Review

REVISTA Ingeniería

Power Flow in Unbalanced Three-Phase Power Distribution Networks Using Matlab: Theory, analysis, and quasi-dynamic simulation

Flujo de potencia en redes de distribución eléctrica trifásicas no equilibradas utilizando Matlab: Teoría, análisis y simulación cuasi-dinámica

Alejandro Garcés-Ruiz [1]

[1] Universidad Tecnológica de Pereira. Department of Electric Power Engineering.

Received: 2nd-April-2022. Modified: 28th-April-2022. Accepted: 23th-May-2022

Abstract

Context: The power flow is a classical problem for analyzing and operating power distribution networks. It is a challenging problem due to a large number of nodes, the high r/x ratio -typical in low voltage networks- and the unbalanced nature of the load.

Method: This paper review four methods for power flow analysis, namely: the conventional Newton's method, Newton's method in a complex domain, the fixed-point algorithm using Y_{bus} representation, and the backward-forward sweep algorithm. It is well-known that Newton's method has quadratic convergence, whereas the backward-forward sweep algorithm has linear convergence. However, the formal analysis of this convergence rate is less known in the engineering literature. Thus, the convergence of these methods is presented in theory and practice.

Results: A set of simulations in the IEEE 900 node test system is presented. This system is large enough to demonstrate the performance of each algorithm. In addition, a Matlab toolbox is presented for making numerical simulations both for the static case and for quasi-dynamic simulations.

Conclusions: Fixed point algorithms were faster than Newton's methods. However, the latter required less number of iterations.

Keywords: load flow, Newton's method, Backward-forward algorithm, power flow, quasi-dynamic simulation.

Acknowledgements: This work was founded by the vicerrectoria de investigaciones y innovacion y extension, Universidad Tecnologica de Pereira. Proyecto ODiN (Optimal Operation of Distribution Networks).

Language: English.

Cite as: A. Garcés-Ruiz. "Power Flow in Unbalanced Three-Phase Power Distribution Networks Using Matlab: Theory, analysis, and quasi-dynamic simulation". *Ing*, vol. 27, no. 3, 2022. e19252.
https://doi.org/10.14483/23448393.19252

Resumen

Contexto: El flujo de potencia es un problema clásico para la operación de redes de distribución de energía. Es un problema desafiante debido a la gran cantidad de nodos, la alta relación r/x típica de las redes de baja tensión y la naturaleza desequilibrada de la carga.
Métodos: Este documento revisa cuatro métodos para el análisis de flujo de potencia, a saber: el método de Newton convencional, el método de Newton en un dominio complejo, el algoritmo de punto fijo que utiliza la representación de admitancia nodal (Y_{bus}) y el algoritmo de barrido hacia atrás y hacia adelante. Se presenta la convergencia de estos métodos en teoría y práctica. Es bien sabido que el método de Newton tiene convergencia cuadrática, mientras que el algoritmo de barrido hacia atrás y adelante tiene convergencia lineal. Sin embargo, el análisis formal de esta tasa de convergencia es menos conocido en la literatura de ingeniería.
Resultados: se desarrolla una caja de herramientas de Matlab para realizar simulaciones numéricas tanto en el caso estático como en una simulacion cuasi-dinámica usando el sistema de pruebe IEEE de 900 nodos.
Conclusiones: Los algoritmos de punto fijo resultaron mas rapidos. No obstante, los algoritmos basados en Newton requirieron menor número de iteraciones.
Palabras clave: flujo de carga, metodo de Newton, algoritmo de barrido iterative, flujo de potencia, simulación cuasi-dinámica.
Agradecimientos: este trabajo es patrocinado por la vicerrectoria de investigaciones y innovacion y extension, Universidad Tecnologica de Pereira. Proyecto ODiN (Optimal Operation of Distribution Networks)
Idioma: Español.

1. Introduction

Power flow is a standard method for analyzing and operating power systems both at high voltage levels and in power distribution networks. It takes nodal power information and returns the system's state, which is usually represented by nodal voltages. The problem consists of a large set of highly nonlinear algebraic equations that requires efficient numerical methods to be solved in practice [1]. Newton's method is undoubtedly the most common approach for high power applications. It searches for the solution using successive linear approximations via Taylor's expansions. These approximations require calculating a Jacobian matrix in each iteration. However, calculating this Jacobian matrix and solving the corresponding linear system may be computationally expensive. Therefore, particular approximations have been proposed according to the type of network. For example, in high-voltage power transmission networks, the resistance value tends to be lower than the inductance of transmission lines, so the Jacobian matrix can be reduced to a constant matrix directly related to the imaginary part of the nodal admittance matrix. This matrix can be efficiently factorized using lower–upper decomposition or Cholesky's factorization in order to solve the linear system quickly ([2] for a complete review of these methods). These quasi-Newton methods are known as *decoupled* and *fast-decoupled* Newton's methods [3]. The most simplified version of the problem consists of a linear system that neglects voltage variations. This method is called *DC-load flow* due to the analogy to the problem of finding voltages in a DC network [4]. Unfortunately, that kind of approximation is not valid in power distribution networks, and the entire Jacobian must therefore be constructed in each iteration [4].

Newton's method (without any approximation of the Jacobian matrix) has quadratic convergence, –even in power distribution networks– when initialized close to the solution. This property has been analyzed mainly in the context of DC grids [5], although it can be extended to AC networks. This method can also be defined in the complex domain using Wirtinger's derivatives [6]. However, the algorithm is still time-consuming in both cases due to the construction and inversion of the Jacobian matrix, which is why Jacobian-free algorithms are used in power distribution networks. These *ad-hoc* algorithms include the backward-forward sweep method and the current injection method [7].

The backward/forward sweep algorithm is the standard in power distribution networks[1]. This algorithm considers power distribution networks as radial, and it executes a two-step iteration based on Kirchhoff's laws: first, line currents are calculated in a backward sweep, *i.e.*, starting from the last node in the direction to the substation; then, voltages are calculated in a forward sweep, *i.e.*, from the substation to the final user. This simple method has been used in academic and commercial software due to its flexibility and efficiency [9]. It allows solving single-phase and three-phase unbalanced systems. Moreover, it may be easily modified to include voltage regulators and other compensation components [10].

A less known aspect of the backward/forward sweep algorithm is its theoretical analysis [11]. The algorithm is, in essence, a fixed-point iteration that guarantees a linear convergence (in contradistinction to the quadratic convergence of Newton's method). This algorithm can be represented in matrix form, thus allowing for a simple implementation in interpreted languages such as Matlab or Python. Moreover, a matrix representation allows for a straightforward analysis [12].

On the other hand, modern analysis in power distribution networks requires simulations in a time window while considering variations in generation and load. This type of simulation, known as quasi-dynamic time series analysis, requires the evaluation of thousands of power flow scenarios, so efficient algorithms are required today more than ever [13].

Despite being a classic problem, power flow is still under research [14], where neural networks [15] and distributed implementations are two areas of constant development [16], [17]. Its implementation in large power systems is also a challenge that is constantly studied [18], as well as its practical implementation [18]. Although using artificial intelligence techniques can be justified in some specific contexts, the scientific community has heavily criticized its abuse [19]. A classic approach is always preferable since it starts from understanding the physical phenomenon and ensures general mathematical properties that cannot be assured by means of artificial intelligence. Therefore, it is important to review and analyze classic problems and their solutions.

This paper analyzes four power flow algorithms for power distribution networks, namely the conventional Newton's method, the Newton's method in a complex domain, the fixed-point algorithm using Y_{bus} representation, and, the backward-forward sweep algorithm. Contributions are twofold: first, the algorithms are presented in general form, and their convergence is analyzed in theory and practice; and second, a toolbox in Matlab is developed to compare each formulations. To the best

[1] See [8] for one seminal work in this field, proposed by Prof. Céspedes from Colombia National University.

of the author's knowledge, there is no other toolbox in Matlab that deals with unbalanced power distribution networks as presented herein.

The rest of this paper is organized as follows: Section 2 presents the basic formulation of the power flow problem for three-phase unbalanced systems; Section 3 presents Newton's method, both in its real formulation and in the complex domain; after that, the fixed-point algorithm and the backward-forward sweep method are presented in Section 4; Section 5 provides a review of the fundamental theoretical analysis of these algorithms; and Section 6 shows numerical experiments using the IEEE 900-node test system. The paper ends with conclusions in Section 7 and a brief appendix about the Matlab implementation.

2. Problem formulation

A power distribution network is represented by an oriented graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, with $\mathcal{N}$ being the set of nodes and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ the set of branches. The size of $\mathcal{N}$ is $n = |\mathcal{N}|$. Uppercase letters represent vectors and matrices, while lowercase letters represent single variables or vector/matrix entries. Thus, the nodal voltage is given by $V_{\mathcal{N}} = [v_k] \in \mathbb{C}^n$ and the nodal power is given by $S_{\mathcal{N}} = [s_k] \in \mathbb{C}^n$. All variables are considered to be complex hereafter unless otherwise specified. The complex conjugate of x is represented by x^*. This operation is only the conjugate without transposing when applied to a matrix[2]. With a slight abuse of notation, X/Y represents the element-wise array division between vectors X and Y of the same size.

Nodal voltages and nodal currents are related by the following matrix equation:

$$I_{\mathcal{N}} = Y V_{\mathcal{N}}, \tag{1}$$

where Y is the nodal admittance matrix $Y = [y_{km}] \in \mathbb{C}^{n \times n}$. Each entry of this matrix is constructed as follows:

$$y_{km} = \left\{ \begin{array}{cc} \sum\limits_{l \in \mathcal{E}} y_l & \text{for } k = m \\ -y_l & \text{for } k \neq m \text{ and, } l = (k, m) \end{array} \right\}. \tag{2}$$

This equation indicates, in simple terms, that the entries in the diagonal collect the admittance of all the lines connected to a node, and that the off-diagonal terms collect the negative of the corresponding line admittance. Another way to construct the nodal admittance matrix is presented below:

$$Y = A^\top Y_{\mathcal{E}} A, \tag{3}$$

where A is the node-to-branch incidence matrix associated to $\mathcal{G}$ and $Y_{\mathcal{E}} = \text{diag}(y_l) to \forall l \in \mathcal{E}$. The nodal power is represented in terms of the nodal current, as given in Eq. (4):

$$s_k^* = \sum_{m=1}^{n} y_{km} v_k^* v_m, \tag{4}$$

[2]Note that x^* is different from the operation x' in Matlab, since, in the latter, this operation returns the conjugate-transpose.

where s_k^* indicates the complex conjugate of s_k. This equation can be split into real and imaginary parts, as given below:

$$p_k = \sum_{m=1}^{n} g_{km} u_k u_m \cos(\theta_{km}) + b_{km} u_k u_m \sin(\theta_{km}), \tag{5}$$

$$q_k = \sum_{m=1}^{n} g_{km} u_k u_m \sin(\theta_{km}) - b_{km} u_k u_m \cos(\theta_{km}), \tag{6}$$

where $y_{km} = g_{km} + jb_{km}$,$s_k = p_k + q_k j$ and $v_k = u_k e^{j\theta}$, with p_k, q_k, u_k and θ_k being real variables. The power flow consists of finding v_k given the value of s_k at all nodes except the substation, as the voltage of that node is already known. The problem is clearly nonlinear and highly complex, so numerical methods are required to find a solution. Note that Eqs. (5) and (6) are completely equivalent to (4), although the latter is undoubtedly a more compact representation. Eq. (4) is known as the complex representation of the power flow, whereas (5)-(6) is a real representation thereof.

On the other hand, power distribution networks are unbalanced, and hence this model must be extended accordingly. Thus, each node in the graph represents three nodes in the system, which are labeled as A, B, and C, as depicted in Fig. 1. Likewise, each branch represents a three-phase line section with its corresponding 3×3 impedance matrix. The algebraic structure of the problem is the same for three-phase systems; only the size is increased.

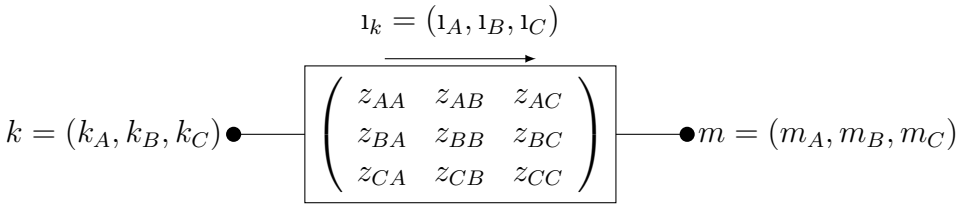

Figure 1. Example of a three-phase line for modeling three-phase unbalanced power distribution networks

The three-phase nodal admittance matrix $Y \in \mathbb{C}^{3n \times 3n}$ is constructed in the same way as in Eq. (2), but, in this case, y_l is a 3×3 block matrix. There is also an equivalent formulation using the node-to-branch incidence matrix, as given in Eq. (7).

$$Y = (A \otimes \mathbb{I}_3)^\top Y_{\mathcal{E}} (A \otimes \mathbb{I}_3), \tag{7}$$

where $\mathbb{I}_3$ represents the identity matrix of size 3, and $\otimes$ is the Kronecker's product. Both formulations return a matrix in which entries 1 to n correspond to phase A, entries $n+1$ to $2n$ correspond to phase B, and entries $2n+1$ to $3n$ correspond to phase C. From now on, the three-phase case is considered in this article since it is the most general.

The substation consists of three nodes corresponding to each phase. These nodes are represented by the set $\mathcal{S} \in \mathcal{N}$. The remaining nodes are $\mathcal{R} = \mathcal{N} - \mathcal{S}$. Then, the nodal voltages are divided into two sets, namely $V_\mathcal{S} = [v_k] \forall k \in \mathcal{S}$, and $V_\mathcal{R} = [v_k] \forall k \in \mathcal{R}$. The same applies to currents $I_\mathcal{S}, I_\mathcal{R}$, nodal powers $S_\mathcal{S}, S_\mathcal{R}$, and nodal admittance matrix $Y_{\mathcal{SR}}, Y_{\mathcal{RR}}$.

Power distribution networks are usually radial. Therefore, their graphs have a particular structure known as *tree*, in which any two vertices are connected by exactly one path. As a consequence of that, each branch has a unique arriving node, as exemplified in Fig. 2.

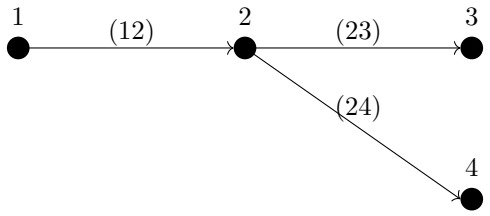

Figure 2. Example of a tree. Note that there is only one arriving node for each branch: $(12) \rightarrow 2$,$(23) \rightarrow 3$, and $(24) \rightarrow 4$

.

A tree structure is vital for the backward-forward sweep algorithm, as presented in Section **??**. This property allows for the creation of an unambiguous map between branch currents and the set of nodal voltages. This map has two consequences, one theoretical and one practical. From a theoretical point of view, the submatrix Y is non-singular if the graph is connected, This property is used by the fixed-point algorithm. From a practical standpoint, this map allows for efficient storage in the same array in the backward-forward sweep algorithm. Along this paper, it is supposed that the branches are sorted from the substation to the final user and the currents in the graph are oriented accordingly ([20] for a graph ordering method).

3. Newton's methods

3.1. Conventional Newton's method in the real domain

Newton's method is a classic algorithm for solving optimization problems and systems of non-linear algebraic equations such as the power flow problem. It approaches the solution by successive linear approximations of the non-linear equations, as depicted in Fig. 3. The method departs from an initial guess x_0, where the non-linear equation is approximated linearly. A new point x_1 is calculated, and the function is again approximated to a linear form. The process continues until an acceptable solution is found [21].

Each iteration of Newton's method has two key steps: first, the derivative of the function is calculated, and then, the new operation point is evaluated. The following expressions show the process:

$$[D_f(x)]\Delta x = \Delta f(x) \tag{8}$$

$$x \leftarrow x + \Delta x \tag{9}$$

In the power flow problem, the function f is usually defined by separating the power flow equations into real and imaginary parts, *i.e.*, (5)-(6). The vector of state variable is $x = (\theta, u)$, and $D_f(x)$ is the Jacobian matrix given by Eq. (10),

$$D_f(x) = \begin{pmatrix} \partial p/\partial\theta & \partial p/\partial u \\ \partial q/\partial\theta & \partial q/\partial u \end{pmatrix} = \begin{pmatrix} J^{p\theta} & J^{pu} \\ J^{q\theta} & J^{qu}. \end{pmatrix} \tag{10}$$

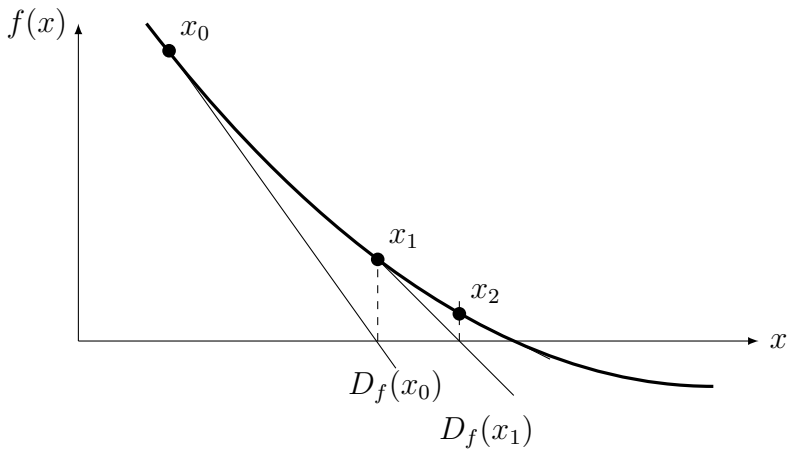

Figure 3. Simple representation of Newton's method

The Jacobian is a block matrix that requires separately considering diagonal and off-diagonal entries. The diagonal entries are given below:

$$J_{kk}^{p\theta} = -b_{kk}v_k^2 - q_k \tag{11}$$
$$J_{kk}^{pu} = g_{kk}v_k + \frac{p_k}{v_k} \tag{12}$$
$$J_{kk}^{q\theta} = -g_{kk}v_k^2 + p_k \tag{13}$$
$$J_{kk}^{qu} = -b_{kk}v_k + \frac{q_k}{v_k}; \tag{14}$$

and the non-diagonal terms are given by the following expressions:

$$J_{km}^{pu} = g_{km}v_k\cos(\theta_{km}) + b_{km}v_k\sin(\theta_{km}) \tag{15}$$
$$J_{km}^{qu} = g_{km}v_k\sin(\theta_{km}) - b_{km}v_k\cos(\theta_{km}) \tag{16}$$
$$J_{km}^{p\theta} = J_{km}^{qu}v_m \tag{17}$$
$$J_{km}^{q\theta} = -J_{km}^{pu}v_m. \tag{18}$$

Algorithm 1 summarizes the main steps of the conventional Newton's method applied to power distribution networks.

Newton's method is computationally expensive since $D_f(x)$ has to be calculated in each iteration. In addition, it is necessary to calculate the step either by inverting $D_f(x)$ or by solving the linear system (8). Both options are time-consuming. In high-voltage power systems, it is possible to simplify the problem by making two main approximations: first, the $p\theta$ problem is separated from that of qu based on the fact that $J^{pu} \to 0$ and $J^{q\theta} \to 0$; second, the sub-matrices $J^{p\theta}$ and J^{qu} are considered to be constant since $r/x \ll 1$, $v \approx 1$, and $\theta \ll 1$. Unfortunately, these approximations are not valid in power distribution networks, and this method is therefore rarely used in this type of networks.

The method can be easily extended to three-phase unbalanced systems, considering initializing the voltages to 1 pu with the proper phase θ (*i.e.*, 0 for phase A, $-2\pi/3$ for phase B, and $2\pi/3$ for

phase C). The mathematical properties of the algorithm are the same in the single-phase and the three-phase cases.

Algorithm 1 Conventional Newton's method

Require: Y, S
  $v \leftarrow \mathbb{I}_n$
  $\theta \leftarrow \theta_{\text{phase}}\mathbb{I}_n$
  $\epsilon \leftarrow \infty$
  while $\epsilon \geq$ tolerance **do**
    Calculate $J^{p\theta}, J^{pv}, J^{q\theta}, J^{qv}$
    Assemble D_f using (10)
    Calculate $\Delta f = (\Delta p, \Delta q)$
    Solve $[D_f(x)]\Delta x = \Delta f(x)$
    $x \leftarrow x + \Delta x$
    $\epsilon \leftarrow \|\Delta x\|$
  end while
  Return x

3.2. Newton's method in the complex domain

The complex representation of the power flow is undoubtedly more compact than the real representation. Therefore, Newton's method is expected to be equally compact in the complex domain. The following definition is required to obtain the linearization:

Definition 3.1. Given a complex function $f = u + jv$ with $u = u(x, y), v = v(x, y)$, the Wirtinger's derivative and its conjugate are defined as follows:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + \frac{j}{2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) \tag{19}$$

$$\frac{\partial f}{\partial z^*} = \frac{1}{2}\left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) + \frac{j}{2}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \tag{20}$$

Definition 3.2. A function $f : \mathbb{C} \to \mathbb{C}$ is holomorphic if its Wirtinger derivatives exist and $\partial f/\partial z^* = 0$.

Holomorphic functions are complex and differentiable. This one infinitely differentiable and locally equal to its own Taylor series. This also means that the following limit can be taken from any direction (see Fig. 4):

$$f'(z) = \lim_{\Delta z \to 0} \frac{f(z + \Delta z) - f(z)}{\Delta z} \tag{21}$$

Another way to identify holomorphic functions is via the Cauchy–Riemann equations presented below:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \tag{22}$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{23}$$

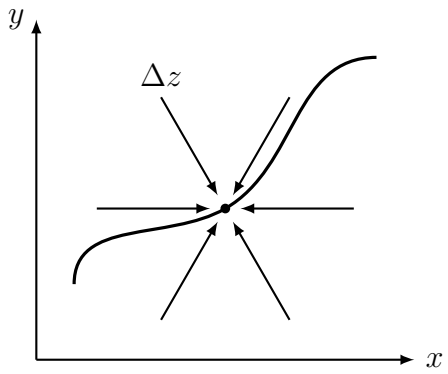

Figure 4. Possible directions in which limit (21) can be taken in the complex domain

Thus, if f is holomorphic, then Wirtinger's derivative is equal to the standard complex derivative. Unfortunately, the power flow equations are non-holomorphic, so a linearization must consider the effect of both the variable and its conjugate.

In general, Wirtinger's derivatives apply the common rules of differentiation known from real-valued analysis concerning the sum, product, and composition of functions, namely:

$$\frac{\partial(f+g)}{\partial z} = \frac{\partial f}{\partial z} + \frac{\partial g}{\partial z} \tag{24}$$

$$\frac{\partial(f+g)}{\partial z^*} = \frac{\partial f}{\partial z^*} + \frac{\partial g}{\partial z^*} \tag{25}$$

$$\frac{\partial(f\cdot g)}{\partial z} = f\frac{\partial g}{\partial z} + g\frac{\partial f}{\partial z} \tag{26}$$

$$\frac{\partial(f\cdot g)}{\partial z^*} = f\frac{\partial g}{\partial z^*} + g\frac{\partial f}{\partial z^*} \tag{27}$$

Operations are similar to those in partial derivatives, so z^* can be regarded as a constant when computing the derivative with respect to z and *vice versa*:

$$\frac{\partial}{\partial z}z^* = \frac{\partial}{\partial z^*}z = 0 \tag{28}$$

A complex linear approximation of f can be obtained using these simple definitions, as shown below:

$$\Delta f(z, z^*) = \left(\frac{\partial f}{\partial z}\right)\Delta z + \left(\frac{\partial f}{\partial z^*}\right)\Delta z^* \tag{29}$$

This approximation behaves like the Jacobian matrix when $f : \mathbb{C}^n \to \mathbb{C}^n$, and it can be applied sequentially by using the iteration $z \leftarrow z + \Delta z$. The complex linearization of Eq. (4) is presented below:

$$\Delta s_k^* = \sum_{m\in\mathcal{S}} y_{km}v_m\Delta v_k + \sum_{m\in\mathcal{R}} y_{km}v_m\Delta v_k^* + y_{km}v_k^*\Delta v_m \tag{30}$$

This equation can be written in matrix form, thus obtaining the following affine equation:

$$J_A\Delta V_{\mathcal{R}} + J_B\Delta V_{\mathcal{R}}^* + \Delta S_{\mathcal{R}}^* = 0, \tag{31}$$

where J_A and J_B are the complex square matrices defined below:

$$J_A = \text{diag}(V_{\mathcal{R}}^*) Y_{\mathcal{R}\mathcal{R}} \tag{32}$$
$$J_B = \text{diag}(Y_{\mathcal{N}\mathcal{R}} V_{\mathcal{S}}) + \text{diag}(Y_{\mathcal{R}\mathcal{R}} V_{\mathcal{R}}) \tag{33}$$

Note that Eq. (31) depends of $\Delta V_{\mathcal{R}}$ and $\Delta V_{\mathcal{R}}^*$. Therefore, it is not as simple as a conventional linear system. First, the complex conjugate is calculated:

$$J_A^* \Delta V_{\mathcal{R}}^* + J_B^* \Delta V_{\mathcal{R}} + \Delta S_{\mathcal{R}} = 0 \tag{34}$$

Then, $\Delta V_{\mathcal{R}}$ is represented as function of $\Delta V_{\mathcal{R}}^*$:

$$\Delta V_{\mathcal{R}} = J_A^{-1}(-J_B V_{\mathcal{R}}^* - \Delta S_{\mathcal{R}}) \tag{35}$$

Finally, (35) is replaced into (34), and $\Delta V_{\mathcal{R}}^*$ is cleared:

$$\Delta V_{\mathcal{R}} = (J_A^* - J_B^* J_A^{-1} J_B)^{-1}(J_B^* J_A^{-1} \Delta S_{\mathcal{R}} - \Delta S_{\mathcal{R}}^*) \tag{36}$$

This equation may be efficiently solved by defining auxiliary matrices M_1 and M_2, as follows:

$$Z_{\mathcal{R}\mathcal{R}} = Y_{\mathcal{R}\mathcal{R}}^{-1} \tag{37}$$
$$M_1 = J_B^* Z_{\mathcal{R}\mathcal{R}} \,\text{diag}(1/V_{\mathcal{R}}^*) \tag{38}$$
$$M_2 = J_A^* - M_1 J_B \tag{39}$$
$$M_2 \Delta V_{\mathcal{R}}^* = M_1 \Delta S_{\mathcal{R}} - \Delta S_{\mathcal{R}}^* \tag{40}$$

In this way, only one inverse is directly calculated (*i.e.*, the inverse of $Y_{\mathcal{R}\mathcal{R}}$), and this inverse is constant. Newton's method in the complex domain is presented in Algorithm 2. The method can be extended to three-phase unbalanced systems by considering the three-phase admittance matrix and a proper initialization of the nodal voltages, taking the corresponding phase into account. Note that the method has a simple and clean code that enables to adapt the script to new problems. This is the main advantage of complex representation.

Algorithm 2 Newton's method in complex domain

Require: $Y_{\mathcal{S}\mathcal{R}}, Y_{\mathcal{R}\mathcal{R}}, S_{\mathcal{R}}$
  Calculate $Z_{\mathcal{R}\mathcal{R}}$ with (37)
  $V_{\mathcal{R}} \leftarrow 1e^{j\theta_{\text{phase}}}\mathbb{I}_n$
  $\epsilon \leftarrow \infty$
  while $\epsilon \geq$ tolerance **do**
    Calculate J_A and J_B with (32) and (33)
    Calculate $\Delta S_{\mathcal{R}}$
    Calculate $\Delta V_{\mathcal{R}}$ with (38) to (40)
    $V_{\mathcal{R}} \leftarrow V_{\mathcal{R}} + \Delta V_{\mathcal{R}}$
    $\epsilon \leftarrow \|\Delta V_{\mathcal{R}}\|$
  end while
  Return $V_{\mathcal{R}}$

4. Fixed-point methods

4.1. Direct formulation

Another widely used method for solving nonlinear equation systems is transforming the problem into a fixed point. The Gauss-Seidel method is an excellent example of this approach. This method is widely used for solving systems of linear equations, and it has also been used for solving power flow equations by employing the following iteration:

$$v_k \leftarrow \frac{1}{y_{kk}} \left(\frac{s_k^*}{v_k^*} - \sum_{m \neq k} y_{km} v_m \right) \tag{41}$$

This method is important for historical reasons, as it was one of the first approaches for solving the power flow problem. However, it suffers from convergence issues, and hence it is rarely used in practice. Nevertheless, it is the origin of other fixed-point algorithms that are more useful in practice. Before presenting these algorithms, the concept of fixed point should be formally defined.

Definition 4.1 (Fixed point)**.** Let $\mathbb{F}$ be any space and T a map of $\mathbb{F}$ into $\mathbb{F}$. A point $v \in \mathbb{F}$ is called a *fixed point* for T if $v = T(v)$.

Fixed point-theory allows solving the equation $f(v) = 0$ by searching for a fixed point of $T(v) = v - f(v)$. In some cases, this fixed point can be calculated by iteratively applying $v \leftarrow T(v)$. The method converges if the map T is a contraction as explained in Section 5.

It is straightforward to find a map T in power distribution networks. First, the nodal power is represented in matrix form, separating the current in the slack from the remaining nodes, as given below:

$$\left(\frac{S_\mathcal{R}}{V_\mathcal{R}} \right)^* = Y_{\mathcal{RS}} V_\mathcal{S} + Y_{\mathcal{RR}} V_\mathcal{R}. \tag{42}$$

Second, the nodal voltage is cleared as follows:

$$V_\mathcal{R} = Y_{\mathcal{RR}}^{-1} \left(\left(\frac{S_\mathcal{R}}{V_\mathcal{R}} \right)^* - Y_{\mathcal{RS}} V_\mathcal{S} \right). \tag{43}$$

Finally, a map T is obtained[3]:

$$T(V_\mathcal{R}) = Y_{\mathcal{RR}}^{-1} \left(\left(\frac{S_\mathcal{R}}{V_\mathcal{R}} \right)^* - Y_{\mathcal{RS}} V_\mathcal{S} \right) \tag{44}$$

Note that the inverse of $Y_{\mathcal{RR}}$ only needs to be calculated once. Another option is to calculate the result of the map without explicitly calculating this inverse. The matrix $Y_{\mathcal{RR}}$ is sparse, so this option may be more efficient. Algorithm 3 summarizes the main steps of this fixed-point method. Note the simplicity of the algorithm.

[3]Note that this fixed point iteration is different from the Gauss-Seidel method presented in Eq. (41). The first term in the Gauss-Seidel method is a scalar, whereas here it is a matrix. This is not a minor difference, since the convergence of the algorithm is radically different.

Algorithm 3 Fixed-point method with Y_{bus} representation

Require: $Y_{\mathcal{SR}}, Y_{\mathcal{RR}}, S_{\mathcal{R}}$
  $V_{\mathcal{R}} \leftarrow 1e^{j\theta_{\text{phase}}}\mathbb{I}_n$
  $U \rightarrow V_{\mathcal{R}}$
  $\epsilon \leftarrow \infty$
  while $\epsilon \geq$ tolerance **do**
    $V_{\mathcal{R}} \leftarrow T(V_{\mathcal{R}})$
    $\epsilon \leftarrow \|U - V_{\mathcal{R}}\|$
    $U \rightarrow V_{\mathcal{R}}$
  end while
  Return $V_{\mathcal{R}}$

The method is the same for single and three-phase networks. As in the previous cases, the admittance matrix is three-phase, and the voltages' angles need to be initialized properly. Note that the method does not impose any constraint related to the radiality of the grid. The method can be easily extended to consider ZIP loads and renewable power (this aspect is beyond the objectives of this review).

4.2. Backward-forward sweep algorithm

Another fixed-point algorithm can be obtained by using some particular characteristics of power distribution networks. The strategy consists of making use of the tree structures of the graph associated with this type of network. This structure allows solving the power flow problem by applying Kirchoff's circuit laws in three steps: first, the nodal current is calculated from the nodal power and voltage; second, Kirchoff's current law is applied from the last user back to the main substation in a backward sweep; third, Kirchoff's voltage law is applied from the substation to the last user in a forward sweep iteration. The method is known as the backward/forward sweep algorithm for obvious reasons. Algorithm 4 represents the pseudo-code for this method. All nodes are expected to be ordered from the substation to the final users.

Algorithm 4 Backward-forward sweep algorithm

Require: $Y_{\mathcal{SR}}, Y_{\mathcal{RR}}, S_{\mathcal{R}}$
  $V_{\mathcal{R}} \leftarrow 1e^{j\theta_{\text{phase}}}\mathbb{I}_n$
  $U \rightarrow V_{\mathcal{R}}$
  $\epsilon \leftarrow \infty$
  while $\epsilon \geq$ tolerance **do**
    $I_{\mathcal{R}} \leftarrow$ nodal current calculation$(S_{\mathcal{R}}, V_{\mathcal{R}})$
    $I_{\mathcal{E}} \leftarrow$ Backward sweep$(I_{\mathcal{R}})$
    $V_{\mathcal{R}} \leftarrow$ Forward sweep$(V_{\mathcal{R}}, I_{\mathcal{R}})$
    $\epsilon \leftarrow \|U - V_{\mathcal{R}}\|$
    $U \rightarrow V_{\mathcal{R}}$
  end while
  Return $V_{\mathcal{R}}$

Each of the steps are explained below with the help of the graph given in Fig. 2. First, the nodal current is calculated for each node except the slack node. The following expression is used:

$$i_k = \left(\frac{s_k}{v_k}\right)^*, \ \forall k \in \mathcal{R} = \{2, 3, 4\} \tag{45}$$

Then, the branch currents $I_{\mathcal{E}}$ are calculated using nodal currents $I_{\mathcal{R}}$. This step is known as *backward sweep*. The calculations for the graph depicted in Fig. 2 are given below:

$$i_2 \leftarrow i_2 + i_4 \tag{46}$$
$$i_2 \leftarrow i_2 + i_3 \tag{47}$$
$$i_1 \leftarrow i_1 + i_2 \tag{48}$$

An important detail of implementation: branch currents are stored in the same array as nodal currents since there is only one receiving node for each branch. Thus, i_4 now stores i_{24}, i_3 stores i_{23} and i_2 stores i_{12}. Finally, the new voltages are calculated in a forward sweep, namely:

$$v_2 \leftarrow v_1 - z_{12} i_2 \tag{49}$$
$$v_3 \leftarrow v_2 - z_{23} i_3 \tag{50}$$
$$v_4 \leftarrow v_2 - z_{24} i_4 \tag{51}$$

The main advantage of this method is that no matrix needs to be inverted –not even the nodal admittance matrix. In addition, the method can be easily extended to the three-phase case, and it constitutes a fixed point since the nodal voltage is calculated from the nodal voltages in each iteration [22]. The only difference is that the map is not explicitly defined [23]. The next section presents a formal analysis of this and the previous methods.

5. Equivalences, convergence, and analysis

This section analyzes the equivalence between the two pairs of algorithms presented in the last section. It also presents results related to the convergence as well as insights about its practical implementation. Four methods have been presented at this point: Newton's method, complex sequential linearization, the fixed-point method with Y_{bus} representation, and the backward-forward sweep algorithm. The first two algorithms are based on linear approximations of the non-linear equations, whereas the last algorithms are based on fixed-point theory. Although each algorithm requires a different implementation, they have some equivalences from the theoretical point of view.

Definition 5.1. Consider two power flow algorithms A and B, where the voltage in iteration k of A is v_k^A and the voltage in iteration k of B is v_k^B. A and B are behaviorally equivalent if $\left\|v_k^A - v_k^B\right\| \leq \epsilon$, where ϵ is a rounding error.

It is easy to obtain the equivalence between the conventional Newton's method and Newton's method in complex domain by using the following result:

Lemma 1. *Sequential complex linearization is behaviorally equivalent to the linearization used by Newton's method.*

Proof. Replace $z = x + jy$ and $f = (u, v)$ in Eq. (29) and note that it is equivalent to the following linearization's real and imaginary parts:

$$\Delta u = \left(\frac{\partial u}{\partial x}\right)\Delta x + \left(\frac{\partial u}{\partial y}\right)\Delta y \tag{52}$$

$$\Delta v = \left(\frac{\partial v}{\partial x}\right)\Delta x + \left(\frac{\partial v}{\partial y}\right)\Delta y. \tag{53}$$

This is clearly the same linearization obtained for Newton's method applied to the vector function $f = (u, v)$. □

Algorithms 1 and 2 are just two representations of the same linearization. Hence, their convergence properties are the same. However, the convergence of two equivalent power flow algorithms does not imply the same time calculation. Two algorithms can have the same convergence plot but different time calculations. Simple aspects such as the programming language and the way the information is stored may drastically change the performance of the algorithm. This aspect is analyzed in the results section.

On the other hand, Newton's method has a well-defined convergence behavior, which is given by the Kantorovich theorem [24]:

Theorem 1. *(Kantorovich theorem in $\mathbb{R}^n$) Let v_0 be a point in $\mathbb{R}^n$ and $F : \mathcal{B}_0 \to \mathbb{R}^n$ a differentiable map with an invertible derivative $[D_f(v)]$. Define*

$$\Delta v_0 = [D_f(v_0)]^{-1} F(v_0) \tag{54}$$

$$v_1 = v_0 + \Delta v_0 \tag{55}$$

$$\mathcal{B}_0 = \{v : \|v - v_1\| \leq \|\Delta v_0\|\} \tag{56}$$

if the derivative $[D_f(v)]$ satisfies the Lipschitz condition,

$$\|D_f(v) - D_f(u)\| \leq K \|v - u\|, \forall v, u \in \mathcal{B}, \tag{57}$$

and if the inequality

$$h = \|F(v_0)\| \left\|D_f(v_0)^{-1}\right\|^2 K \leq \frac{1}{2} \tag{58}$$

is satisfied, then the equation $F(v) = 0$ has a unique solution in $\mathcal{B}_0$, and Newton's methods converges to it with Newton's step 9 and an initial condition v_0. Moreover, if $h < \frac{1}{2}$, the order of convergence is at least quadratic.

This result is well-known in the scientific literature [25]. However, for the sake of completeness, a sketch of proof is presented below. This proof is built upon the work of [26].

Proof. Let us define $A = I - D_f(v_0)^{-1} D_f(v_1)$. Then, by replacing $I = D_f(v_0)^{-1} D_f(v_0)$ and by the Lipschitz condition of $D_f(v)$, following is obtained:

$$\|A\| \leq \left\|D_f(v_0)^{-1}\right\| K \|v_0 - v_1\| \tag{59}$$

but $\|v_0 - v_1\| = \|\Delta v_0\| \leq \|D_f(v_0)^{-1}\| \, \|F(v_0)\|$, and, due to condition (58), $\|A\| \leq 1/2$. Therefore, the Banach Lemma can be used, which guarantees the existence of the inverse of $(I - A)$ and gives boundaries as follows:

$$\left\|D_f(v_1)^{-1}\right\| \leq \left\|D_f(v_0)^{-1}\right\| \left\|(I - A)^{-1}\right\| \leq 2 \left\|D_f(x_0)^{-1}\right\|. \tag{60}$$

On the other hand, let us define a function $g : \mathbb{R} \to \mathbb{R}^n$ as $g(t) = F(v + t\Delta v)$. Then, $g'(t) = [D_f(v + t\Delta v)]\Delta t$, and hence

$$F(v + \Delta v) - F(v) = g(1) - g(0) = \int_0^1 g'(t)dt, \tag{61}$$

that is,

$$F(v + \Delta v) - F(v) = D_f(v)\Delta v + \int_0^1 D_f(v + t\Delta v)\Delta v - D_f(v)\Delta v dt. \tag{62}$$

By the Lipschitz condition of D_f, the following is obtained:

$$\|F(v + \Delta v) - F(v) - D_f(v)\Delta v\| \leq \int_0^1 K \|v + t\Delta v - v\| \, \|\Delta v\| \, dt \tag{63}$$

$$\leq \frac{K}{2} \|\Delta v\|^2 \tag{64}$$

since $\Delta v_k = D_f(v_k)^{-1} F(v_k)$ in each iteration. Then,

$$\|F(v_{k+1})\| \leq \frac{K}{2} \|\Delta v_k\|^2. \tag{65}$$

Finally, let us analyze the step $\Delta v_1 = -D_f(v_1)F(v_1)$, which, by applying (60), (65), and (58), yields the following:

$$\|\Delta v_1\| \leq \|D_f(v_1)\| \, \|F(v_1)\| \tag{66}$$

$$\leq (2 \|D_f(v_0)\|) \left(K/2 \|\Delta v_0\|^2\right) \tag{67}$$

$$\leq 1/2 \|\Delta v_0\| \tag{68}$$

By applying the same argument to the next iterations, it can be concluded that there is a contraction of Δv and $F(v)$. □

The Kantorovich theorem ensures quadratic convergence when the initial approximations are such that conditions (54) to (56) hold. However, these conditions are quite severe and may not be fulfilled by a given scenario. Therefore, it is very important to select a suitable initial condition. In the case of three-phase unbalanced systems, the initial conditions are $v = 1$ and $\theta = \theta_{\text{phase}}$ according to the phase. When all conditions are fulfilled, the algorithm converges to the power flow solution, and this solution is unique.

The uniqueness of the solution is an essential aspect of this type of analysis. The convergence of an algorithm may be evaluated through a sequence of numerical simulations. However, there can always be doubt about whether this is the only possible solution. Theorems like Theorem 1 ensure that there are no other possible solutions in the region $\mathcal{B}_0$. There might be other solutions in other regions, but the solution in $\mathcal{B}_0$ is unique. It is usual that this other solution lacks physical meaning by considering, for example, voltages with negative magnitude and angles higher than 2π. Understanding the physics of the problem is important to use theoretical results correctly.

There are also convergence and uniqueness conditions for fixed-point algorithms, which are analyzed below:

Lemma 2. *The fixed-point method with Y_{bus} representation is behaviorally equivalent to the backward-forward sweep algorithm.*

Proof. The first step of the backward-forward sweep algorithm consists of calculating nodal currents. This step is represented in matrix form as given below:

$$I_{\mathcal{R}} = (S_{\mathcal{R}}/V_{\mathcal{R}})^*. \tag{69}$$

Nodal currents are related to branch currents by the incidence matrix given below:

$$\begin{pmatrix} I_{\mathcal{S}} \\ I_{\mathcal{R}} \end{pmatrix} = \begin{pmatrix} A_0 \\ A_{\mathcal{N}} \end{pmatrix}^\top I_{\mathcal{E}}. \tag{70}$$

The sub-matrix $A_{\mathcal{R}}$ is non-singular, since the grid is radial (*i.e.*, the graph is a connected tree). Therefore, it is possible to obtain branch currents from nodal currents as $I_{\mathcal{E}} = A_{\mathcal{R}}^{-1} I_{\mathcal{R}}$. This step corresponds to the backward current sweep.

On the other hand, branch voltages are related to nodal voltages as follows:

$$V_{\mathcal{E}} = A_{\mathcal{S}} V_{\mathcal{S}} + A_{\mathcal{R}} V_{\mathcal{R}} \tag{71}$$

and to branch currents by the Ohm's law, namely

$$V_{\mathcal{E}} = Z_{\mathcal{E}} I_{\mathcal{E}}. \tag{72}$$

This calculation corresponds to the forward voltage sweep. By collecting all the previous expressions, the following fixed-point is obtained:

$$V_{\mathcal{R}} = A_{\mathcal{R}}^{-1} Z_{\mathcal{E}} A_{\mathcal{R}}^{-1\top} \left(\frac{S_{\mathcal{R}}}{V_{\mathcal{R}}} \right)^* - A_{\mathcal{R}}^{-1} A_{\mathcal{S}} V_{\mathcal{S}}. \tag{73}$$

It is straightforward to demonstrate the following equivalences:

$$Y_{\mathcal{R}\mathcal{R}}^{-1} = A_{\mathcal{R}}^{-1} Z_{\mathcal{E}} A_{\mathcal{R}}^{-1\top} \tag{74}$$

$$Y_{\mathcal{R}\mathcal{R}}^{-1} Y_{\mathcal{R}\mathcal{S}} V_{\mathcal{S}} = A_{\mathcal{R}}^{-1} Z_{\mathcal{E}} A_{\mathcal{R}}^{-1\top} A_{\mathcal{R}}^{-1} A_{\mathcal{S}} V_{\mathcal{S}}. \tag{75}$$

Hence, Eq. (73) is equivalent to Eq. (43). □

Considering that Algorithms 3 and 4 are equivalent, their convergence can be analyzed.

Definition 5.2. Let $\mathcal{B} = \{v : \|v - v_0\| \leq \delta\}$ be a closed ball of $\mathbb{F}^n$, and let $T : \mathcal{B} \to \mathbb{F}^n$. Then, T is said to be a contraction mapping if there is a β such that $\|T(v) - T(u)\| \leq \beta \|v - u\|$, with $0 \leq \beta < 1,\ \forall\ v, u \in \mathcal{B}$.

Theorem 2. *If T is a contraction mapping, then there is a unique $v \in \mathcal{B}$ satisfying $v = T(v)$, which can be obtained by applying iteration $v \leftarrow T(v)$ starting from an initial point in $\mathcal{B}$.*

Proof. The contraction mapping theorem is general for any Banach space, but we are interested in $\mathbb{C}^n$. Let $T : \mathcal{B} \to \mathcal{B}$ be a contraction mapping in a closed ball $\mathcal{B} \in \mathbb{C}^n$. Considering two points $u, v \in \mathcal{B}$, the following is obtained:

$$\|u - v\| = \|u - v + T(u) - T(v) - T(u) + T(v)\| \tag{76}$$
$$\leq \|u - T(u)\| + \|v - T(v)\| + \|T(u) - T(v)\| \tag{77}$$
$$\leq \|u - T(u)\| + \|v - T(v)\| + \alpha \|u - v\| . \tag{78}$$

Rearranging the expression yields the following result:

$$\|u - v\| \leq \frac{1}{1 - \alpha}(\|u - T(u)\| + \|v - T(v)\|). \tag{79}$$

Clearly, if $u = T(u)$ and $v = T(v)$, then $\|u - v\| \leq 0$. Since a norm is always positive except in zero, then necessarily $u = v$, which means that the fixed point is unique.

Now, let us define a sequence $\{v_k\}_0^\infty$ by the iteration $v_{k+1} \leftarrow T(v_k)$. Then, it follows that

$$\|v_{k+n} - v_k\| \leq \left(\alpha^{k-1} \sum_{m=0}^{\infty} \alpha^m\right) \|v_2 - v_1\| = \frac{\alpha^{k-1}}{1 - \alpha} \|v_2 - v_1\| . \tag{80}$$

Therefore, $\{v_k\}_0^\infty$ is a Cauchy sequence. Since $\mathbb{R}^n$ is complete, then $\{v_k\}_0^\infty$ converges to a fixed $v \in \mathbb{R}^n$. More details about this theorem can be found in [27] and [28]. □

Contraction mapping theory is important because it guarantees the convergence of the fixed-point algorithm. In addition, it ensures the uniqueness of the solution, just as in Newton's method. However, the convergence rate given by Eq. (80) is linear, while the convergence rate of Newton's method is quadratic. These convergence properties are numerically evaluated in the next section.

6. Results

The four algorithms mentioned above were implemented in Matlab for the three-phase unbalanced case. The European Low Voltage Test Feeder was used as a reference [29]. This feeder was proposed by the IEEE Working Group of the Distribution System Analysis Subcommittee. It is a real low-voltage feeder that operates at 416 V and has 906 nodes. It has load shapes with a one-minute time resolution over 24 h for time-series simulation. Despite being a test system with typical European voltages, its size and details make it ideal for studying the behavior of the studied algorithms. Fig. 5 shows the single-line diagram. All algorithms were implemented in Matlab 2019. The code is available for download in [30]. Appendix A shows the use of the toolbox.

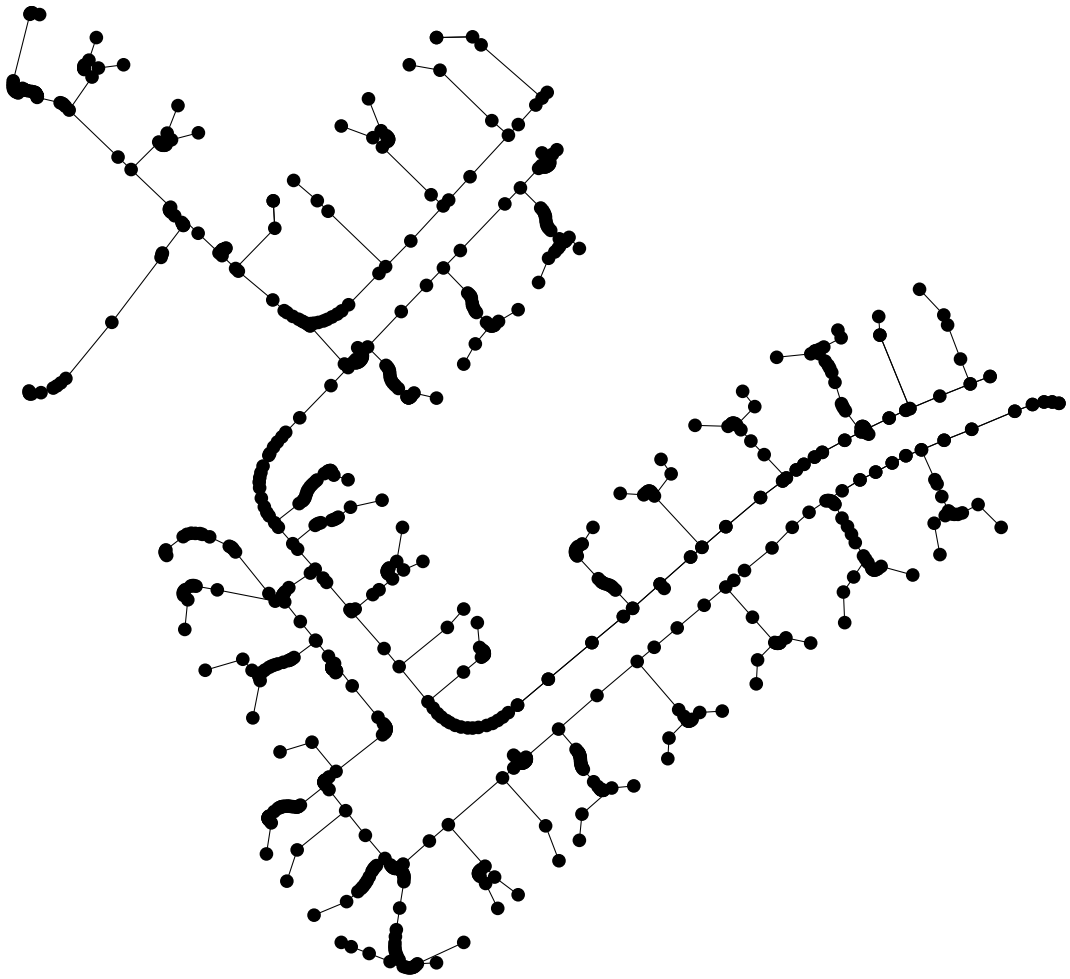

Figure 5. Single-line diagram for the IEEE European low-voltage test feeder

First, a quasi-dynamic simulation was executed in order to evaluate the performance of the algorithms and their precision with respect to the values reported in [31]. Fig. 6 shows the active and reactive power at the main substation for the 1.400 scenarios. The four algorithms returned similar results in terms of nodal voltage despite a significant difference in time calculation.

The voltages in each scenario may be initialized at 1 pu or in the voltage of the previous minute. Fig. 7 shows a comparison between the histograms associated with the iterations of these two approaches. As expected, the initialization in the previous scenario reduces the number of iterations since the initial point is close to the final solution. This, however, is not a significant improvement.

Next, each of the power flow algorithms was evaluated for the scenario $t = 566$min, which corresponds to the maximum load. The convergence plot is depicted in Fig. 8. As expected, the fixed-point method and backward-forward sweep algorithm exhibit exactly the same behavior. The same can be said for Newton's method in the conventional real and complex domains. There are minor differences in the last iteration, which are related to rounding errors, considering the large set of matrices that require inverse calculation in Newton's method.

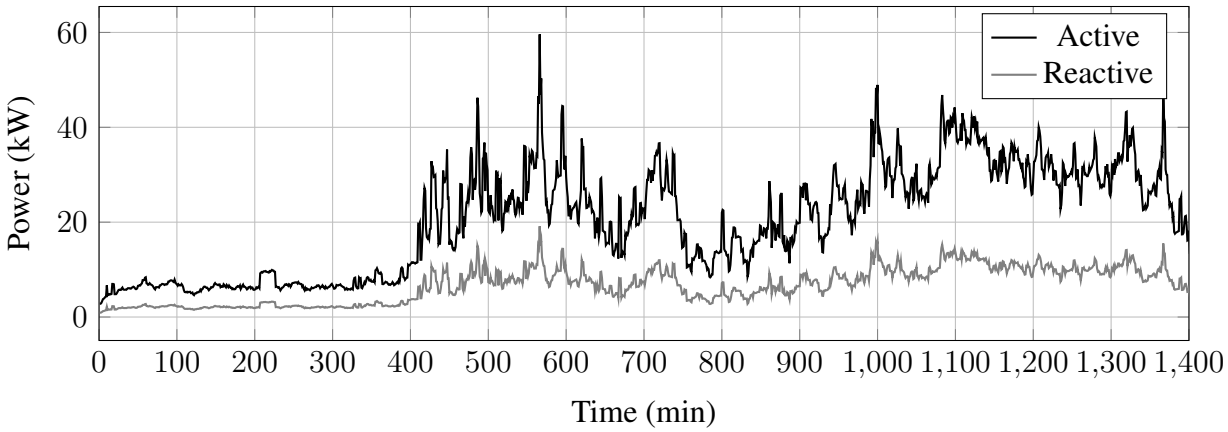

Figure 6. Active and reactive power in the substation for the IEEE 900-node test system using quasi-dynamic simulation

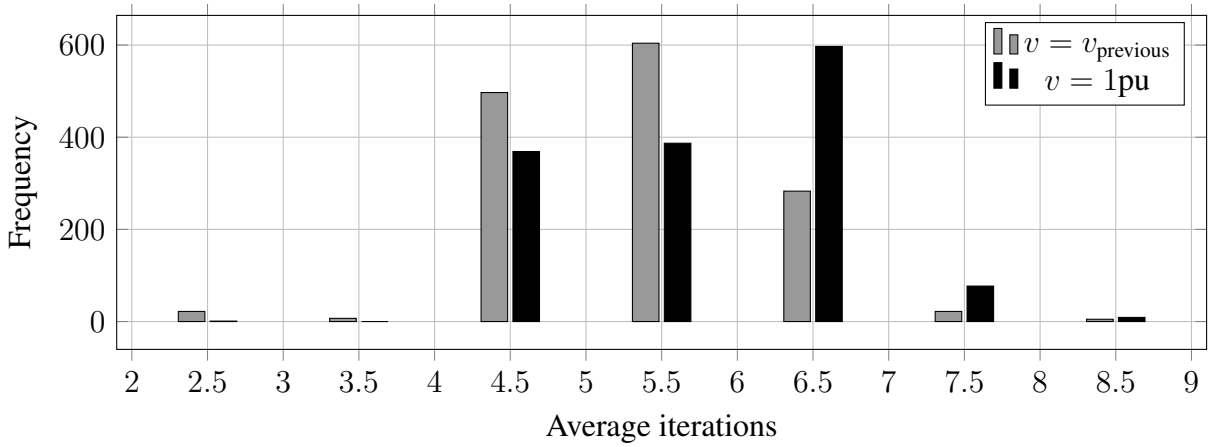

Figure 7. Histogram of the number of iterations with initialization at 1 pu and initialization in the last scenario

While convergence in the fixed-point methods is linear, convergence in Newton's methods is quadratic. Therefore, Newton's methods have a faster convergence in terms of iterations. However, the proposed algorithms have very different calculation time. The backward-forward sweep method has the fastest time, followed by its equivalent, the fixed-point method with the Y_{bus} formulation. Newton's method has the highest time, especially in the complex domain formulation. Table I shows the calculation times for each algorithm. This time was obtained using Matlab's online application.

Table I. Comparisons between different power flow methods for IEEE European Low-Voltage System

Method	**Total time (s)**
Fixed-point method with Y_{bus} formulation	0.062034
Backward-forward sweep algoritm	0.035090
Conventional Newton's method	10.986928
Newton's method in the complex domain	41.662216

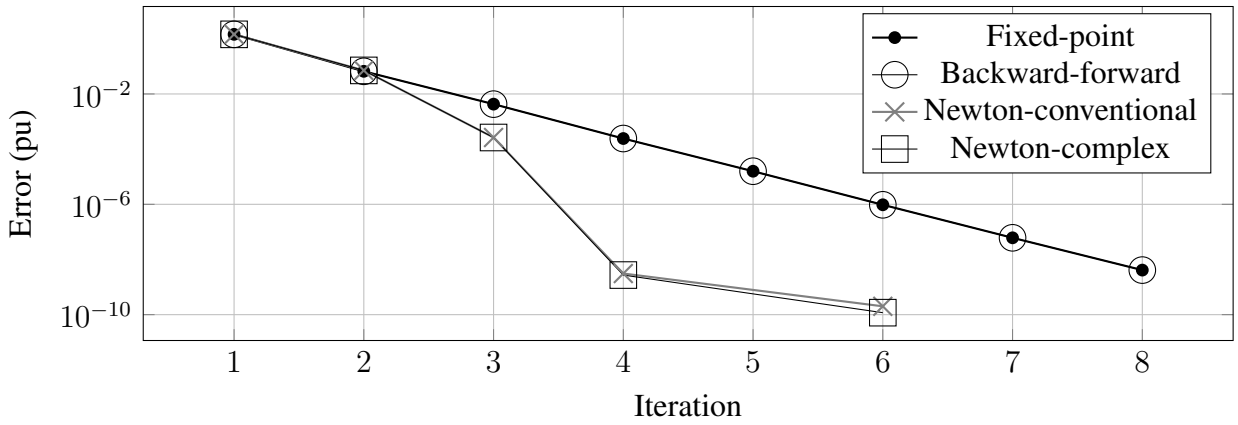

Figure 8. Convergence plot for different power flow algorithms

The fixed-point algorithm with Y_{bus} requires slightly more time. However, it deals directly with meshed grids, and it may be a good option in practice.

7. Conclusions

This paper revisited the power flow problem in power distribution networks while including theoretical and practical aspects. Four power flow methods for three-phase unbalanced power distribution networks were evaluated, namely the conventional Newton's method, Newton's method in the complex domain, the fixed-point method with Y_{bus} representation, and the backward-forward sweep algorithm. It was demonstrated that both Newton's methods were equivalent. In addition, quadratic convergence and uniqueness of the solution may be ensured by these methods. Likewise, it was demonstrated that the fixed-point method and the backward-forward algorithm are equivalent. Linear convergence of these algorithms was demonstrated both theoretically and numerically.

Newton's methods require less iterations than the fixed-point methods. However, each iteration in the fixed-point methods is faster. Therefore, fixed-point methods are preferred in practice for power distribution networks.

A Matlab toolbox was also developed. This toolbox may be used for teaching and research. Its functions allow the calculation of three-phase unbalanced power distribution networks with different load profiles. It is easy to modify the functions in order to include renewable power generation, voltage regulators, electric vehicles, and any other new distributed component.

Acknowledgment

This work was partially funded by the project project Integra2023, code 111085271060, contract 80740-774-2020, by the Ministry of Science, Technology, and Innovation (Minciencias), and the project ODiN (Optimal Operation of Distribution Networks).

References

[1] F. Milano, *Power system modelling and scripting*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-13669-6_4 2

[2] G. W. Stagg and A. H. El-Abiad, *Computer methods in power systems analysis*. Deli, India: McGraw Hill, 1988, vol. 4. 2

[3] B. Stott and O. Alsac, "Fast decoupled load flow," *IEEE Trans. Power Appar. Syst.*, vol. PAS-93, no. 3, pp. 859–869, May 1974. [Online]. Available: https://doi.org/10.1109/TPAS.1974.293985 2

[4] R. K. Portelinha, C. C. Durce, O. L. Tortelli, and E. M. Lourenço, "Fast-decoupled power flow method for integrated analysis of transmission and distribution systems," *Electr. Power Syst. Res.*, vol. 196, p. 107215, 2021. [Online]. Available: https://doi.org/10.1016/j.epsr.2021.107215 2

[5] A. Garcés, "On the Convergence of Newton's Method in Power Flow Studies for DC Microgrids," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 5770–5777, Sept 2018. [Online]. Available: https://doi.org/10.1109/TPWRS.2018.2820430 3

[6] A. Garcés, "A linear three-phase load flow for power distribution systems," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 827–828, 2016. [Online]. Available: https://doi.org/10.1109/TPWRS.2015.2394296 3

[7] U. Ghatak and V. Mukherjee, "An improved load flow technique based on load current injection for modern distribution system," *Int. J. Electr. Power Energy Syst.*, vol. 84, pp. 168–181, 2017. [Online]. Available: https://doi.org/10.1016/j.ijepes.2016.05.008 3

[8] R. Cespedes, "New method for the analysis of distribution networks," *IEEE Trans. Power Deliv.*, vol. 5, no. 1, pp. 391–396, 1990. [Online]. Available: https://doi.org/10.1109/61.107303 3

[9] G. Díaz, J. Gómez-Aleixandre, and J. Coto, "Direct backward/forward sweep algorithm for solving load power flows in ac droop-regulated microgrids," *IEEE Trans. Smart Grid*, vol. 7, no. 5, pp. 2208–2217, 2016. [Online]. Available: https://doi.org/10.1109/TSG.2015.2478278 3

[10] D. M. Fobes, S. Claeys, F. Geth, and C. Coffrin, "Power models distribution.jl: An open-source framework for exploring distribution power flow formulations," *Electr. Power Syst. Res.*, vol. 189, p. 106664, 2020. [Online]. Available: https://doi.org/10.1016/j.epsr.2020.106664 3

[11] A. Garcés, "Uniqueness of the power flow solutions in low voltage direct current grids," *Electr. Power Syst. Res.*, vol. 151, no. Supplement C, pp. 149 – 153, 2017. [Online]. Available: https://doi.org/10.1016/j.epsr.2017.05.031 3

[12] A. Garcés, "Mathematical programming for power systems operation: From theory to applications in python," pp. 171–193, 2022. [Online]. Available: https://doi.org/10.1002/9781119747291.ch10 3

[13] A. Keane, L. F. Ochoa, C. L. T. Borges, G. W. Ault, A. D. Alarcon-Rodriguez, R. A. F. Currie, F. Pilo, C. Dent, and G. P. Harrison, "State-of-the-art techniques and challenges ahead for distributed generation planning and optimization," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1493–1502, 2013. [Online]. Available: https://doi.org/10.1109/TPWRS.2012.2214406 3

[14] P. Arboleya, C. González-Morán, and M. Coto, "Unbalanced power flow in distribution systems with embedded transformers using the complex theory in $\alpha\beta 0$ stationary reference frame," *IEEE Trans. Power Syst.*, vol. 29, no. 3, pp. 1012–1022, 2014. [Online]. Available: https://doi.org/10.1109/TPWRS.2013.2292112 3

[15] B. Donon, R. Clément, B. Donnot, A. Marot, I. Guyon, and M. Schoenauer, "Neural networks for power flow: Graph neural solver," *Electr. Power Syst. Res.*, vol. 189, p. 106547, 2020. [Online]. Available: https://doi.org/10.1016/j.epsr.2020.106547 3

[16] T.-L. Nguyen, Q.-T. Tran, R. Caire, Y. Wang, Y. Besanger, and N.-A. Luu, "Distributed optimal power flow and the multi-agent system for the realization in cyber-physical system," *Electr. Power Syst. Res.*, vol. 192, p. 107007, 2021. [Online]. Available: https://doi.org/10.1016/j.epsr.2020.107007 3

[17] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017. [Online]. Available: https://doi.org/10.1109/TSG.2017.2720471 3

[18] A. S. Nair, S. Abhyankar, S. Peles, and P. Ranganathan, "Computational and numerical analysis of ac optimal power flow formulations on large-scale power grids," *Electr. Power Syst. Res.*, vol. 202, p. 107594, 2022. [Online]. Available: https://doi.org/10.1016/j.epsr.2021.107594 3

[19] S. Pineda and J. Morales, "Is learning for the unit commitment problem a low-hanging fruit?" *Electr. Power Syst. Res.*, vol. 207, p. 107851, 2022. [Online]. Available: https://doi.org/10.1016/j.epsr.2022.107851 3

[20] R. Akiyoshi and N. Yamaguchi, "Algorithms for renumbering nodes in distribution systems for fast computation of power flow," in *2017 IEEE-PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ISGTEurope.2017.8260314 6

[21] L. Thurner, A. Scheidler, F. Schäfer, J.-H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "Pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 6510–6521, 2018. [Online]. Available: https://doi.org/10.1109/TPWRS.2018.2829021 6

[22] M. F. AlHajri and M. E. El-Hawary, "Exploiting the radial distribution structure in developing a fast and flexible radial power flow for unbalanced three-phase networks," *IEEE Trans. Power Deliv.*, vol. 25, no. 1, pp. 378–389, 2010. [Online]. Available: https://doi.org/10.1109/TPWRD.2009.2021039 13

[23] Y. Wang, N. Zhang, H. Li, J. Yang, and C. Kang, "Linear three-phase power flow for unbalanced active distribution networks with PV nodes," *CSEE J. Power. Energy. Syst.*, vol. 3, no. 3, pp. 321–324, 2017. [Online]. Available: https://doi.org/10.17775/CSEEJPES.2017.00240 13

[24] J. M. Ortega, "The newton-kantorovich theorem," *Amer. Math. Monthly*, vol. 75, no. 6, pp. 658–660, 1968. [Online]. Available: http://www.jstor.org/stable/2313800 14

[25] U. Sur and G. Sarkar, "Existence of explicit and unique necessary conditions for power flow insolvability in power distribution systems," *IEEE Syst J*, vol. 13, no. 1, pp. 702–709, 2019. [Online]. Available: https://doi.org/10.1109/JSYST.2018.2870178 14

[26] J. H. Hubbard and B. B. Hubbard, *Vector calculus, linear algebra, and differential forms: A unified approach*. Prentice Hall, 1999. 14

[27] L. H. Loomis and S. Sternberg, *Advanced calculus*. World Scientific, 2014. 17

[28] A. Granas and J. Dugundji, *Fixed point theory*. Springer, New York, NY, 2003. 17

[29] K. P. Schneider, B. A. Mather, B. C. Pal, C.-W. Ten, G. J. Shirek, H. Zhu, J. C. Fuller, J. L. R. Pereira, L. F. Ochoa, L. R. de Araujo, R. C. Dugan, S. Matthias, S. Paudyal, T. E. McDermott, and W. Kersting, "Analytic considerations and design basis for the IEEE distribution test feeders," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 3181–3188, 2018. [Online]. Available: https://doi.org/10.1109/TPWRS.2017.2760011 17

[30] A. Garcés, "Matlab central file exchange," https://la.mathworks.com/matlabcentral/profile/authors/3009175, accessed: 2022-03-28. 17, 23

[31] "IEEE-PES distribution systems analysis subcommittee radial test feeders," http://ewh.ieee.org/soc/pes/dsacom/testfeeders.html, accessed: 2022-03-28. 18

Prof. Dr. Ing. Alejandro Garcés Ruiz

He received his Bachelor's degree in Electrical Engineering (2004) and a Master's degree in Power Systems from Universidad Tecnológica de Pereira (Pereira, Colombia). After that, he received his PhD in Electrical Engineering from Norges Teknisk Naturvitenskapelige Universitet (NTNU) in Trondheim, Norway, where he developed a new control for HVDC integration of offshore wind farms with a reduced-matrix converter. He is currently an associate professor at the Department of Electric Power Engineering of Universidad Tecnológica de Pereira. He participated in the study *Smart Grids Colombia Visión 2030*, which defined the road map for implementing Smart Grids in Colombia. He is a senior member of IEEE and a Senior researcher of the National Research System in Colombia. In addition, he is an associate editor in IEEE Transactions on Industrial Electronics and IET-Renewable Power Generation. He participates in several groups of CIGRE-Colombia related to microgrids and power electronics. He is also a member of the Colombian chapter of the Society for Industrial and Applied Mathematics (CoSIAM). His current research interests include mathematical optimization and control for power systems applications, dynamics in electric grids, renewable energy, energy storage devices, microgrids, and HVDC transmission. In 2020 he was awarded the Georg Forster Research Fellowship for Experienced Researchers from the Alexander Von Humboldt Foundation in Germany to continue his research on mathematical optimization in active distribution systems.

Appendix

A. Matlab functions

A toolbox of Matlab functions was created to evaluate the power flow methods. The code of these functions is open-source, so that it may be helpful for future research. The code can be downloaded at [30]. It uses only the standard Matlab function as well as the functions `xlsread()` for reading Excel files and `sparse()` for creating sparse matrices. This toolbox may also be used to evaluate other three-phase unbalanced systems if the input information is stored correctly. The input file is an Excel workbook with six worksheets. The information required in each worksheet is presented below:

- Worksheet name: general. It contains general information of the test system. At least, the following information is required:
  1. Nominal line-to-line voltage in kV
  2. Nominal power in MW
- Worksheet name: lines. It contains the connection of the feeder with the following columns:
  1. Bus 1
  2. Bus 2
  3. Length (m)
  4. Line code (a number referring to the worksheet line_codes)
- Worksheet name: line_codes
  1. Identification number
  2. R1 (Ohm/km)
  3. X1(Ohm/km)
  4. R0(Ohm/km)
  5. X0(Ohm/km)
- Worksheet name: loads.
  1. Bus
  2. phase (1 for A, 2 for B and 3 for C)
  3. power factor
  4. Profile (a number referring to the worksheet profiles)
- Worksheet name: profiles. Each column of this worksheet contains a set of active power for the corresponding load.
- Worksheet name: coordinates. It contains the coordinates for piloting the system's graph. It contains the following information:

1. Bus
2. x coordinate
3. y coordinate

The toolbox contains the following basic functions:

- load_feeder.m: It reads the excel file, organizes the information, and calculates per-unit equivalents

  Input : a string with the name and address of the workbook

  Output : a struct with the following fields:

  1. graph: coordinates of the nodes
  2. z_line $3 \times 3 \times n$ tensor with the equivalent impedance of each line in per unit
  3. ybus: $n \times n$ three-phase nodal admittance matrix
  4. yns: sub-matrix $Y_{\mathcal{RS}}$
  5. ynn: sub-matrix $Y_{\mathcal{RR}}$
  6. znn: inverse of $Y_{\mathcal{RR}}$
  7. loads: table of loads
  8. profiles: table of profiles
  9. xy: node coordinates
  10. vs_initial: initial voltages for the slack node
  11. vn_initial: initial voltages for the remaining voltages
  12. p_base: nominal phase power
  13. v_base: nominal line-to-neutral voltage
  14. n_slack: slack nodes for each phase
  15. n_other: list of remaining nodes
  16. num_n: number of nodes
  17. num_l: number of lines
  18. num_d: number of loads
  19. num_e: number of profile scenarios

- load_flow_newton.m : It solves the power flow using conventional Newton's method

  Input : struct generated by load_feeder.m and profile scenario

  Output : struct with the following information:

  1. v_node: vector of nodal voltages
  2. s_node: vector of nodal powers
  3. error: error in each iteration
  4. iter: total iterations
  5. scenario: scenario number

- load_flow_sweep.m: load flow using the backward-forward sweep method. Same input and output information as load_flow_newton.m.
- load_flow_scl.m: load flow using Newton's method in the complex domain. Same input and output information as load_flow_newton.m.
- load_flow_ybus.m: load flow using the fixed-point method with Y_{bus} representation. Same input and output information as load_flow_newton.m.
- show_results.m: it shows a table with nodal voltages and powers, and it plot the feeder's graph.