angle \gamma_{nh} = G_{nh} + jB_{nh}$ is the (n, h)-th element of the nodal admittance matrix $\bar{Y}$. In polar coordinates

$$P_n = E_n \sum_{h \in \mathcal{N}} Y_{nh} E_h \cos(\theta_{nh} - \gamma_{nh}) \tag{3}$$

$$Q_n = E_n \sum_{h \in \mathcal{N}} Y_{nh} E_h \sin(\theta_{nh} - \gamma_{nh}) \tag{4}$$

where $\theta_{nh} = \theta_n - \theta_h$. In rectangular (a.k.a. Cartesian) coordinates

$$P_n = E'_n \sum_{h \in \mathcal{N}} \left\{ G_{nh} E'_h - B_{nh} E''_h \right\} + E''_n \sum_{h \in \mathcal{N}} \left\{ B_{nh} E'_h + G_{nh} E''_h \right\} \tag{5}$$

$$Q_n = -E'_n \sum_{h \in \mathcal{N}} \left\{ B_{nh} E'_h + G_{nh} E''_h \right\} + E''_n \sum_{h \in \mathcal{N}} \left\{ G_{nh} E'_h - B_{nh} E''_h \right\} \tag{6}$$

Looking more closely at (3)–(4) and (5)–(6), observe that there are two equations per bus (i.e, active/reactive power), but potentially four variables per bus (i.e., P_n, Q_n, E_n, θ_n). Thus, one first needs to identify the known and unknown variables of the system.

Which variables are known or unknown depends on the type of bus and the choice of coordinates.

- At *load* or *PQ-buses* $u \in \mathcal{U}$, the active and reactive power are set to P_u^* and Q_u^*, respectively. In polar coordinates

$$P_u^* = E_u \sum_{h \in \mathcal{N}} Y_{uh} E_h \cos(\theta_{uh} - \gamma_{uh}) \tag{7}$$

$$Q_u^* = E_u \sum_{h \in \mathcal{N}} Y_{uh} E_h \sin(\theta_{uh} - \gamma_{uh}) \tag{8}$$

  The unknowns are E_u and θ_u (i.e., two per bus). In rectangular coordinates

$$P_u^* = E'_u \sum_{h \in \mathcal{N}} \left\{ G_{uh} E'_h - B_{uh} E''_h \right\} + E''_u \sum_{h \in \mathcal{N}} \left\{ B_{uh} E'_h + G_{uh} E''_h \right\} \tag{9}$$

$$Q_u^* = -E'_u \sum_{h \in \mathcal{N}} \left\{ B_{uh} E'_h + G_{uh} E''_h \right\} + E''_u \sum_{h \in \mathcal{N}} \left\{ G_{uh} E'_h - B_{uh} E''_h \right\} \tag{10}$$

  The unknowns are E'_u and E''_u (i.e., two per bus).

- At *voltage-controlled* or *PV-buses* $g \in \mathcal{G}$, the voltage magnitude and active power are set to E_g^* and P_g^*, respectively. In polar coordinates

$$P_g^* = E_g \sum_{h \in \mathcal{N}} Y_{gh} E_h \cos(\theta_{gh} - \gamma_{gh}) \tag{11}$$

$$E_g^* = E_g \tag{12}$$

  The only unknown is θ_u (i.e., one per bus). In rectangular coordinates

$$P_g^* = E'_g \sum_{h \in \mathcal{N}} \left\{ G_{gh} E'_h - B_{gh} E''_h \right\} + E''_g \sum_{h \in \mathcal{N}} \left\{ B_{gh} E'_h + G_{gh} E''_h \right\} \tag{13}$$

$$(E_g^*)^2 = (E'_g)^2 + (E''_g)^2 \tag{14}$$

  The unknowns are E'_g and E''_g (i.e., two per bus).

- At the *slack bus* $s = 1$, which is considered as the reference bus of the system, both the voltage magnitude and phase angle are specified. The active and reactive power are not fixed, as this would mean fixing

the network losses, which would imply an overdetermined equation. Hence, there is no unknown variable for this type of bus. Usually, the voltage is set to 1 *p.u.*, and the angle to 0. In polar coordinates

$$E_s = 1 \tag{15}$$

$$\theta_s = 0 \tag{16}$$

In rectangular coordinates

$$E'_s = 1 \tag{17}$$

$$E''_s = 0 \tag{18}$$

If the LF equations are written in polar coordinates, there are $g + 2u$ nontrivial equations[1]. Conversely, if the LF equations are written in rectangular coordinates, there are $2g + 2u$ nontrivial equations.

Due to the fact that LF equations are nonlinear, numerical methods have to be used in order to obtain a solution (with an acceptable tolerance). Such methods are usually iterative, and start from an initial guess of the voltage profile. The initial voltage profile is chosen so that it favors the convergence of the numerical method towards a physically meaningful solution. Typically, a "flat start" is used to initialize the NR method (i.e., all voltage magnitudes equal to 1 *p.u.*, and all phase angles equal to 0 rad).

2.2 The Newton-Raphson Algorithm

The *Newton-Raphson* (NR) method is frequently used for solving the LF equations. Starting from an initial voltage profile defined by $E_n^{(0)}$ and $\theta_n^{(0)}$ ($n \in \mathcal{N}$), the NR method iteratively finds a solution of the LF equations (within a given tolerance). In the following, the main steps of the algorithm are formulated both in polar and rectangular coordinates (the process is also summarized in the lecture notes).

[1] An equations is *trivial* if a variable is simply set to a constant value (e.g., $E_g = E_g^*$).

2.2.1 Formulation in Polar Coordinates

The steps of the NR algorithm at iteration $k+1$ are the following:

1. Compute the power *mismatches* $\Delta P_n^{(k)}$ and $\Delta Q_n^{(k)}$.

$$\Delta P_i^{(k)} = P_i^* - P_i(E_i^{(k)}, \theta_i^{(k)}),\ i \in \mathcal{U} \cup \mathcal{G} \tag{19}$$
$$\Delta Q_i^{(k)} = Q_i^* - Q_i(E_i^{(k)}, \theta_i^{(k)}),\ i \in \mathcal{U} \tag{20}$$

   In other words, active and reactive power mismatches are computed only for the nodes for which we fixed the active (PQ and PV nodes) and reactive power (PQ nodes).

2. Compute the *Jacobian matrix* $\mathbf{J}^{(k)}$ of the LF equations (3)–(4).

$$\mathbf{J}^{(k)} = \begin{bmatrix} \mathbf{J}_{PE} & \mathbf{J}_{P\theta} \\ \mathbf{J}_{QE} & \mathbf{J}_{Q\theta} \end{bmatrix}^{(k)} = \begin{bmatrix} \frac{\partial \mathbf{P}}{\partial \mathbf{E}} & \frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}} \\ \frac{\partial \mathbf{Q}}{\partial \mathbf{E}} & \frac{\partial \mathbf{Q}}{\partial \boldsymbol{\theta}} \end{bmatrix}^{(k)} \tag{21}$$

   The Jacobian relates power mismatches and voltage corrections in magnitude and angle:

$$\begin{bmatrix} \Delta \mathbf{P} \\ \Delta \mathbf{Q} \end{bmatrix}^{(k)} = \begin{bmatrix} \mathbf{J}_{PE} & \mathbf{J}_{P\theta} \\ \mathbf{J}_{QE} & \mathbf{J}_{Q\theta} \end{bmatrix}^{(k)} \times \begin{bmatrix} \Delta \mathbf{E} \\ \Delta \boldsymbol{\theta} \end{bmatrix}^{(k)} \tag{22}$$

3. The goal is to solve the system of equations (22) to obtain the corrections in voltage magnitude and angle. For generator (PV) buses and the slack bus, E_i ($\forall i \in \mathcal{G}$) and E_s are already known. In addition, for the slack bus, θ_s is also known. Therefore, the corresponding elements are excluded from the vector of unknowns corrections. Consequently, we need to reduce the Jacobian matrix and to remove:

   (i) The columns which correspond to:
   - the voltage magnitudes of the generator buses, which are fixed (i.e., $\Delta E_i = 0\ \forall i \in \mathcal{G}$),
   - the slack bus, which is the reference (i.e., $\Delta E_s = 0$ and $\Delta \theta_s = 0$).

   (ii) The rows corresponding to the mismatches for which the power injections are not fixed. These are:
   - the active power for the slack bus (i.e., ΔP_s),

- the reactive power for the slack bus and generator buses (i.e., ΔQ_s and $\Delta Q_i \ \forall i \in \mathcal{G}$).

Accordingly, the reduced Jacobian matrix is a **square** matrix of size $(2|\mathcal{U}| + |\mathcal{G}|) \times (2|\mathcal{U}| + |\mathcal{G}|)$.

4. Compute the *corrections* in magnitude $\Delta E_i^{(k)}$ and angle $\Delta\theta_i^{(k)}$.

$$\begin{bmatrix} \Delta\mathbf{E} \\ \Delta\boldsymbol{\theta} \end{bmatrix}^{(k)} = \left\{ \begin{bmatrix} \mathbf{J}_{PE} & \mathbf{J}_{P\theta} \\ \mathbf{J}_{QE} & \mathbf{J}_{Q\theta} \end{bmatrix}^{(k)} \right\}^{-1} \times \begin{bmatrix} \Delta\mathbf{P} \\ \Delta\mathbf{Q} \end{bmatrix}^{(k)} \tag{23}$$

where $\Delta\mathbf{E}^{(k)}$, $\Delta\boldsymbol{\theta}^{(k)}$, $\Delta\mathbf{P}^{(k)}$, and $\Delta\mathbf{Q}^{(k)}$ are column vectors composed of the elements $\Delta E_i^{(k)}$, $\Delta\theta_i^{(k)}$, $\Delta P_i^{(k)}$, and $\Delta Q_i^{(k)}$, respectively.

5. Update the magnitudes and angles with the corresponding corrections.

$$\begin{bmatrix} \mathbf{E} \\ \boldsymbol{\theta} \end{bmatrix}^{(k+1)} = \begin{bmatrix} \mathbf{E} \\ \boldsymbol{\theta} \end{bmatrix}^{(k)} + \begin{bmatrix} \Delta\mathbf{E} \\ \Delta\boldsymbol{\theta} \end{bmatrix}^{(k)} \tag{24}$$

6. If the mismatches are below a given *tolerance* ε (*convergence criterion*)

$$\max\left(\max(|\Delta\mathbf{P}|), \max(|\Delta\mathbf{Q}|)\right) < \varepsilon \tag{25}$$

then stop. Otherwise, set $k = k + 1$, and go to step 1.

The elements of the Jacobian matrix (21) are the *partial derivatives* of the active and reactive powers w.r.t. the voltage magnitudes and angles. They can be computed from (3)–(4), which yields

$$\frac{\partial P_i}{\partial E_j} = \begin{cases} Y_{ij} E_i \cos(\theta_i - \theta_j - \gamma_{ij}) & (i \neq j) \\ 2Y_{ii} E_i \cos(\gamma_{ii}) + \sum\limits_{h \neq i} Y_{ih} E_h \cos(\theta_i - \theta_h - \gamma_{ih}) & (i = j) \end{cases} \tag{26}$$

$$\frac{\partial P_i}{\partial \theta_j} = \begin{cases} Y_{ij} E_i E_j \sin(\theta_i - \theta_j - \gamma_{ij}) & (i \neq j) \\ -E_i \sum\limits_{h \neq i} Y_{ih} E_h \sin(\theta_i - \theta_h - \gamma_{ih}) & (i = j) \end{cases} \tag{27}$$

$$\frac{\partial Q_i}{\partial E_j} = \begin{cases} Y_{ij} E_i \sin(\theta_i - \theta_j - \gamma_{ij}) & (i \neq j) \\ -2Y_{ii} E_i \sin(\gamma_{ii}) + \sum\limits_{h \neq i} Y_{ih} E_h \sin(\theta_i - \theta_h - \gamma_{ih}) & (i = j) \end{cases} \tag{28}$$

$$\frac{\partial Q_i}{\partial \theta_j} = \begin{cases} -Y_{ij} E_i E_j \cos(\theta_i - \theta_j - \gamma_{ij}) & (i \neq j) \\ E_i \sum\limits_{h \neq i} Y_{ih} E_h \cos(\theta_i - \theta_h - \gamma_{ih}) & (i = j) \end{cases} \tag{29}$$

2.2.2 Formulation in Rectangular Coordinates

The steps of the NR algorithm at iteration $k+1$ are the following:

1. Compute the mismatches w.r.t. active/reactive powers and (squared) voltage magnitudes.

$$\Delta P_i^{(k)} = P_i^* - P_i(E_i'^{(k)}, E_i''^{(k)}),\ i \in \mathcal{U} \cup \mathcal{G} \tag{30}$$

$$\Delta Q_i^{(k)} = Q_i^* - Q_i(E_i'^{(k)}, E_i''^{(k)}),\ i \in \mathcal{U} \tag{31}$$

$$\Delta (E_i^2)^{(k)} = (E_i^*)^2 - (E_i^{(k)})^2,\ i \in \mathcal{G} \tag{32}$$

   In other words, active/reactive power and (squared) voltage magnitude mismatches are computed only for the nodes for which we fixed the active (PQ and PV nodes)/reactive power (PQ nodes) and voltage magnitude (PV nodes), respectively.

2. Compute the *Jacobian matrix* of the LF equations (5)–(6).

$$\mathbf{J}^{(k)} = \begin{bmatrix} \mathbf{J}_{PR} & \mathbf{J}_{PX} \\ \mathbf{J}_{QR} & \mathbf{J}_{QX} \\ \mathbf{J}_{ER} & \mathbf{J}_{EX} \end{bmatrix}^{(k)} = \begin{bmatrix} \frac{\partial \mathbf{P}}{\partial \mathbf{E}'} & \frac{\partial \mathbf{P}}{\partial \mathbf{E}''} \\ \frac{\partial \mathbf{Q}}{\partial \mathbf{E}'} & \frac{\partial \mathbf{Q}}{\partial \mathbf{E}''} \\ \frac{\partial \mathbf{E}^2}{\partial \mathbf{E}'} & \frac{\partial \mathbf{E}^2}{\partial \mathbf{E}''} \end{bmatrix}^{(k)} \tag{33}$$

   The Jacobian relates active and reactive power and squared voltage magnitude mismatches with voltage corrections in real and imaginary parts:

$$\begin{bmatrix} \Delta \mathbf{P} \\ \Delta \mathbf{Q} \\ \Delta (\mathbf{E}^2) \end{bmatrix}^{(k)} = \begin{bmatrix} \mathbf{J}_{PR} & \mathbf{J}_{PX} \\ \mathbf{J}_{QR} & \mathbf{J}_{QX} \\ \mathbf{J}_{ER} & \mathbf{J}_{EX} \end{bmatrix}^{(k)} \times \begin{bmatrix} \Delta \mathbf{E}' \\ \Delta \mathbf{E}'' \end{bmatrix}^{(k)} \tag{34}$$

3. The goal is to solve the system of equations (34) to obtain the corrections of the real and imaginary parts of the nodal voltages. For the slack bus, E_s' and E_s'' are already known. Therefore, we need to reduce the Jacobian matrix and to remove:

   (i) The columns which correspond to the real and imaginary parts of the nodal voltage of the slack bus (i.e., $\Delta E_s' = 0$ and $\Delta E_s'' = 0$),

   (ii) The rows corresponding to the mismatches for which the power injections or voltage magnitudes are not fixed and the identity of the slack bus. These are:

- active and reactive power for the slack bus (i.e., non-specified ΔP_s, ΔQ_s and the identity $\Delta E_s^2 = 0$),
- reactive power for generator buses (i.e., ΔQ_s and $\Delta Q_i \; \forall i \in \mathcal{G}$)
- voltage magnitude (squared) for PQ buses.

The reduced Jacobian matrix is a square matrix of size $(2|\mathcal{U}| + 2|\mathcal{G}|) \times (2|\mathcal{U}| + 2|\mathcal{G}|)$.

4. Compute the corrections of the real and imaginary parts.

$$\begin{bmatrix} \Delta \mathbf{E}' \\ \Delta \mathbf{E}'' \end{bmatrix}^{(k)} = \left\{ \begin{bmatrix} \mathbf{J}_{PR} & \mathbf{J}_{PX} \\ \mathbf{J}_{QR} & \mathbf{J}_{QX} \\ \mathbf{J}_{ER} & \mathbf{J}_{EX} \end{bmatrix}^{(k)} \right\}^{-1} \times \begin{bmatrix} \Delta \mathbf{P} \\ \Delta \mathbf{Q} \\ \Delta(\mathbf{E}^2) \end{bmatrix}^{(k)} \tag{35}$$

5. Update the real and imaginary parts with the corrections.

$$\begin{bmatrix} \mathbf{E}' \\ \mathbf{E}'' \end{bmatrix}^{(k+1)} = \begin{bmatrix} \mathbf{E}' \\ \mathbf{E}'' \end{bmatrix}^{(k)} + \begin{bmatrix} \Delta \mathbf{E}' \\ \Delta \mathbf{E}'' \end{bmatrix}^{(k)} \tag{36}$$

6. Stop if all mismatches are below the specified tolerance.

$$\max\left(\max(|\Delta \mathbf{P}|), \max(|\Delta \mathbf{Q}|), \max(|\Delta(\mathbf{E}^2)|)\right) < \varepsilon \tag{37}$$

Otherwise, set $k = k + 1$, and go to step 1.

The elements of the Jacobian matrix (33) are the *partial derivatives* of the active and reactive powers with respect to the real and imaginary parts

of the voltage phasors. They can be computed from (5)–(6), which yields

$$\frac{\partial P_i}{\partial E'_j} = \begin{cases} G_{ij}E'_i + B_{ij}E''_i & (i \neq j) \\ 2G_{ii}E'_i + \sum\limits_{h\neq i}\{G_{ih}E'_h - B_{ih}E''_h\} & (i = j) \end{cases} \tag{38}$$

$$\frac{\partial P_i}{\partial E''_j} = \begin{cases} -B_{ij}E'_i + G_{ij}E''_i & (i \neq j) \\ 2G_{ii}E''_i + \sum\limits_{h\neq i}\{B_{ih}E'_h + G_{ih}E''_h\} & (i = j) \end{cases} \tag{39}$$

$$\frac{\partial Q_i}{\partial E'_j} = \begin{cases} -B_{ij}E'_i + G_{ij}E''_i & (i \neq j) \\ -2B_{ii}E'_i - \sum\limits_{h\neq i}\{B_{ih}E'_h + G_{ih}E''_h\} & (i = j) \end{cases} \tag{40}$$

$$\frac{\partial Q_i}{\partial E''_j} = \begin{cases} -G_{ij}E'_i - B_{ij}E''_i & (i \neq j) \\ -2B_{ii}E''_i + \sum\limits_{h\neq i}\{G_{ih}E'_h - B_{ih}E''_h\} & (i = j) \end{cases} \tag{41}$$

$$\frac{\partial E_i^2}{\partial E'_j} = \begin{cases} 0 & (i \neq j) \\ 2E'_i & (i = j) \end{cases} \tag{42}$$

$$\frac{\partial E_i^2}{\partial E''_j} = \begin{cases} 0 & (i \neq j) \\ 2E''_i & (i = j) \end{cases} \tag{43}$$

2.3 References

1. Mario Paolone, "Numerical solution of the Load Flow Problem Formulated via the Nodal Analysis", EPFL (lecture in the course "Smart Grids Technologies"), 2025.

3 Exercises

3.1 Introduction to the MATLAB Toolbox

In the following, a brief tutorial on the MATLAB toolbox for the load flow computation is given. Download the MATLAB code via the below link:

```
https://moodle.epfl.ch/mod/folder/view.php?id=1290267
```

Unzip the folder and open the folder LF and open `main.m`. The script consists of five blocks, whose functionality is explained subsequently.

Step 1: In the first step, the nodal admittance matrix as well as the base power and voltage are imported. These data are provided for a "big" and a "small" test system. The user has to specify which data shall be loaded:

```
% ! Define the network to use !
network = 'big'; % either 'small' or 'big'
```

Step 2: In the second step, the profiles of the absorbed and injected powers are imported. The generator sign convention is used: *injected* powers are *positive*, *absorbed* powers are *negative*. More precisely, we provide you with profiles for a single timestep and for an entire day (i.e., 24 hours sampled every 15 minutes). The user has to specify which profiles shall be loaded:

```
% ! Define the profile type !
% if network == 'small' -> 'daily' or 'single',
% if network == 'big' -> 'daily'
profile_type = 'daily'; % either 'single' or 'daily'
```

Step 3: In the third step, the load flow simulation is configured. The *slack node* (`idx.slack`), the *PQ nodes* (`idx.pq`), and the *PV nodes* (`idx.pv`) are given by the respective test system, and are thus hardcoded in the script. The user has to specify in which coordinate system (i.e., rectangular or polar) the load flow problem is formulated, and which initial point is used for the numerical solution (i.e., a flat start or the solution of a previous calculation). To this end, the variables `coordinate_type` and `start_type` need to be set accordingly:

```
% ! Customize the load flow simulation !

% Choose LF formulation either 'rectangular' or 'polar'.
coordinate_type = 'rectangular';

% if coordinate_type='daily' -> 'flat' or 'previous',
% if coordinate_type='single' -> 'flat' or 'bad'
start_type = 'previous';
```

Finally, the user needs to set the *maximum number of iterations* `n_max` and the *convergence tolerance* `tol` for the NR algorithm:

```
% ! Enter Newton-Raphson algorithm parameters !

% maximum number of iterations
Parameters.n_max = 100;

% convergence tolerance
Parameters.tol = 1e-7;
```

Step 4: In the fourth step, the NR algorithm is executed. First, the initial voltage profile `E_0` are set based on the value of the variable `start_type`. Then, the functions which implement the NR method in rectangular or polar

coordinates, which are named `NR_rectangular` and `NR_polar`, are called. Each functions contain a `for` loop, which performs the NR algorithm as described in Sec. 2.2.

Step 5: In the fifth and final step, the obtained results are plotted.

3.2 Implementation in MATLAB

Now, it is your turn to write code. You need to complete the functions `NR_polar` and `NR_rectangular`. Both functions have the same interface. Namely, they take the following inputs

```
% INPUT
% - Y           nodal admittance matrix
% - S_star      given complex powers (active/reactive powers)
% - E_star      given voltage magnitudes
% - E_0         initial voltages (phasors)
% - idx_slack   index of the slack bus
% - idx.pq      indices of the PQ buses
% - idx.pv      indices of the PV buses
% - Parameters.tol         tolerance for convergence criterion
% - Parameters.n_max       maximum number of iterations
```

and return the following outputs

```
% OUTPUT
% - E           solution voltages (phasors)
% - J           Jacobian at the solution
% - n_iter      number of iterations
```

Please write/change the code in the places that are marked as follows

```
% ****************************************
% ! write your own code here !
% [Instructions]
% ****************************************
```

by following the set of instructions [Instructions], and leave the rest of the code untouched – except for the parameters that you are asked to change.

Q1/ Answer the following questions:

1. The key differences between a slack bus, a PQ bus and a PV bus - which quantities (voltage, active power, reactive power) are fixed or variable in each bus type?

2. The physical meaning behind these bus types - categorise following buses with injections as "'slack", "PV" or "PQ" bus. i) bus connecting a big synchronous generator, ii) bus providing electricity to a steel industry, iii) bus connected to a small hydro power plant through a synchronous generator iv) bus providing electricity to the distribution network of EPFL campus v) bus with zero active and reactive power.

[A1]

Q2/ Complete the function `NR_polar`, i.e., the calculation of the mismatches, the modifications to correctly compute the Jacobian matrix and updating load flow solutions, using the instructions given in the code. Paste *only* the modifications you added to the original code here (i.e. modified code lines).

[A2]

Q3/ Complete the function `NR_rectangular`, i.e., the calculation of the mismatches, the modifications to correctly compute the Jacobian matrix and updating load flow solutions, using the instructions given in the code. Paste *only* the modifications you added to the original code here (i.e. modified code lines).

[A3]

3.3 Validation using EMTP-RV

In order to verify that your implementation is correct, you will now compare the solutions obtained using your own code and EMTP-RV. EMTP-RV is a professional software for the simulation and analysis of power systems, both in steady-state and transient conditions. You can find it installed on the VM you have already used in Module 1 labs.

Open the file `load_flow.ecf` in the folder `emtp`. The network topology and the location of the loads are already defined, but the active and reactive powers of the loads need to be specified. If you double-click on the arrow representing a load, the window shown in Fig. 1 opens. You can enter the load data into the table. ***Attention:*** *EMTP-RV considers the values you enter as balanced three-phase loads, whereas the MATLAB code is based on an equivalent single-phase network. Therefore, you need to divide the active/reactive powers by* 3 *when entering them into the table.*

When you have configured loads, press the button *StartEMTP* (see Fig. 1, red circle). Once EMTP-RV has finished, press the button *Load-Flow web* (see Fig. 1, green circle). This will show LF results in your browser under *Show Node Voltages*[2]. Now, run the MATLAB code once with each version of the NR algorithm for the small system. For this analysis, set

[2]Note that the EMTP-RV results are three-phase, whereas your MATLAB code computes a single-phase equivalent. So, you should compare your results with phase *a* of the EMTP-RV output.

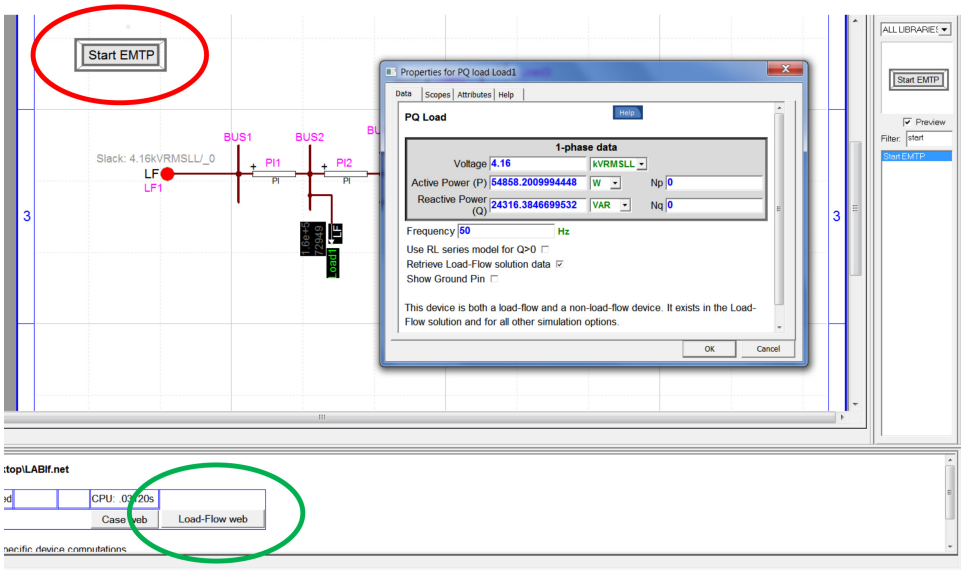

Figure 1: Using EMTP-RV to insert load values at the network buses.

`network='small'`, set `profile_type='single'` and `start_type='flat'`.

Q4/ EMTP-RV returns phase-to-ground rather than phase-to-phase voltages, and amplitudes rather than RMS values. Moreover, EMTP works in absolute units instead of p.u.. How do the results of the MATLAB code have to be transformed such that they are comparable to those of EMTP? Do the MATLAB results agree with the EMTP ones?

[A4]

3.4 Sensitivity Analysis and Performance Evaluation

3.4.1 Impact of the Initial Point on the Solution

Now, you will investigate the impact of the initial point on the solution found by the NR method. To this end, set `network='small'`, `profile_type='single'` and `start_type='bad'`. The last option initializes the phase angle of the voltage at bus 3 to $-\pi/4$ instead of 0 (i.e, not a flat start).

Q5/ Run the LF simulation using either the polar or rectangular formulation.

1. Does the NR method converge? If yes, in how many iterations?
2. Does the solution match the one in Q4? If not,
   - Comment on whether the obtained voltage magnitudes and phase angles in Q4 have a physical meaning or not?
   - Considering that both simulations in Q4 and Q5 use the same network and nodal power injections, how can you explain the differences between the solutions (i.e. nodal voltages (states) outputted from the NR algorithms)?

[A5]

3.4.2 Impact of the Initial Point on the Convergence Speed

Now, you will examine how the initial point affects the convergence speed of the NR method. Set `network='big'` and `profile_type='daily'`, so that the daily profiles are used.

Q6/ Use the rectangular formulation. Run the code two times. First, set `start_type='flat'`, so that a flat voltage start is used as initial point

in every timestep. Then, set `start_type='previous'`, so that the solution from the previous timestep is used instead. For each case, look at Fig. 3 generated by the MATLAB code, which shows the number of iterations `n_iter` and the execution time `t_exec` per timestep. Are there differences between Figs. 3 of both cases? If yes, how do you explain those differences (i.e. why is one case converging faster than the other?) ?

[A6]

3.4.3 Rectangular versus Polar Formulation

Now, you will evaluate how the coordinate system in which the load flow problem is formulated influences the performance of the NR method (i.e., in terms of number of iterations `n_iter` and execution time `t_exec`). Set `network='big'`, `profile_type='daily'` and `start_type='previous'`.

Q7/ Run the code twice, once with the rectangular and once with the polar formulation. For each case, look at Fig. 3 generated by the MATLAB code, which shows the number of iterations `n_iter` and the execution time `t_exec` per timestep. Are the results *exactly* the same? If not, how do you explain those differences (i.e. why are there different convergence speeds and number of iterations depending on the formulation?) ?

[A7]

3.4.4 Interpretation of the Load Flow Results

Finally, you shall interpret the results of the daily simulations. To this end, keep `profile_type='daily'`, and `start_type='previous'`.

Q8/ Effect of loading: run the LF method for the small network (`network='small'`). You may use either formulation. Analyse the profiles of voltage magnitude, and phase angle (i.e., Fig. 2 generated by the MATLAB code) for following cases. Line 103 in the file `main.m`, modify the factors `gen_scale, load_scale` to study effect of loading on load flow solutions.

1. **Zero loading and generation:** set the injections per node to 0, i.e., `gen_scale = 0, load_scale = 0`.
2. Increase generation per node by a factor of 5, i.e., `gen_scale = 5, load_scale = 1`.
3. Increase load per node by a factor of 5, i.e., `gen_scale = 1, load_scale = 5`.

Explain how and why the shape and values of voltage magnitude and phase angles change from the nominal case (`gen_scale = 1, load_scale = 1`) for cases 1, 2 and 3.

[A8]

Q9/ Run the LF method for the big network (`network='big'`) in *rectangular coordinates*. Look at the different subplots in Fig. 2 generated by the MATLAB code:

1. Look at the subplot *PV Nodes - Voltage Magnitudes*. Are these values

correct? If yes, explain why they are not all *exactly* equal (hint: think about the NR formulation that is used).

2. As explained in the theory part, the voltage magnitudes at PV nodes are fixed. How is this achieved physically (i.e. what physical quantity needs to be controlled)? Which subplot shows this?

3. Compute the sum of the active and reactive powers of all the nodes (`sum(S)`) in the network. Explain why the sum (both real and imaginary) is non-zero. How the sum changes for the **zero loading case** (i.e., `gen_scale = 0, load_scale = 0`.)

[A9]

3.5 The Load Flow Approximations

Now, we review and implement different Load flow approximations. They are *(i) Ward-Hale approximation*, *(ii) Carpentier approximation* and *(iii) Stott approximation.*. You are provided with a MATLAB toolbox in which the three main functions are already implemented. You will be asked to compare the three approximations with the actual solution and interpret the obtained results.

3.5.1 Ward-Hale

The Ward - Hale approximation is related to the application of the Newton-Raphson method to the Load-Flow solution using Cartesian rectangular coordinates.

According to this approximation, we consider the variations ΔP and ΔQ

of the powers injected into the network in a generic node depending only on the voltage of that node, i.e.

$$\frac{\partial P_i}{\partial E'_j} = 0 \qquad (i \neq j) \tag{44}$$

$$\frac{\partial P_i}{\partial E''_j} = 0 \qquad (i \neq j) \tag{45}$$

$$\frac{\partial Q_i}{\partial E'_j} = 0 \qquad (i \neq j) \tag{46}$$

$$\frac{\partial Q_i}{\partial E''_j} = 0 \qquad (i \neq j) \tag{47}$$

With this approximation, all the Jacobian submatrices of $\mathbf{J}$ in (35) are purely diagonal, as they contain non-zero elements only on the diagonal since the matrices $\mathbf{J}_{ER}, \mathbf{J}_{EX}$ were already diagonal.

3.5.2 Carpentier

The Carpentier approximation assumes that the active powers injected into the nodes depend only on the phases of the voltages and that the reactive powers depend only on the modules of the voltages (i.e. active-reactive decoupling). Decoupling between the active power variables (i.e. voltage phases) and reactive powers (i.e. voltage modules) results in

$$\mathbf{J}_{PE} = 0; \tag{48}$$

$$\mathbf{J}_{QT} = 0; \tag{49}$$

$$\begin{bmatrix} \mathbf{J}_{PE} & \mathbf{J}_{P\theta} \\ \mathbf{J}_{QE} & \mathbf{J}_{Q\theta} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{J}_{P\theta} \\ \mathbf{J}_{QE} & \mathbf{0} \end{bmatrix} \tag{50}$$

3.5.3 Stott

In this approximation, the following assumptions are considered:

- decoupling between the active power variables (i.e. voltage phases) and reactive powers (i.e. voltage modules), i.e. $\mathbf{J}_{PE} = 0$ and $\mathbf{J}_{QT} = 0$.
- $B_{il}\cos(\theta_{il}) \approx B_{il}$ since θ_{il} are small, i.e. $\cos(\theta_{il}) \approx 1$
- $G_{il}\sin(\theta_{il}) << B_{il}$ since the values of G_{il} are extremely small

- $Q_i << B_{ii}V_i^2$

Due to these simplifications, the partial derivatives can be simplified as following:

$$\frac{\partial P_i}{\partial \theta_j} = \begin{cases} -B_{ij}E_iE_j & (i \neq j) \\ -B_{ij}E_i^2 & (i = j) \end{cases} \Rightarrow -B_{ij}E_iE_j \qquad \forall j. \tag{51}$$

$$\frac{\partial Q_i}{\partial E_j} = \begin{cases} -B_{ij}E_i & (i \neq j) \\ -B_{ii}E_i & (i = j) \end{cases} \Rightarrow -B_{ij}E_i \qquad \forall j. \tag{52}$$

With further assumption of i) approximating $E_j = 1$ p.u. for $i \neq j$ and ii) neglecting the shunt parameters in (51), (52), the Jacobian can be further simplified by dividing the active and reactive power deviations with E_i as

$$\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{B}' \\ \mathbf{B}'' & \mathbf{0} \end{bmatrix} \tag{53}$$

where,

$$\mathbf{B}' = \mathbf{B}'' = -\mathbf{B} \tag{54}$$

As a consequence, the NR update for $k-$th iteration becomes

$$\begin{bmatrix} \Delta\mathbf{E} \\ \Delta\boldsymbol{\theta} \end{bmatrix}^{(k)} = \begin{bmatrix} \mathbf{0} & -\mathbf{B} \\ -\mathbf{B} & \mathbf{0} \end{bmatrix}^{-1} \times \begin{bmatrix} \frac{\Delta\mathbf{P}}{\mathbf{E}} \\ \frac{\Delta\mathbf{Q}}{\mathbf{E}} \end{bmatrix}^{(k)} \tag{55}$$

Thanks to the Stott approximation, it can be seen that the Jacobian matrix $\mathbf{J}$ in (55) is a constant and does not need to be updated within the NR iterations.

3.6 Numerical simulations in MATLAB

Open the folder "**LFApproximations**" and open the script `main.m`. The script consists of six blocks, whose functionality is explained subsequently.

Step 1: In the first step, the line data as well as the base power and voltage are imported, they are used to compute the admittance matrices. These line data are provided for a "high voltage", a "medium voltage" and a "low voltage" test system. The text file contains the information on topology in the first two columns, resistances (Ω/km) in the third column, reactances (Ω/km) in the fourth column, shunts (μS/km) in the fifth column and length of the lines in the final column.

The user has to specify which data shall be loaded:

```
% ! Define the network to use !
network = 'high voltage'; % either 'high voltage' or
 'medium voltage' or 'low voltage'
```

The code for computing the admittance matrix is provided. The function `computeY` computes the admittance matrix of the system. The input to this function is `text_file` where the linedata is saved, and base values of power and voltages `Ab` and `Vb`.

Step 2: In the second step, the profiles of the absorbed and injected powers are imported. The generator sign convention is used: *injected* powers are *positive*, *absorbed* powers are *negative*. More precisely, we provide you with profiles for a single timestep and for an entire day (i.e., 24 hours sampled every 15 minutes). The user has to specify which profiles shall be loaded. We provide "daily" profiles for the first part of the exercise corresponding to each test system.

Step 3: In the third step, the load flow simulation is configured. The *slack node* (`idx.slack`), the *PQ nodes* (`idx.pq`), and the *PV nodes* (`idx.pv`) are given by the respective test system, and are thus hardcoded in the script. In this exercise we only use PQ nodes.

The user has to specify the approximation of the load flow solver is to be used.

```
 % ! Customize the load flow simulation !

% Choose LF solver  'wardhale' or 'carpentier' or 'stott' for
% 'profile_type = 'daily'
```

Finally, the user needs to set the *maximum number of iterations* `n_max` and the *convergence tolerance* `tol` for the NR algorithm. We fix to following for this lab.

```
% ! Enter Newton-Raphson algorithm parameters !

% maximum number of iterations
n_max = 100;

% convergence tolerance
tol = 1e-7;
```

Step 4: In the fourth step, the NR algorithm with different approximations are executed. The functions implement the NR method while implementing the "wardhale", "carpentier", or "stott" approximations.

Step 5: In this step we compare with the "true" load flow solutions computed with `NR_polar.m` (**you can copy the function you implemented earlier or re-fill the one provided in the template code**). The "true" quantities are saved with variables named as `_true`.

Step 6: In this step, we visualise the results and compute the error on the magnitudes and phase of the nodal voltages with different approximations. We compute root mean square error (rmse) and maximum absolute errors on the voltage magnitudes and phases.

Now, run the MATLAB code once with each version of the NR algorithm for the high voltage system with different LF approximation methods. For this analysis, set `network='high voltage'`, set `profile_type='daily'`.

Q10/ Compare the voltage magnitude and phase of all the approximations with "true" values computed with `NR_polar`.

1. Report the rmse and max absolute of error in voltage magnitude (in pu) and phase (deg) for each approximation method using metrics defined in Step 6 of the `main` code. Which approximation is the best performing in terms of error?

2. Report the mean and max number of NR iterations for each approximation method.

[A10]

3.7 High voltage *vs.* medium voltage *vs.* low voltage networks

Till now we worked with high voltage transmission network. Now we will switch to medium and low voltage distribution networks. We will examine the difference in performances of the Load flow approximations with different types of the networks.

Q11/ Here, we compare the solutions obtained by LF approximations on three different networks.

1. List the three differences in performance of the LF approximation methods on "medium voltage" and "low voltage" compared to "high voltage" networks. (*hint: differences related to convergence, error on solutions and NR iterations.*)

2. Do the LF approximation methods perform better or poorly on "medium voltage" and "low voltage" networks? Explain why. (*hint: look at the linedata for* "high-voltage,"medium voltage" *and* "low voltage" *networks, especially the ratio of resistance to reactance (R/X).*)

[A11]