A Three-Phase Unbalanced Load Flow Solver for Large-Scale Distribution Power Systems

Jonathan Abraham Avilés Cedeño

Master of Science Thesis

TU Delft

A Three-Phase Unbalanced Load Flow Solver for Large-Scale Distribution Power Systems

By

Jonathan Abraham Avilés Cedeño

in partial fulfilment of the requirements for the degree of

Master of Science
in Electrical Engineering
Track Electrical Sustainable Energy

at Delft University of Technology,
to be defended publicly on Friday, August 25th, 2017.

Supervisor:	Dr. Domenico Lahaye,	Delft University of Technology
Thesis committee:	Prof.dr. Peter Palensky,	Delft University of Technology
	Dr.ir. Milos Cvetkovic,	Delft University of Technology

TU Delft

Abstract

The calculation of power flow through electrical networks has been utilized commonly in transmission networks as the first calculation tool to assess their steady state conditions. However, the current introduction of renewable micro-sources (many of them single-phase connected) and the implementation of microgrids makes the calculation of power flow in distribution networks a more valuable and desirable tool.

This project presents the development of a software tool to calculate power flow, losses, voltages and currents of large-scale unbalanced distribution networks. The project has consisted of four main sections: modelling of the distribution network, prototype implementation using MATLAB, C-code implementation targeting large-scale networks and testing of the performance and reliability of the program.

The modelling of the elements of the distribution network has been performed under the phase domain (ABC frame) and its validity has been verified using a MATLAB prototype and the IEEE distribution test feeders. After verifying the models, a program has been written using C to improve performance. The PETSc library has been used to solve the nonlinear and linear problems required.

Finally, a test network of 452745 three-phase nodes has been created and used as input for the program, with the objective of testing the software.

Keywords: *Distribution Network, Phase Domain Modelling, Unbalanced Power Flow Calculation*

Acknowledgements

I would like to sincerely thank my supervisor, Dr. Domenico Lahaye, for his guidance during my thesis research. He has always been available to kindly answer my questions and provide comments about my work. He has also helped me to give shape to my research and to stay always motivated.

I want to thank the members of the thesis committee for their assessment of my work.

I would also like to thank my classmates and friends for all their help.

I want to acknowledge the support from the Ecuadorian government, that through SENESCYT and IFTH, fully sponsored my studies and stay in The Netherlands.

Finally, I want to express my greatest gratitude to my parents, Melva and Abraham, my brother Daniel and my girlfriend Gabriela for their support and patience during these years living abroad.

Jonathan Abraham Avilés Cedeño

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1. Objectives

The current development has the following objectives:

- Model the elements of a typical distribution network under the phase-domain, i.e. in terms of phase quantities, in order to solve the power flow problem on an unbalanced network as an extension of the per-phase solution of balanced networks.

- Develop a software tool that allows to simulate a large-scale electrical distribution network under unbalanced three-phase power conditions and calculate its relevant electrical quantities: power flow, power losses, voltages and currents.

- Select the most efficient mathematical methods to perform such calculations by testing the software against a large distribution network case.

1.2. Motivation

The electrical infrastructure in the world is currently undergoing major changes. Conventional energy sources are being replaced by renewable sources such as wind turbine generators and solar panels. These sources are intermittent and the location where the power is generated will change drastically.

At distribution level, these sources create new requirements because consumers will install photovoltaic systems or small-scale combined heat and power systems and therefore, they will become in part producers of power. Furthermore, electricity consumption will increase due to the electrification of transport (e-vehicle charging), heating, ventilating and air conditioning. These renewable sources at the lowest power levels will increase the dynamic behavior of the network and the direction of the flow of power during time, making necessary power flow calculators for planning on the distribution system.

Distribution networks not always carry the same power through the three phases, they are inherently unbalanced. Asymmetric loads and not transposed distribution lines are among the reasons for this unbalanced three-phase power condition. Moreover, renewables micro-sources can be connected to a single-phase and increase it.

Traditional software tools, used for transmission network studies are not adequate for distribution systems because they assume a perfectly balance system so that a single-phase equivalent is used. Moreover, the number of available (open-source) simulation tools designed specifically for distribution grids is limited, because there was not as much need and the modelling is more complicated.

Therefore, a software tool that consider all three phases of the distribution network (in the “phase-frame”, not through symmetrical components) and can take into a simulation a large number of nodes, including those at the lowest voltage levels will be very helpful.

The current project has consisted of developing such software. Since this tool targets large cases, a low-level programming language as C has been used. Additionally, for the solution

of the mathematic problems that appear in the algorithms, the PETSc [2] numerical library has been utilized. PETSc stands for Portable, Extensible Toolkit for Scientific Computation, and will be discussed in more details in chapter 3.

1.3. Software release

The power flow solver has been implemented using both MATLAB (matrix laboratory) and C/PETSc (Portable, Extensible Toolkit for Scientific Computation) [2]. The final implementation was intended to be written using C, because as a low-level programming language, it performs faster than other languages.

Another reason to write the program in C is the availability of the very powerful library PETSc to facilitate the applied mathematics algorithms required. It is important to mention that PETSc performs much better when using GNU/Linux as operative system, therefore the PETSc version of the program has been written for GNU/Linux machines.

Since C-programs are more complex and harder to write and test, all the models were firstly tested using a MATLAB prototype. In the end, both implementations have demonstrated to perform well, as it will be shown in chapter 6 and their source codes have been made public and are available in the following repositories:

- **MATLAB implementation:** *https://github.com/JonathanAviles/DPFLOW---MATLAB.*

This repository contains:

- Function and script MATLAB files required by the solver.
- Input files corresponding to test cases used in chapter 6: IEEE distribution feeders and a test case with 452745 nodes that includes transmission and distribution nodes, so it is possible to reproduce the results obtained by the present work.
- MATLAB file called 'distributionCaseFormat.m' where the format of the input files is explained in detail.

- **PETSc implementation:** *https://github.com/JonathanAviles/DPFLOW---PETSc.*

This repository contains:

- Source code for the GNU/Linux program and make file that allows to generate the executable file.
- The same input files as for the MATLAB implementation.
- It also includes the explanatory file 'distributionCaseFormat.m'.

It is worth mentioning that all the MATLAB tests have been performed in a Windows 10 machine with a processor Intel Core i7-4710MQ 2.5 GHz (8 logical processors) and 16 GB of RAM memory.

For the C / PETSc performance tests, a virtual machine has been used. It has had assigned 4 logical processors and 10 GB of RAM. The operative system used in the virtual machine has been Linux Mint 18.1 Serena, MATE flavor.

1.4. Thesis outline

Chapter 1: Introduction
This chapter establishes the objectives of the project, explains its motivation and defines the structure of the thesis report.

Chapter 2: Electrical Distribution Network Modelling
This chapter explains the modelling of the different electrical elements that are part of an electrical distribution network.

Chapter 3: Power Flow Calculations Theory
This chapter contains the basic theory of power flow calculations. It presents the linear and non-linear equations derived from the power flow problem and discusses the methods used to solve them.

Chapter 4: PETSc and its use on power flow computations
This chapter presents information regarding the Portable, Extensible Toolkit for Scientific Computation (PETSc) [2], and its different libraries and functions that are of use for the power flow problem.

Chapter 5: Program implementation
This chapter discusses the implementation of the algorithm. It shows the prototyping phase of the project in the numerical environment provided by MATLAB (matrix laboratory) and finally the process followed to develop the program written in C.

Chapter 6: Program Testing
This chapter deals with the application of the program on different case scenarios in order to analyze its performance.

Chapter 7: Conclusions and Recommendations
This chapter summarizes the results obtained from the project and the recommendations for future research.

2 Electrical Distribution Network Modelling

2.1. Distribution Networks

Electric power distribution is the portion of the power delivery infrastructure that takes the electricity from the highly meshed, high-voltage transmission circuits and delivers it to customers [7].

The distribution network starts at the distribution substations, where a power transformer takes the incoming transmission-level voltage and steps it down to several primary circuits, which fan out from the substation. Primary distribution lines are “medium-voltage” circuits, that transmit the power to local distribution transformers, close to each end user [6] [7].

The local distribution transformers step the electrical potential down to the secondary distribution circuits, at consumer voltage level. This secondary distribution circuits, managed by the local utility or distribution company, carry the electricity to the end users, either residential or commercial establishments [6] [7].

The distribution infrastructure and facilities are very extensive, since electricity is required by customers concentrated in cities, suburbs and very remote regions. Distribution circuits are found along most secondary roads and streets. Urban construction is mainly underground; rural construction is mainly overhead. Currently, few places in the industrialized world do not have electricity from a distribution system readily available.

Commonly, they have a radial topology, with the power flowing from one source to a set of loads, but the introduction of renewable sources will change the direction of the flow of power and consequently the network topology may require to be more meshed to increase reliability.

It is important to remark that even though the cases have not been solved using a per-phase equivalent, the representation of the networks has often been done using single-line diagrams. A single-line diagram is a concise portrayal of the topology and assemblage of the components of a network, by using single lines to show the interconnections and standard symbols instead of equivalent circuits to depict its components.

2.2. Unbalanced conditions on distribution networks

As it was mentioned before, distribution networks are inherently unbalanced. It means that the magnitudes of the currents carried by each phase are not the same. The sum of these phase currents will not be equal to zero and will flow through the neutral conductor.

There are several reasons for this unbalance to occur, which are mentioned below:

1. At low voltage level, either on residential or industrial networks, many loads are single phase or two-phase with different power requirements. Therefore, depending on the kind

of loads existing at a building or facility and the amount of electrical power required, the feeder could consist of three-phase, two-phase or even single-phase cables.

A distribution feeder is depicted at figure 1. The feeder supplies energy to single phase, two-phase and three-phase loads. Because of this diversity of loads, distribution systems are mainly unbalanced, even though usually non-three-phase loads are equally distributed among the three phases.

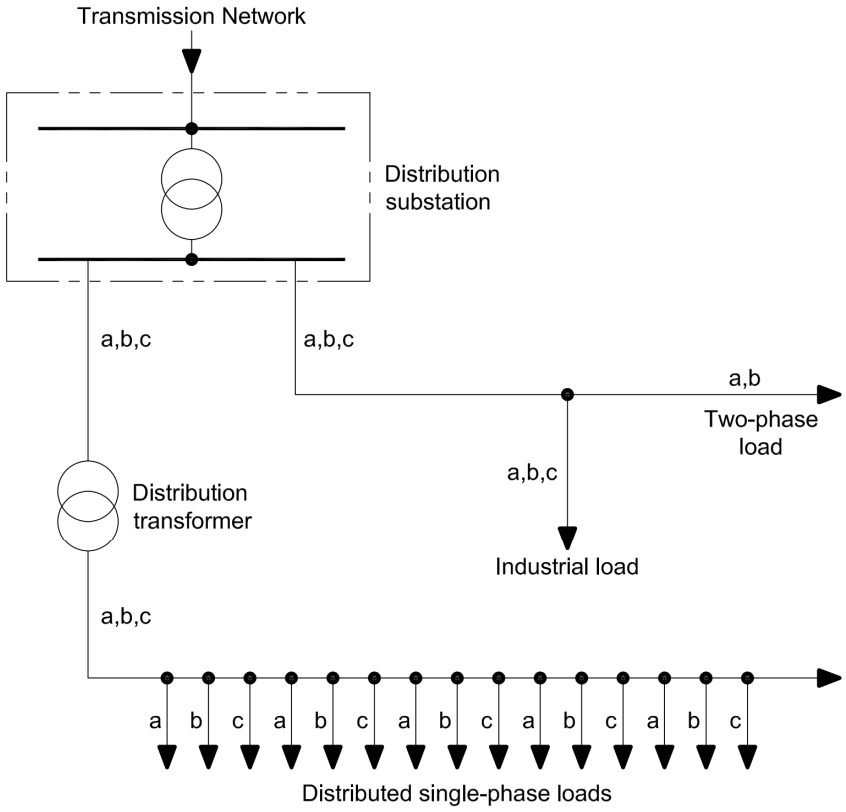


Figure 1. *Typical electrical distribution feeder*

2. The impact of renewable energy micro-sources on power balance and grid frequency may become more significant over the years. If they are small enough to be connected to a single phase and their generation is not equally distributed among the three phases, the electrical network is becoming more unbalanced [14].

3. An additional unbalance is introduced by the non-equilateral conductor spacings of the three-phase overhead and underground line segments. To solve these issues overhead transmission lines are transposed over their support structures, but this is not usually the case for distribution feeders.

2.3. Modeling approach

For this project, the models have been developed in the “phase frame” rather than applying the method of symmetrical components, so the data correspondent to each phase is immediately available and the unbalanced flow problem can be solved as a direct extension to the single-phase equivalent solution used for three-phase balanced systems.

Distribution networks mainly consist of a source coming from the transmission network, a set of cables and lines interconnecting the different buses or nodes of the system and different loads distributed along a geographical area. Each of this units constitutes an electric network in its own right and their interconnection constitutes the distribution system [5] [6].

Among the many alternative ways of describing electrical systems to comply with Kirchhoff's laws, two methods are usually used: mesh and nodal analysis.

2.3.1. Nodal Analysis

The nodal analysis is based on the application of the Kirchhoff's equations to the current injected to each node: the sum of all the currents flowing into a node shall be equal to zero.

For any electrical system, the node voltages and the currents injected to the system at each node can be related using the following equation:

$$I = Y_{bus} * V \tag{2. 1}$$

Where I is the vector of currents injected to the system through each node, Y_{bus} is the nodal admittance matrix and V is the vector of node voltages of the system.

If such system has n nodes, the elements of the equation shall be structured as follows:

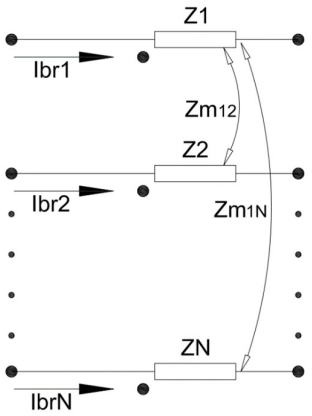
$$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \cdots & Y_{nn} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

Where for any row k:

- Y_{kk} ($1 \leq k \leq n$) is the sum of all the admittances connected to node k
- Y_{km} ($1 \leq k, m \leq n$) is the negative of the total admittances connected between nodes k and m.

2.3.2. Mesh Analysis

The mesh analysis is based on the application of electrical equations to each branch of the network, as follows: the voltage difference through the branch will be equal to the product of its current and impedance plus the voltage induced by the currents through the other branches that are magnetically coupled to the first one. Figure 2 illustrates the voltage equation for one branch in a network with N magnetically coupled branches.



$$V_{br1} = I_{br1}Z_1 + I_{br2}Z_{m12} + \ldots + I_{brN}Z_{m1N}$$
$$V_{br1} = [Z_1 + Z_{m12} + \ldots + Z_{m1N}][I_{br1} + I_{br2} + \ldots + I_{brN}]^T$$

Figure 2. *Three magnetically coupled branches*

By adding the equations for all the branches of the network the following matrix equation can be determined:

$$V_{br} = Z_{pr} * I_{br} \tag{2. 2}$$

Where I_{br} is the vector of branch currents, Z_{pr} is the so called primitive impedance matrix and V_{br} is the vector of branch voltages of the network.

The inverse of the primitive impedance matrix is the primitive admittance matrix Y_{pr}:

$$Y_{pr} = (Z_{pr})^{-1} \rightarrow I_{br} = Y_{pr} * V_{br} \tag{2. 3}$$

It is important to recall that there is a relation between the system matrices of the mesh and nodal analysis:

$$Y_{bus} = C^T * Y_{pr} * C \tag{2. 4}$$

Where C is the connection matrix of the system. This connection matrix is what in graph theory is known as a branch-to-node incidence matrix.

2.3.3. Approach selection

Nodal analysis has been found to be particularly suitable for digital computer work, and is almost exclusively used for routine network calculations. The nodal approach has the following advantages for power networks [1]:

- The numbering of nodes, performed directly from a system diagram, is very simple.
- Data preparation is easy.
- The number of variables and equations is usually less than with the mesh method
- Network crossover branches present no difficulty.
- Parallel branches do not increase the number of variables or equations.
- Node voltages are available directly from the solution, and branch currents are easily calculated
- Off-nominal transformer taps can easily be represented.

Based on the advantages above exposed, nodal analysis have been used for the solution of the system equations and, therefore a nodal admittance matrix has been utilized to model the network. Nevertheless, the primitive admittance matrix has been required to calculate the branch currents of the network and its losses after the voltage nodes were obtained, as shown in section 3.6.

2.4. Primitive admittance matrix for three-phase branches

As a first step to model the distribution network, the matrix that relates the vector of phase voltages and the vector of phase currents for one branch have been defined.

Let us say, there is a three-phase branch with neutral conductor. Each phase will have a self-impedance that relates the voltage across the branch and the current flowing through the branch. However, due to the inductive nature of cables, there will be mutual impedance couplings between phases, which will cause induced voltages across one phase due to the current flowing by each different phase. The following figure depicts this case:

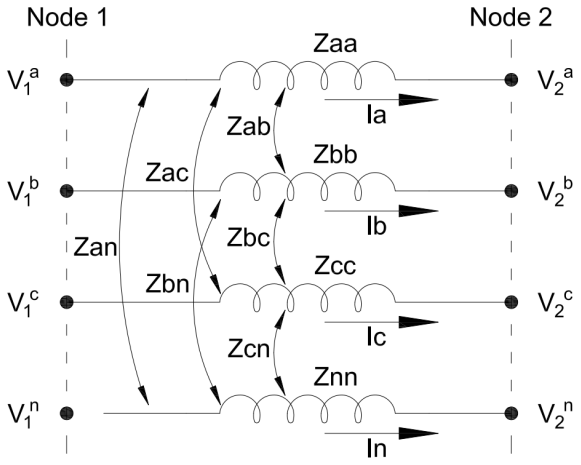


Figure 3. *Three-phase branch with neutral conductor*

For this configuration, the relation between the node voltages and the currents is given by the following equation:

$$\begin{bmatrix} V_1^a \\ V_1^b \\ V_1^c \\ V_1^n \end{bmatrix} - \begin{bmatrix} V_2^a \\ V_2^b \\ V_2^c \\ V_2^n \end{bmatrix} = \begin{bmatrix} Z_{aa} & Z_{ab} & Z_{ca} & Z_{an} \\ Z_{ab} & Z_{bb} & Z_{bc} & Z_{bn} \\ Z_{ca} & Z_{bc} & Z_{cc} & Z_{cn} \\ Z_{an} & Z_{bn} & Z_{cn} & Z_{nn} \end{bmatrix} \begin{bmatrix} I_{12}^a \\ I_{12}^b \\ I_{12}^c \\ I_{12}^n \end{bmatrix}$$

$$V_1^{abcn} - V_2^{abcn} = Z_{12}^{abcn} I_{12}^{abcn} \qquad (2.\ 5)$$

This equation is the Ohm's law in complex matrix version, where:

- V_k^{abcn} is the vector of voltages for the three phases and the neutral conductor of the node k.
- I_{km}^{abcn} is the vector of currents flowing through the three phases and the neutral conductor of the branch between nodes k and m.

- Z_{km}^{abcn} is the primitive impedance matrix corresponding to the three-phase branch connected between nodes k and m.

The inverse of the primitive impedance matrix is the primitive admittance matrix:

$$\left(Z_{12}^{abcn}\right)^{-1} = Y_{12}^{abcn} \Rightarrow I_{12}^{abcn} = Y_{12}^{abcn}\left(V_1^{abcn} - V_2^{abcn}\right) \quad (2.\ 6)$$

Since the basic laws (Kirchhoff's and Ohm's laws) hold for their multiphase equivalents, a nodal admittance matrix can be constructed using the same structure as for the per-phase representation.

An important remark needs to be given. Not always the neutral conductor is carried out with the live phases and in this case the nodal admittance and impedance matrices will have a dimension 3x3.

There is not current injection into the neutral point of electrical devices, therefore even if there is a neutral conductor, this node can always be eliminated, for instance using Kron's reduction. The disadvantage of eliminating this node is that the voltage in the neutral point and the current through the neutral conductor would not be calculated directly from the nodal admittance matrix.

In this development, the neutral nodes have not been considered on the nodal admittance matrix.

2.5. Nodal admittance matrix for three-phase networks

If it is necessary to consider all the exact values per phase, the number of nodes will be tripled for some nodes (three phases) or quadrupled for other nodes (three phases plus neutral conductor) and the dimension of the admittance matrix will simply be longer than for a per-phase representation.

However, in order to handle the data in a better way, the elements corresponding to each phase per node will be grouped as follows:

$$\begin{bmatrix} I_1^{abc} \\ I_2^{abc} \\ \vdots \\ I_n^{abc} \end{bmatrix} = \begin{bmatrix} Y_{11}^{abc} & Y_{12}^{abc} & \cdots & Y_{1n}^{abc} \\ Y_{21}^{abc} & Y_{22}^{abc} & \cdots & Y_{2n}^{abc} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1}^{abc} & Y_{n2}^{abc} & \cdots & Y_{nn}^{abc} \end{bmatrix} \begin{bmatrix} V_1^{abc} \\ V_2^{abc} \\ \vdots \\ V_n^{abc} \end{bmatrix} \quad (2.\ 7)$$

Where:

- n is the number of nodes in the distribution network
- a, b, c are the phases of the system
- I_k^{abc} is the vector of current injections for the three phases of the kth node, 1 ≤ k ≤ n
- V_k^{abc} is the vector of voltages for the three phases of the kth node, 1 ≤ k ≤ n
- Y_{kk}^{abc} is the sum of all the primitive admittance matrices corresponding to the three-phase branches connected to the kth node of the system, 1 ≤ k ≤ n
- Y_{km}^{abc} is the negative of the primitive admittance matrix corresponding to the three-phase branch connected between nodes k and m, 1 ≤ k,m ≤ n

For instance, let us consider the node 1 of the system depicted in figure 4. The first row of the nodal admittance matrix, corresponding to this node will be equal to:

$$\begin{bmatrix} I_1^{abc} \\ \vdots \end{bmatrix} = \begin{bmatrix} Y_{pr1}^{abc} + Y_{pr2}^{abc} + Y_{pr3}^{abc} + \cdots + Y_{prn}^{abc} & -Y_{pr2}^{abc} & -Y_{pr3}^{abc} & \cdots & -Y_{prn}^{abc} \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{bmatrix} \begin{bmatrix} V_1^{abc} \\ V_2^{abc} \\ V_3^{abc} \\ \vdots \\ V_n^{abc} \end{bmatrix}$$

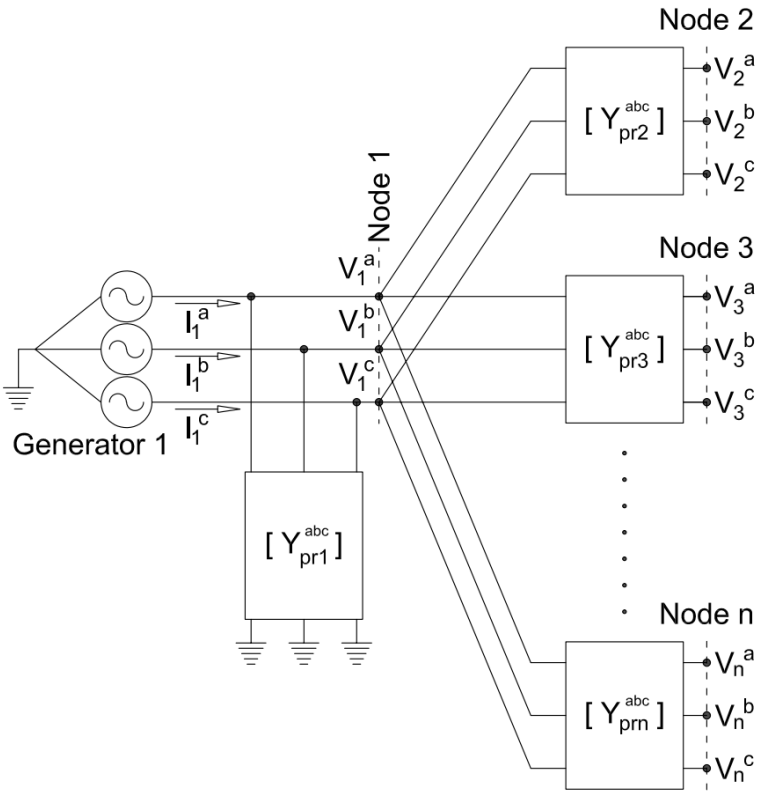


Figure 4. *Three-phase model for one node and its connections*

2.6. Distribution overhead lines and underground cables modeling

For representation of distribution lines and cables, the pi-model will be used. This model is composed by the series impedance and the shunt admittance of the line or cable [5].

Since the distribution network may include up to three phases, both the series impedance and the shunt admittance will be treated as matrices of 3x3 elements. The model for a line connected between nodes n and m is shown in figure 5.

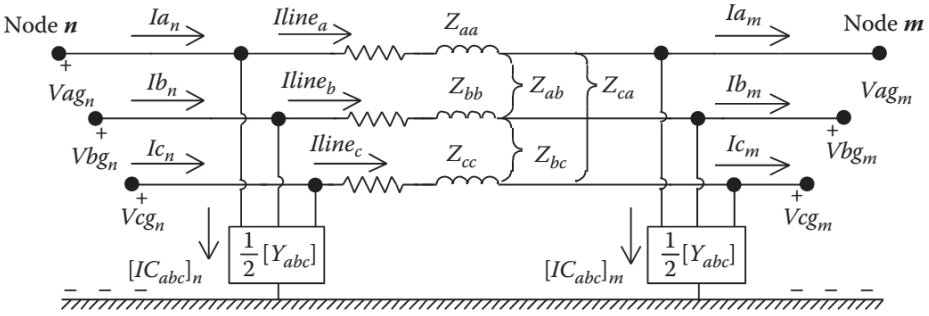


Figure 5. *Three-phase line segment model [5]*

In the case that one segment line is two-phase or single phase, the correspondent series impedance matrix or shunt admittance matrix will contain zeros for the missing phases and couplings, so it always consists of a 3x3 dimension matrix.

The values of the elements of these matrices depend on the material and the size of the conductors and their geometric location on space and one phase respect to the others.

2.7. Distribution transformers modeling

The inherent assumption that the transformer is a balanced three-phase device is justified in most practical situations, however when unbalanced conditions are present, all three-phase transformer connections need to be accurately modelled in phase coordinates.

The primitive admittance matrix, used as a basis for the phase coordinate transformer model is derived from the primitive or unconnected network for the transformer windings and the method of linear transformation enables the admittance matrix of the actual connected network to be found [1].

2.7.1. Transformer primitive admittance matrix

Many three-phase transformers are wound on a common core and all windings are therefore coupled to all other windings. Therefore, in general, a basic two-winding three-phase transformer has a primitive or unconnected network consisting of six coupled coils. If a tertiary winding is also present, the primitive network consists of nine coupled coils [1].

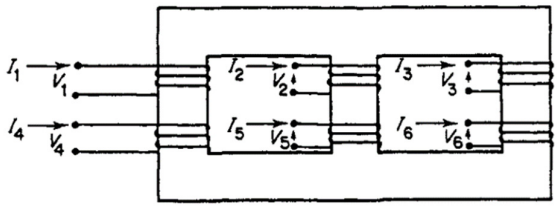


Figure 6. *Diagrammatic representation of two-winding transformer [1]*

The primitive network, can be represented by the primitive admittance matrix that has the following general form:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15} & y_{16} \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25} & y_{26} \\ y_{31} & y_{32} & y_{33} & y_{34} & y_{35} & y_{36} \\ y_{41} & y_{42} & y_{43} & y_{44} & y_{45} & y_{46} \\ y_{51} & y_{52} & y_{53} & y_{54} & y_{55} & y_{56} \\ y_{61} & y_{62} & y_{63} & y_{64} & y_{65} & y_{66} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \end{bmatrix}$$

If interphase coupling can be ignored, or the transformer bank consists of three single phase transformers and the three set of coils are considered of equal characteristics, the structure of the primitive admittance matrix can be simplified to:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{bmatrix} = \begin{bmatrix} +y_p & 0 & 0 & -y_m & 0 & 0 \\ 0 & +y_p & 0 & 0 & -y_m & 0 \\ 0 & 0 & +y_p & 0 & 0 & -y_m \\ -y_m & 0 & 0 & +y_s & 0 & 0 \\ 0 & -y_m & 0 & 0 & +y_s & 0 \\ 0 & 0 & -y_m & 0 & 0 & +y_s \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \end{bmatrix} \quad (2.\ 8)$$

$$Y_{pr} = \begin{bmatrix} +y_p & 0 & 0 & -y_m & 0 & 0 \\ 0 & +y_p & 0 & 0 & -y_m & 0 \\ 0 & 0 & +y_p & 0 & 0 & -y_m \\ -y_m & 0 & 0 & +y_s & 0 & 0 \\ 0 & -y_m & 0 & 0 & +y_s & 0 \\ 0 & 0 & -y_m & 0 & 0 & +y_s \end{bmatrix} \quad (2.\ 9)$$

Where, y_p is the admittance of the primary windings, y_s is the admittance of the secondary windings and y_m is the admittance corresponding to the coupling between primary and secondary windings per phase.

2.7.2. Models for common transformer connections

The network admittance matrix for any two-winding three-phase transformer can now be composed by the method of linear transformation. We need a connection matrix C, that relates the voltage differences across the windings with the node voltages, according to equation 2.4. With this matrix, the nodal admittance matrix is given by $Ybus = C^T * Ypr * C$.

As an example of the procedure for the nodal admittance matrix obtainment, the transformer in figure 7 will be used:

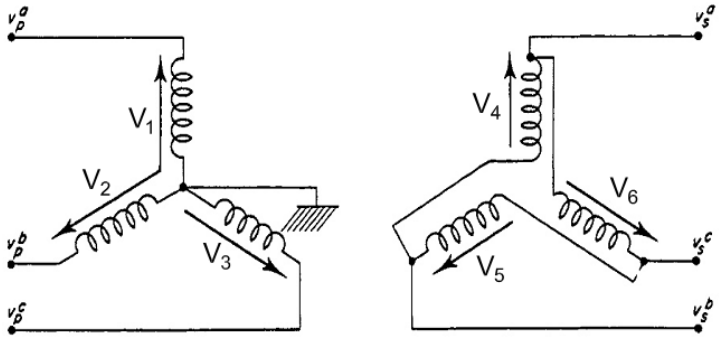


Figure 7. *Diagrammatic representation of two-winding transformer*

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_p^a \\ V_p^b \\ V_p^c \\ V_s^a \\ V_s^b \\ V_s^c \end{bmatrix} \Rightarrow C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

And so, the nodal admittance matrix is equal to:

$$Ybus = C^T * Ypr * C = \begin{bmatrix} y_p & 0 & 0 & -y_m & y_m & 0 \\ 0 & y_p & 0 & 0 & -y_m & y_m \\ 0 & 0 & y_p & y_m & 0 & -y_m \\ -y_m & 0 & y_m & 2y_s & -y_s & -y_s \\ y_m & -y_m & 0 & -y_s & 2y_s & -y_s \\ 0 & y_m & -y_m & -y_s & -y_s & 2y_s \end{bmatrix}$$

We can consider this matrix to be formed only by per-unit admittance values for primary windings, secondary windings and couplings, and approximate them to the per-unit leakage admittance of the transformer. As the values now will be expressed in line-to-ground voltage bases, the admittance correspondent to star windings will remain the same, but the admittance correspondent to delta windings must be divided by three. The coupling admittance between star and delta windings must be divided by $\sqrt{3}$. Therefore, for this case the nodal admittance matrix expressed as a function of the per-unit admittance of the transformer is equal to:

$$Y_{bus}^{abc} = \begin{bmatrix} y_t & 0 & 0 & \frac{-y_t}{\sqrt{3}} & \frac{y_t}{\sqrt{3}} & 0 \\ 0 & y_t & 0 & 0 & \frac{-y_t}{\sqrt{3}} & \frac{y_t}{\sqrt{3}} \\ 0 & 0 & y_t & \frac{y_t}{\sqrt{3}} & 0 & \frac{-y_t}{\sqrt{3}} \\ \frac{-y_t}{\sqrt{3}} & 0 & \frac{y_t}{\sqrt{3}} & \frac{2y_t}{3} & \frac{-y_t}{3} & \frac{-y_t}{3} \\ \frac{y_t}{\sqrt{3}} & \frac{-y_t}{\sqrt{3}} & 0 & \frac{-y_t}{3} & \frac{2y_t}{3} & \frac{-y_t}{3} \\ 0 & \frac{y_t}{\sqrt{3}} & \frac{-y_t}{\sqrt{3}} & \frac{-y_t}{3} & \frac{-y_t}{3} & \frac{2y_t}{3} \end{bmatrix}$$

This process can be applied to different transformer connections. In general, any two-winding three-phase transformer may be represented using two coupled compound coils [1]. This representation is illustrated in figure 8:

$$\begin{bmatrix} I_p^{abc} \\ I_s^{abc} \end{bmatrix} = \begin{bmatrix} Y_{pp}^{abc} & Y_{ps}^{abc} \\ Y_{sp}^{abc} & Y_{ss}^{abc} \end{bmatrix} \begin{bmatrix} V_p^{abc} \\ V_s^{abc} \end{bmatrix}$$

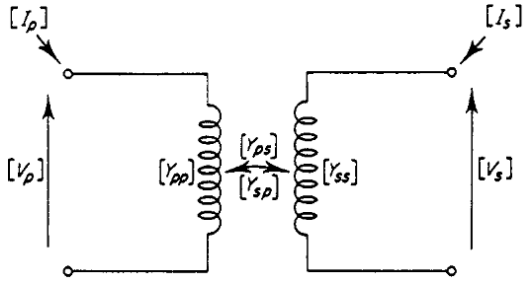


Figure 8. Two-winding three-phase transformer as two coupled compound coils [1]

It is important to notice that $Y_{sp}^{abc} = \left(Y_{ps}^{abc}\right)^T$, as the coupling between coils is bilateral and the nodal admittance matrix is symmetrical.

Table 1, shows the submatrices of the nodal admittance matrix for some of the IEC transformer connections:

Transformer connection	Self-admittance matrices		Mutual admittance matrices	
	Ypp	Yss	Yps	Ysp
YNyn0	Y_I	Y_I	$-Y_I$	$-Y_I$
Yy0	Y_{II}	Y_{II}	$-Y_{II}$	$-Y_{II}$
YNd1	Y_I	Y_{II}	Y_{III}	Y_{III}^T
Yd1	Y_{II}	Y_{II}	Y_{III}	Y_{III}^T
Dyn1	Y_{II}	Y_I	Y_{III}	Y_{III}^T
Dyn11	Y_{II}	Y_I	Y_{III}^T	Y_{III}
Dd0	Y_{II}	Y_{II}	$-Y_{II}$	$-Y_{II}$

Table 1. *Characteristic submatrices for different transformer connections*

Where:

$$Y_I = \begin{bmatrix} y_t & 0 & 0 \\ 0 & y_t & 0 \\ 0 & 0 & y_t \end{bmatrix} \tag{2. 10}$$

$$Y_{II} = \left(\frac{1}{3}\right) \cdot \begin{bmatrix} 2y_t & -y_t & -y_t \\ -y_t & 2y_t & -y_t \\ -y_t & -y_t & 2y_t \end{bmatrix} \tag{2. 11}$$

$$Y_{III} = \left(\frac{1}{\sqrt{3}}\right) \cdot \begin{bmatrix} -y_t & y_t & 0 \\ 0 & -y_t & y_t \\ y_t & 0 & -y_t \end{bmatrix} \quad (2.\ 12)$$

Finally, these submatrices must be modified to account for off-nominal tap ratio. Let $\propto$ and β be the tap variations at the primary and secondary side respect to their nominal values, respectively, then the modifications must be performed as follows:

- Divide the self-admittance of the primary winding by $\propto^2$.
- Divide the self-admittance of the secondary winding by β^2.
- Divide the mutual admittance matrices by $(\propto \cdot \beta)$.

2.8. Load modeling

In a balanced power flow problem, loads are normally represented by a constant power demand connected to each node. However, when the complete three-phase scheme is considered, their characteristics and connections need to be considered. The following two sections explain this with more detail.

2.8.1. Polynomial load models

For transmission systems, there are big loads correspondent to geographical zones. These loads usually present a constant behavior respect to time per day for each season. However, on distribution networks, loads present a different behavior with some dependence on the voltage.

There are different mathematical models for these loads, and among them we have considered the polynomial model as the more adequate for this scenario.

This model specifies the complex power consumption as the summation of elements with constant power, constant current or constant admittance/impedance, as follows:

$$S_{total} = P_{total} + jQ_{total} = \left(P_{e0} + P_{e1} \cdot |V| + P_{e2} \cdot |V|^2\right) + j\left(Q_{e0} + Q_{e1} \cdot |V| + Q_{e2} \cdot |V|^2\right) \quad (2.\ 13)$$

Where:

- $P_{e0} + jQ_{e0}$ is the specified complex power component with constant power
- $P_{e1} + jQ_{e1}$ is the specified complex power component with constant current
- $P_{e2} + jQ_{e2}$ is the specified complex power component with constant admittance
- V is the complex voltage of the node

2.8.2. Load connections

Loads can be connected in star (wye) or delta configurations, as shown in figure 9:

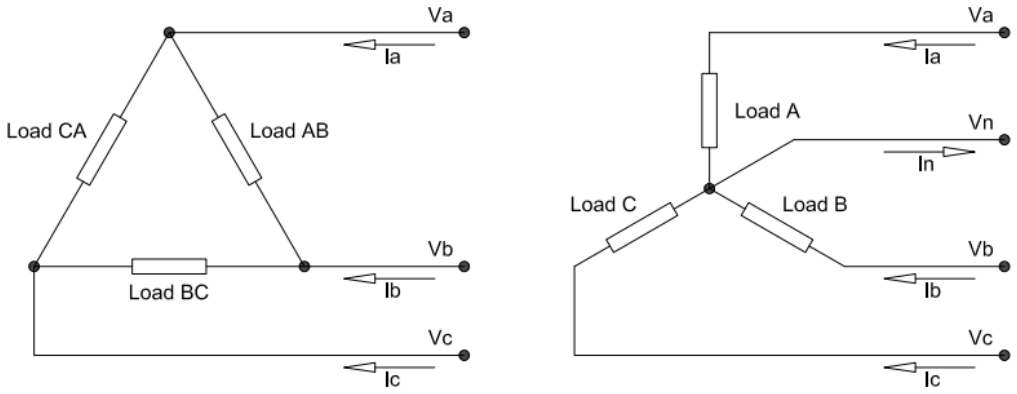


Figure 9. *Delta and star connections for three-phase loads*

Either for star or delta connection, each of the three loads can be represented using the polynomial model, taking into consideration that for star connections all the values will be line-to-ground and for delta connections all the values will be line-to-line.

2.8.3. Single-phase and two-phase loads

As mentioned on 2.2, distribution networks serve many non-three-phase loads. For modelling purpose, loads connected between phases are considered delta-connected even if there are only one or two loads and loads connected between phase and ground are considered wye-connected even if there are one or two loads. The delta or wye arrangement is completed by adding a load or loads equal to zero.

2.9. Capacitor Banks

Capacitor banks are common elements of the distribution network. They are utilized for voltage regulation and reactive power support [5]. They are usually specified by their nominal voltage and nominal reactive power generation. As for other three-phase loads, they can be connected in delta or wye.

2.9.1. Wye-Connected Capacitor Bank

A wye-connected capacitor bank can be modeled as a constant impedance (or admittance) load. The value of its admittance matrix as function of the specified nominal values is equal to:

$$Y_{bank} = j \cdot \begin{bmatrix} B_a & 0 & 0 \\ 0 & B_b & 0 \\ 0 & 0 & B_c \end{bmatrix} = \frac{j}{|V_{LN}|^2} \cdot \begin{bmatrix} Q_a & 0 & 0 \\ 0 & Q_b & 0 \\ 0 & 0 & Q_c \end{bmatrix} \quad (2.\ 14)$$

Where:

- B_a, B_b, B_c are the susceptances per phase of the capacitor units
- Q_a, Q_b, Q_c are the nominal reactive powers per phase
- V_{LN} is the nominal line-to-neutral voltage

Figure 10 shows an example of such capacitor bank:

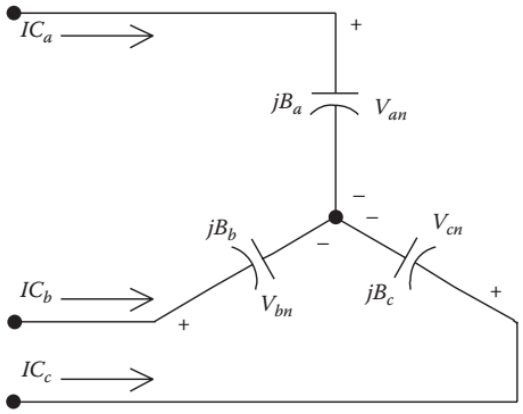


Figure 10. *Wye-connected capacitor bank [5]*

2.9.2. Delta-Connected Capacitor Bank

A delta-connected capacitor bank can also be modeled as a constant impedance (or admittance) load. The value of its admittance matrix as function of the specified nominal values is equal to:

$$Y_{bank} = j \cdot \begin{bmatrix} B_{ab} + B_{ca} & -B_{ab} & -B_{ca} \\ -B_{ab} & B_{ab} + B_{bc} & -B_{bc} \\ -B_{ca} & -B_{bc} & B_{bc} + B_{ca} \end{bmatrix}$$

$$\Rightarrow Y_{bank} = \frac{j}{|V_{LL}|^2} \cdot \begin{bmatrix} Q_{ab} + Q_{ca} & -Q_{ab} & -Q_{ca} \\ -Q_{ab} & Q_{ab} + Q_{bc} & -Q_{bc} \\ -Q_{ca} & -Q_{bc} & Q_{bc} + Q_{ca} \end{bmatrix} \quad (2.\ 15)$$

Where:

- B_{ab}, B_{bc}, B_{ca} are the susceptances of the capacitor units connected between phases
- Q_{ab}, Q_{bc}, Q_{ca} are the nominal reactive powers per unit
- V_{LL} is the nominal line-to-line voltage

Figure 11 shows an example of such capacitor bank:

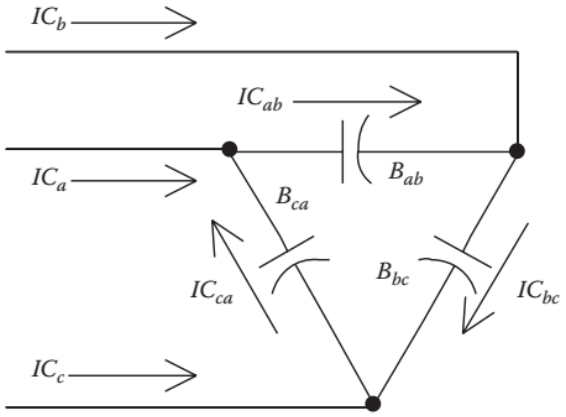


Figure 11. *Delta-connected capacitor bank [5]*

3 Power Flow Calculations Theory

3.1. Introduction

The objective of power flow calculations is to determine the steady-state operating characteristics of the electrical power system for a given set of busbars. Power-flow studies are very important to determine the optimal operation conditions of existing systems, as well as to plan and design power systems expansions [10].

The electrical network may be modeled using the nodal impedance matrix or the nodal admittance matrix. Loads are usually specified by their constant active and reactive power requirement, assumed unaffected by the small variations of voltage and frequency expected during normal steady-state operation.

The solution is expected to provide at least the following information for each bus of the system:

- Bus voltage magnitude
- Bus voltage angle
- Active power flowing through all the lines and branches connected to the bus
- Reactive power flowing through all the lines and branches connected to the bus

Among the different approaches to calculate the quantities mentioned above, two have been selected [10] [11]:

- Power Injection Mismatch Method (PIM)
- Current Injection Mismatch Method (CIM)

The common power flow solution is developed under the assumption that the electrical system is balanced respect to its phases (three-phase system), therefore it can be totally represented by a per-phase model, condition that is frequently met on transmission systems, but not on distribution systems.

3.2. Power flow on distribution systems

Power flow calculations on distribution systems will present some important differences respect to transmission networks:

- These systems become very large and may consist of thousands of buses. Primary distribution feeders and networks, secondary networks and isolated systems shall be modelled simultaneously.

- The system is inherently unbalanced; therefore, all three phases need to be considered during the calculations, causing much greater requirements of storage and computational capacity. The need for efficient algorithms is therefore significant.

- The co-generation from RES (renewable energy sources) and from conventional small generators should be capable of being modeled on primary and secondary systems.

- When all the phases are considered, the specific connections of all the elements play a very important role. Loads are not simply connected to the buses; they can be connected in star or delta to three phases of one bus. Transformers will not be modelled anymore as just an impedance with an angle shift, but their actual connections, i.e. star-delta, delta-delta, etc., modify the calculations.

3.3. Power Injection Mismatch Method (PIM)

3.3.1. Problem solution approach

It is the most widely used methodology to solve the power flow problem, and it relies on the equality to zero of the complex power summation injected into each node [10].

Using nodal notation, the active and reactive power entering to a specific node i can be written as follows:

$$S_i = P_i + jQ_i = V_i \cdot I_i^* = V_i \cdot \left(\sum_{j=1}^{n} Y_{ij} \cdot V_j\right)^* = \sum_{j=1}^{n} \left|V_i \cdot Y_{ij} \cdot V_j\right| \measuredangle\left(\delta_i - \theta_{ij} - \delta_j\right) \quad (3.\ 1)$$

Where:

- $V_i = |V_i| \measuredangle \delta_i$ and $V_j = \left|V_j\right| \measuredangle \delta_j$ are the complex voltage levels on nodes i and j.
- $Y_{ij} = \left|Y_{ij}\right| \measuredangle \theta_{ij}$ is the element on the ith row and the jth column of the nodal admittance matrix.

Therefore, for each node we obtain the following real equations:

$$P_{i,calc} = \sum_{j=1}^{n} \left|V_i \cdot Y_{ij} \cdot V_j\right| cos\left(\delta_i - \theta_{ij} - \delta_j\right) \quad (3.\ 2)$$

$$Q_{i,calc} = \sum_{j=1}^{n} \left|V_i \cdot Y_{ij} \cdot V_j\right| sin\left(\delta_i - \theta_{ij} - \delta_j\right) \quad (3.\ 3)$$

These equations give us the power based on the nodal admittance matrix, therefore based on the characteristics of the network. These powers are called the calculated components of the complex power. Since the net power injection per node must be equal to zero, this calculated complex power must be equal to the scheduled power, composed by the expected load consumption and the expected power generation per node [10].

The power mismatch shall be defined as differences between the calculated and scheduled power injections at each bus:

$$\Delta P_i = P_{i,calc} - P_{i,sch} = P_{i,calc} - \left(P_{i,gen} - P_{i,dem}\right) = 0 \quad (3.\ 4)$$

$$\Delta Q_i = Q_{i,calc} - Q_{i,sch} = Q_{i,calc} - \left(Q_{i,gen} - Q_{i,dem}\right) = 0 \quad (3.\ 5)$$

Finally, the Power Injection Mismatch (PIM) method consists on solving the system of nonlinear equations formed by the power mismatch equations of all nodes, having as unknowns the voltage magnitude and angle on the respective nodes.

3.3.2. Type of nodes

Voltage magnitude, voltage angle, active power injection and reactive power injection are the four unknowns per node of the power injection mismatch equations. However, there are only two power mismatch equations. Therefore, it is required to establish extra conditions for nodes to get only two unknowns per node.

The solution usually used consists on classifying the buses into 3 distinct categories:

- **Slack bus:** Since total network losses are not known in the beginning, a swing (slack) bus is required where the voltage is specified, but the active and reactive power injected into the bus are unknown. The slack bus is usually designated as the voltage reference for the system. The voltage angle of the slack bus serves as reference for the angles of all other bus voltages [10].

- **Voltage-controlled bus (PV bus):** The active power injected into the bus is specified along with the voltage magnitude. PV buses are typically used to represent generators where the active power injected is the generator dispatch power and an automatic voltage regulator controls the bus voltage magnitude to a specified value. PV buses can also be used for reactive power support devices which control the bus voltage to a specified value [10].

- **Load bus (PQ bus):** The active and reactive power injected into (or consumed at) the bus is specified. PQ buses are typically used to represent constant power loads [10].

3.3.3. Extension to unbalanced distribution systems

The model for distribution power networks will be developed as a similar set of buses and branches, but with three nodes (corresponding to the three phases) for each bus. In order proceed with this approach, all the elements of the network have been modelled under the phase frame.

Following this idea, the power flow problem can be defined in an analogous way, only with some differences related to the connection of the loads.

Voltages and power and current injections per bus will be treated as vectors of three elements, and a nodal admittance matrix structured as in point 2.7 will be used. For any node k, the following equation must be accomplished:

$$\Delta S_k^{abc} = S_{k,calc}^{abc} - S_{k,sch}^{abc} = S_{k,calc}^{abc} - (S_{k,gen}^{abc} - SY_{k,dem}^{abc} - S\Delta_{k,dem}^{abc}) = [0] \qquad (3.6)$$

Where:

- a, b, c are the phases of the system
- ΔS_k^{abc} is the complex power mismatch for node k
- $S_{k,calc}^{abc}$ is the calculated complex power injection into node k
- $S_{k,gen}^{abc}$ is the scheduled generation connected to node k
- $SY_{k,dem}^{abc}$ is the scheduled load per phase due to a star load connected to node k
- $S\Delta_{k,dem}^{abc}$ is the scheduled load per phase due to a delta load connected to node k

The calculated power will be computed as follows:

$$S_{k,calc}^{abc} = V_k^{abc} \odot \left(I_k^{abc} \right)^* = V_k^{abc} \odot \sum_{i=1}^{n} \left(Y_{ki}^{abc} \cdot V_i^{abc} \right)^* \quad (3.7)$$

Where:

- $\odot$ represents the Hadamard product (also known as the element-wise product)
- n is the number of nodes
- V_k^{abc} is the vector of bus voltages for the three phases of the k[th] node, 1 ≤ k ≤ n
- I_k^{abc} is the vector of current injections for the three phases of the k[th] node, 1 ≤ k ≤ n
- Y_{ki}^{abc} is the 3x3 block of the admittance matrix correspondent to nodes k and i

The expressions to calculate the specified complex vectors from the usually provided data for generation and demand (the polynomial model) are as follows:

$$S_{k,gen}^{abc} = \begin{bmatrix} S_{k,gen}^{a} \\ S_{k,gen}^{b} \\ S_{k,gen}^{c} \end{bmatrix} \quad (3.8)$$

$$SY_{k,dem}^{abc} = \begin{bmatrix} SY_{k,dem}^{a} \\ SY_{k,dem}^{b} \\ SY_{k,dem}^{c} \end{bmatrix} = \begin{bmatrix} (P_{e0}^{a} + jQ_{e0}^{a}) + (P_{e1}^{a} + jQ_{e1}^{a}) \cdot \left| V_k^{a} \right| + (P_{e2}^{a} + jQ_{e2}^{a}) \cdot \left| V_k^{a} \right|^2 \\ (P_{e0}^{b} + jQ_{e0}^{b}) + (P_{e1}^{b} + jQ_{e1}^{b}) \cdot \left| V_k^{b} \right| + (P_{e2}^{b} + jQ_{e2}^{b}) \cdot \left| V_k^{b} \right|^2 \\ (P_{e0}^{c} + jQ_{e0}^{c}) + (P_{e1}^{c} + jQ_{e1}^{c}) \cdot \left| V_k^{c} \right| + (P_{e2}^{c} + jQ_{e2}^{c}) \cdot \left| V_k^{c} \right|^2 \end{bmatrix} \quad (3.9)$$

$$S\Delta_{k,dem}^{abc} = \begin{bmatrix} S\Delta_{k,dem}^{a} \\ S\Delta_{k,dem}^{b} \\ S\Delta_{k,dem}^{c} \end{bmatrix} = \begin{bmatrix} V_k^{a} \cdot \left(\frac{S\Delta_{k,dem}^{ab}}{V_k^{a} - V_k^{b}} - \frac{S\Delta_{k,dem}^{ca}}{V_k^{c} - V_k^{a}} \right) \\ V_k^{b} \cdot \left(\frac{S\Delta_{k,dem}^{bc}}{V_k^{b} - V_k^{c}} - \frac{S\Delta_{k,dem}^{ab}}{V_k^{a} - V_k^{b}} \right) \\ V_k^{c} \cdot \left(\frac{S\Delta_{k,dem}^{ca}}{V_k^{c} - V_k^{a}} - \frac{S\Delta_{k,dem}^{bc}}{V_k^{b} - V_k^{c}} \right) \end{bmatrix} \quad (3.10)$$

The delta loads from the last expression will be calculated also using the polynomial model, and their values will be equal to:

$$S\Delta_{k,dem}^{ab} = (P_{e0}^{ab} + jQ_{e0}^{ab}) + (P_{e1}^{ab} + jQ_{e1}^{ab}) \cdot \left| V_k^{a} - V_k^{b} \right| / \sqrt{3} + (P_{e2}^{ab} + jQ_{e2}^{ab}) \cdot \left| V_k^{a} - V_k^{b} \right|^2 / 3 \quad (3.11)$$

$$S\Delta_{k,dem}^{bc} = (P_{e0}^{bc} + jQ_{e0}^{bc}) + (P_{e1}^{bc} + jQ_{e1}^{bc}) \cdot \left| V_k^{b} - V_k^{c} \right| / \sqrt{3} + (P_{e2}^{bc} + jQ_{e2}^{bc}) \cdot \left| V_k^{b} - V_k^{c} \right|^2 / 3 \quad (3.12)$$

$$S\Delta_{k,dem}^{ca} = (P_{e0}^{ca} + jQ_{e0}^{ca}) + (P_{e1}^{ca} + jQ_{e1}^{ca}) \cdot \left| V_k^{c} - V_k^{a} \right| / \sqrt{3} + (P_{e2}^{ca} + jQ_{e2}^{ca}) \cdot \left| V_k^{c} - V_k^{a} \right|^2 / 3 \quad (3.13)$$

The nonlinear system of equations will be solved using the Newton-Raphson method.

3.3.4. Newton-Raphson method

Given a vector x of n variables (x_1, x_2, …, x_n), so $x \in \mathbb{R}^n$ and a set of n nonlinear equations, such that:

$$F(x) = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_m(x_1, x_2, \dots, x_n) \end{bmatrix} = 0, \qquad F: \mathbb{R}^n \to \mathbb{R}^n$$

A solution for this nonlinear equation is a vector $x^* \in \mathbb{R}^n$ such that $F(x^*) = 0$. Plenty methods exist to find a solution. However, the Newton-Raphson method is the most widely used method for solving simultaneous, non-linear algebraic equations.

Newton-Raphson method is an iterative procedure based on an initial estimate of the unknown variables and the use of the first terms of a Taylor's series expansion. A linearization of the nonlinear problem around the current iterate is solved in each iteration. The Newton-Raphson algorithm is as follows:

1. $i \leftarrow 0$
2. *Initial iterate value is assigned* x_0
3. ***while*** *not converged* ***do***
4. *solve equation:* $-J(x_i)\Delta x_i = F(x_i)$
5. *update iterate:* $x_{i+1} \leftarrow x_i + \Delta x_i$
6. $i \leftarrow i + 1$
7. ***end while***

The matrix J represents the Jacobian of F:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

3.3.5. Method algorithm

Finally, the algorithm is defined by the following steps:

1. Calculate and specify the nodal admittance matrix of the system Y_{bus}^{abc}, based on the models developed in chapter 2.

2. Select a slack bus for the network. Preferably it should be a large generator responsible for load-frequency control or the point of common coupling to a larger external grid. This bus now will consist of 3 slack bus nodes that as a default case will have the following per-unit values:

$$\text{Slack bus} = \begin{bmatrix} 1\angle 0^{o} \\ 1\angle -120^{o} \\ 1\angle 120^{o} \end{bmatrix}$$

3. For all other buses, define the bus type and specify the known variables.

4. Select initial voltages and angles for all buses. Since now we have three phases, a good initial approximation (flat start) will be:

$$\text{Initial conditions} = \begin{bmatrix} 1\angle 0^{o} \\ 1\angle -120^{o} \\ 1\angle 120^{o} \end{bmatrix}$$

   This value shall be used for all the nodes that are connected to the slack bus without transformers with phase shifts in the path of connection. When there is a transformer with phase shifting, the shift needs to be accounted for the initial condition, or the problem will not converge.

   A much better approximation will be given by firstly solving the power flow problem as a balanced system, with the load equally distributed among the phases and with a per-phase model for all the elements. This value shall be used for the first phase and then the same value with phase shifts of -120 degrees and +120 degrees for the other phases.

5. Calculate the power mismatch vector by using the initial voltages in Step 4 and the specified power injections in Step 3.

6. Apply the Newton-Raphson method to solve the power mismatch equations of all nodes, taken as unknowns the magnitude and angle of the node voltages:

$$\Delta S_k^{abc} = f(|V_k|^{abc}, \delta_k^{abc}) = [0]_{3x1}$$

$$\left(V_k^{abc} = \left[|V_k^a| \angle \delta_k^a \quad |V_k^b| \angle \delta_k^b \quad |V_k^c| \angle \delta_k^c \right]^T \right)$$

   Since the complex mismatch is composed by active and reactive power and apart from the slack bus node (we already know the voltage values for the slack bus) there are n buses with three nodes each one, there are in total 2 * (n-1) * 3 = 6 * n - 6 equations to solve, if all the other (n - 1) buses are PQ.

   Let us assume the first bus is the slack bus. The complex mismatch can be discomposed into active and reactive power mismatch and then we have the following approximation based on Taylor's series expansion:

$$0-\Delta S=-\begin{bmatrix}\Delta P_2^{abc}\\ \Delta P_3^{abc}\\ \vdots\\ \Delta P_n^{abc}\\ \Delta Q_2^{abc}\\ \Delta Q_3^{abc}\\ \vdots\\ \Delta Q_n^{abc}\end{bmatrix}\approx \boldsymbol{J}\cdot\begin{bmatrix}\Delta\delta_2^{abc}\\ \Delta\delta_3^{abc}\\ \vdots\\ \Delta\delta_n^{abc}\\ \Delta|V_2|^{abc}\\ \Delta|V_3|^{abc}\\ \vdots\\ \Delta|V_n|^{abc}\end{bmatrix}$$

Where J is the proper Jacobian matrix.

Based on this approximation, the equation for one iteration (j[th] iteration) of the Newton-Raphson algorithm is written as follows:

$$-\boldsymbol{J}^{(j)}\bullet\begin{bmatrix}\Delta\delta_2^{abc}\\ \Delta\delta_3^{abc}\\ \vdots\\ \Delta\delta_n^{abc}\\ \Delta|V_2|^{abc}\\ \Delta|V_3|^{abc}\\ \vdots\\ \Delta|V_n|^{abc}\end{bmatrix}^{(j)}=\begin{bmatrix}\Delta P_2^{abc}\\ \Delta P_3^{abc}\\ \vdots\\ \Delta P_n^{abc}\\ \Delta Q_2^{abc}\\ \Delta Q_3^{abc}\\ \vdots\\ \Delta Q_n^{abc}\end{bmatrix}^{(j)} \tag{3. 14}$$

And then:

$$\begin{bmatrix}\delta_2^{abc}\\ \delta_3^{abc}\\ \vdots\\ \delta_n^{abc}\end{bmatrix}^{(j+1)}=\begin{bmatrix}\delta_2^{abc}\\ \delta_3^{abc}\\ \vdots\\ \delta_n^{abc}\end{bmatrix}^{(j)}+\begin{bmatrix}\Delta\delta_2^{abc}\\ \Delta\delta_3^{abc}\\ \vdots\\ \Delta\delta_n^{abc}\end{bmatrix}^{(j)} \tag{3. 15}$$

$$\begin{bmatrix}|V_2|^{abc}\\ |V_3|^{abc}\\ \vdots\\ |V_n|^{abc}\end{bmatrix}^{(j+1)}=\begin{bmatrix}|V_2|^{abc}\\ |V_3|^{abc}\\ \vdots\\ |V_n|^{abc}\end{bmatrix}^{(j)}+\begin{bmatrix}\Delta|V_2|^{abc}\\ \Delta|V_3|^{abc}\\ \vdots\\ \Delta|V_n|^{abc}\end{bmatrix}^{(j)} \tag{3. 16}$$

The Jacobian matrix will have the following structure:

$$\begin{bmatrix}
\left[\frac{\partial P_2}{\partial \delta_2}\right]^{abc} & \left[\frac{\partial P_2}{\partial \delta_3}\right]^{abc} & \cdots & \left[\frac{\partial P_2}{\partial \delta_n}\right]^{abc} & \left[\frac{\partial P_2}{\partial |V_2|}\right]^{abc} & \left[\frac{\partial P_2}{\partial |V_3|}\right]^{abc} & \cdots & \left[\frac{\partial P_2}{\partial |V_n|}\right]^{abc} \\
\left[\frac{\partial P_3}{\partial \delta_2}\right]^{abc} & \left[\frac{\partial P_3}{\partial \delta_3}\right]^{abc} & \cdots & \left[\frac{\partial P_3}{\partial \delta_n}\right]^{abc} & \left[\frac{\partial P_3}{\partial |V_2|}\right]^{abc} & \left[\frac{\partial P_3}{\partial |V_3|}\right]^{abc} & \cdots & \left[\frac{\partial P_3}{\partial |V_n|}\right]^{abc} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\left[\frac{\partial P_n}{\partial \delta_2}\right]^{abc} & \left[\frac{\partial P_n}{\partial \delta_3}\right]^{abc} & \cdots & \left[\frac{\partial P_n}{\partial \delta_n}\right]^{abc} & \left[\frac{\partial P_n}{\partial |V_2|}\right]^{abc} & \left[\frac{\partial P_n}{\partial |V_3|}\right]^{abc} & \cdots & \left[\frac{\partial P_n}{\partial |V_n|}\right]^{abc} \\
\left[\frac{\partial Q_2}{\partial \delta_2}\right]^{abc} & \left[\frac{\partial Q_2}{\partial \delta_3}\right]^{abc} & \cdots & \left[\frac{\partial Q_2}{\partial \delta_n}\right]^{abc} & \left[\frac{\partial Q_2}{\partial |V_2|}\right]^{abc} & \left[\frac{\partial Q_2}{\partial |V_3|}\right]^{abc} & \cdots & \left[\frac{\partial Q_2}{\partial |V_n|}\right]^{abc} \\
\left[\frac{\partial Q_3}{\partial \delta_2}\right]^{abc} & \left[\frac{\partial Q_3}{\partial \delta_3}\right]^{abc} & \cdots & \left[\frac{\partial Q_3}{\partial \delta_n}\right]^{abc} & \left[\frac{\partial Q_3}{\partial |V_2|}\right]^{abc} & \left[\frac{\partial Q_3}{\partial |V_3|}\right]^{abc} & \cdots & \left[\frac{\partial Q_3}{\partial |V_n|}\right]^{abc} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\left[\frac{\partial Q_n}{\partial \delta_2}\right]^{abc} & \left[\frac{\partial Q_n}{\partial \delta_3}\right]^{abc} & \cdots & \left[\frac{\partial Q_n}{\partial \delta_n}\right]^{abc} & \left[\frac{\partial Q_n}{\partial |V_2|}\right]^{abc} & \left[\frac{\partial Q_n}{\partial |V_3|}\right]^{abc} & \cdots & \left[\frac{\partial Q_n}{\partial |V_n|}\right]^{abc}
\end{bmatrix}$$

And the submatrices shown above have correspondent to nodes k and m, have the following structure:

$$\left[\frac{\partial P_k}{\partial \delta_m}\right]^{abc} = \begin{bmatrix}
\frac{\partial P_k^a}{\partial \delta_m^a} & \frac{\partial P_k^a}{\partial \delta_m^b} & \frac{\partial P_k^a}{\partial \delta_m^c} \\
\frac{\partial P_k^b}{\partial \delta_m^a} & \frac{\partial P_k^b}{\partial \delta_m^b} & \frac{\partial P_k^b}{\partial \delta_m^c} \\
\frac{\partial P_k^c}{\partial \delta_m^a} & \frac{\partial P_k^c}{\partial \delta_m^b} & \frac{\partial P_k^c}{\partial \delta_m^c}
\end{bmatrix} \qquad \left[\frac{\partial P_k}{\partial |V_m|}\right]^{abc} = \begin{bmatrix}
\frac{\partial P_k^a}{\partial |V_m|^a} & \frac{\partial P_k^a}{\partial |V_m|^b} & \frac{\partial P_k^a}{\partial |V_m|^c} \\
\frac{\partial P_k^b}{\partial |V_m|^a} & \frac{\partial P_k^b}{\partial |V_m|^b} & \frac{\partial P_k^b}{\partial |V_m|^c} \\
\frac{\partial P_k^c}{\partial |V_m|^a} & \frac{\partial P_k^c}{\partial |V_m|^b} & \frac{\partial P_k^c}{\partial |V_m|^c}
\end{bmatrix}$$

$$\left[\frac{\partial Q_k}{\partial \delta_m}\right]^{abc} = \begin{bmatrix}
\frac{\partial Q_k^a}{\partial \delta_m^a} & \frac{\partial Q_k^a}{\partial \delta_m^b} & \frac{\partial Q_k^a}{\partial \delta_m^c} \\
\frac{\partial Q_k^b}{\partial \delta_m^a} & \frac{\partial Q_k^b}{\partial \delta_m^b} & \frac{\partial Q_k^b}{\partial \delta_m^c} \\
\frac{\partial Q_k^c}{\partial \delta_m^a} & \frac{\partial Q_k^c}{\partial \delta_m^b} & \frac{\partial Q_k^c}{\partial \delta_m^c}
\end{bmatrix} \qquad \left[\frac{\partial Q_k}{\partial |V_m|}\right]^{abc} = \begin{bmatrix}
\frac{\partial Q_k^a}{\partial |V_m|^a} & \frac{\partial Q_k^a}{\partial |V_m|^b} & \frac{\partial Q_k^a}{\partial |V_m|^c} \\
\frac{\partial Q_k^b}{\partial |V_m|^a} & \frac{\partial Q_k^b}{\partial |V_m|^b} & \frac{\partial Q_k^b}{\partial |V_m|^c} \\
\frac{\partial Q_k^c}{\partial |V_m|^a} & \frac{\partial Q_k^c}{\partial |V_m|^b} & \frac{\partial Q_k^c}{\partial |V_m|^c}
\end{bmatrix}$$

In case of having also PV buses, the rows and columns of the Jacobian corresponding to their equations involving Δ|V| and ΔQ are eliminated, since in that case the magnitude of the node is known.

7. Re-calculate the power mismatch vector and check if the power mismatches are within a pre-specified error tolerance. If so, then the solution has been reached successfully. If not, then continue iterating until the solution is reached or stop after a maximum number of iterations.

3.4. Current injection mismatch method (CIM)

3.4.1. Motivation for the method

Other different approach to solve the power flow problem consists on applying Kirchhoff's current law to one node: the summation of currents injected to each node must also be always equal to zero. Therefore, one different possibility is a method based on current mismatches.

Using nodal notation, the total current injected to a node k will be equal to:

$$I_{k,calc} = I_{r,k} + jI_{m,k} = \sum_{i=1}^{n} Y_{ki} \cdot V_i =$$
$$\rightarrow I_{k,calc} = \sum_{i=1}^{n} \left[\left(G_{ki} \cdot V_{r,i} - B_{ki} \cdot V_{m,i} \right) + j\left(B_{ki} \cdot V_{r,i} + G_{ki} \cdot V_{m,i} \right) \right] \quad (3.\ 17)$$

Where:

$$Y_{ki} = G_{ki} + \mathrm{j}B_{ki} \ \ and \ \ V_i = V_{r,i} + jV_{m,i}$$

This equation presents an interesting characteristic, and it is that the derivatives of the rectangular components of the current injection into one node respect to the rectangular components of the voltage of another node will depend only on admittance component values. The derivatives of the current injection components in node k respect to the voltage components of node q are equal to:

$$\frac{\partial I_{r,k}}{\partial V_{r,q}} = G_{kq} \quad \frac{\partial I_{r,k}}{\partial V_{m,q}} = -B_{kq}$$
$$\frac{\partial I_{m,k}}{\partial V_{r,q}} = B_{kq} \quad \frac{\partial I_{m,k}}{\partial m_{r,q}} = G_{kq}$$

It means that the correspondent Jacobian for current mismatch equation will be formed mainly by elements from the nodal admittance matrix, and will not need to be updated during each iteration of any algorithm.

We can extend this result to a three-phase case, but with slightly different order, to make the susceptances values be in the diagonal. Susceptances are normally of bigger magnitude than conductances, making the matrix diagonally dominant:

$$\left[\frac{\partial I_k}{\partial V_q}\right]^{abc} = \begin{bmatrix} \frac{\partial I^a_{m,k}}{\partial V^a_{r,q}} & \frac{\partial I^a_{m,k}}{\partial V^b_{r,q}} & \frac{\partial I^a_{m,k}}{\partial V^c_{r,q}} & \frac{\partial I^a_{m,k}}{\partial V^a_{m,q}} & \frac{\partial I^a_{m,k}}{\partial V^b_{m,q}} & \frac{\partial I^a_{m,k}}{\partial V^c_{m,q}} \\ \frac{\partial I^b_{m,k}}{\partial V^a_{r,q}} & \frac{\partial I^b_{m,k}}{\partial V^b_{r,q}} & \frac{\partial I^b_{m,k}}{\partial V^c_{r,q}} & \frac{\partial I^b_{m,k}}{\partial V^a_{m,q}} & \frac{\partial I^b_{m,k}}{\partial V^b_{m,q}} & \frac{\partial I^b_{m,k}}{\partial V^c_{m,q}} \\ \frac{\partial I^c_{m,k}}{\partial V^a_{r,q}} & \frac{\partial I^c_{m,k}}{\partial V^b_{r,q}} & \frac{\partial I^c_{m,k}}{\partial V^c_{r,q}} & \frac{\partial I^c_{m,k}}{\partial V^a_{m,q}} & \frac{\partial I^c_{m,k}}{\partial V^b_{m,q}} & \frac{\partial I^c_{m,k}}{\partial V^c_{m,q}} \\ \frac{\partial I^a_{r,k}}{\partial V^a_{r,q}} & \frac{\partial I^a_{r,k}}{\partial V^b_{r,q}} & \frac{\partial I^a_{r,k}}{\partial V^c_{r,q}} & \frac{\partial I^a_{r,k}}{\partial V^a_{m,q}} & \frac{\partial I^a_{r,k}}{\partial V^b_{m,q}} & \frac{\partial I^a_{r,k}}{\partial V^c_{m,q}} \\ \frac{\partial I^b_{r,k}}{\partial V^a_{r,q}} & \frac{\partial I^b_{r,k}}{\partial V^b_{r,q}} & \frac{\partial I^b_{r,k}}{\partial V^c_{r,q}} & \frac{\partial I^b_{r,k}}{\partial V^a_{m,q}} & \frac{\partial I^b_{r,k}}{\partial V^b_{m,q}} & \frac{\partial I^b_{r,k}}{\partial V^c_{m,q}} \\ \frac{\partial I^c_{r,k}}{\partial V^a_{r,q}} & \frac{\partial I^c_{r,k}}{\partial V^b_{r,q}} & \frac{\partial I^c_{r,k}}{\partial V^c_{r,q}} & \frac{\partial I^c_{r,k}}{\partial V^a_{m,q}} & \frac{\partial I^c_{r,k}}{\partial V^b_{m,q}} & \frac{\partial I^c_{r,k}}{\partial V^c_{m,q}} \end{bmatrix} = \begin{bmatrix} B^{abc}_{kq} & G^{abc}_{kq} \\ G^{abc}_{kq} & -B^{abc}_{kq} \end{bmatrix}$$

Where: $Y^{abc}_{kq} = G^{abc}_{kq} + jB^{abc}_{kq}$ is the correspondent element of the three-phase nodal admittance matrix shown in point 2.7.

3.4.2. Problem solution approach

As the power mismatch definition, we will define a current injection mismatch equation per node:

$$\Delta I^{abc}_k = I^{abc}_{k,calc} - I^{abc}_{k,sch} = \left(I^{abc}_{k,calc}\right) - (I^{abc}_{k,gen} - IY^{abc}_{k,dem} - I\Delta^{abc}_{k,dem}) = [0] \qquad (3.\ 18)$$

Where:

- a, b, c are the phases of the system
- ΔI^{abc}_k is the complex power mismatch for node k
- $I^{abc}_{k,calc}$ is the calculated complex current injection into node k
- $I^{abc}_{k,gen}$ is the scheduled generation current to node k
- $IY^{abc}_{k,dem}$ is the load current per phase due to a star load connected to node k
- $I\Delta^{abc}_{k,dem}$ is the load current per phase due to a delta load connected to node k

The calculated current injection will be computed as follows:

$$I^{abc}_{k,calc} = \sum_{i=1}^{n}\left(Y^{abc}_{ki} \cdot V^{abc}_i\right) \qquad (3.\ 19)$$

Where:

- n is the number of nodes
- V^{abc}_k is the vector of bus voltages for the three phases of the k[th] node, 1 ≤ k ≤ n
- $Y^{abc}_{ki}$is the 3x3 block of the admittance matrix correspondent to nodes k and i

For the following expressions, the Hadamard division (also known as the element-wise division) will be used and represented with the symbol ⊘. The expressions to calculate the specified complex current injections from the usually provided data for generation and demand (polynomial model) are as follows:

$$I^{abc}_{k,\,generated} = \left(S^{abc}_{k,\,gen} \oslash V^{abc}_{k} \right)^{*} \tag{3. 20}$$

$$IY^{abc}_{k,dem} = \left(SY^{abc}_{k,dem} \oslash V^{abc}_{k} \right)^{*} = \begin{bmatrix} \left(\dfrac{(P^{a}_{e0} + jQ^{a}_{e0}) + (P^{a}_{e1} + jQ^{a}_{e1}) \cdot \left| V^{a}_{k} \right| + (P^{a}_{e2} + jQ^{a}_{e2}) \cdot \left| V^{a}_{k} \right|^{2}}{V^{a}_{k}} \right)^{*} \\ \left(\dfrac{(P^{b}_{e0} + jQ^{b}_{e0}) + (P^{b}_{e1} + jQ^{b}_{e1}) \cdot \left| V^{b}_{k} \right| + (P^{b}_{e2} + jQ^{b}_{e2}) \cdot \left| V^{b}_{k} \right|^{2}}{V^{b}_{k}} \right)^{*} \\ \left(\dfrac{(P^{c}_{e0} + jQ^{c}_{e0}) + (P^{c}_{e1} + jQ^{c}_{e1}) \cdot \left| V^{c}_{k} \right| + (P^{c}_{e2} + jQ^{c}_{e2}) \cdot \left| V^{c}_{k} \right|^{2}}{V^{c}_{k}} \right)^{*} \end{bmatrix} \tag{3. 21}$$

$$I\Delta^{abc}_{k,demended} = \begin{bmatrix} \left(\dfrac{S\Delta^{ab}_{k,dem}}{V^{a}_{k} - V^{b}_{k}} - \dfrac{S\Delta^{ca}_{k,dem}}{V^{c}_{k} - V^{a}_{k}} \right)^{*} \\ \left(\dfrac{S\Delta^{bc}_{k,dem}}{V^{b}_{k} - V^{c}_{k}} - \dfrac{S\Delta^{ab}_{k,dem}}{V^{a}_{k} - V^{b}_{k}} \right)^{*} \\ \left(\dfrac{S\Delta^{ca}_{k,dem}}{V^{c}_{k} - V^{a}_{k}} - \dfrac{S\Delta^{bc}_{k,dem}}{V^{b}_{k} - V^{c}_{k}} \right)^{*} \end{bmatrix} \tag{3. 22}$$

The delta loads from the last expression will be calculated using equations 3.11, 3.12 and 3.13.

The current mismatch injection will become a set of non-linear equations when considering all the nodes and the real and imaginary part of the currents. This set of equations will be solved to find the real and imaginary components of the voltage per node as unknowns.

3.4.3. Method algorithm

The algorithm is defined by the following steps:

1. Calculate and specify the nodal admittance matrix of the system Y^{abc}_{bus}.

2. Select a slack bus for the network. The default value is the same as for the PIM method:

$$\text{Slack bus} = \begin{bmatrix} 1\angle 0^{\text{o}} \\ 1\angle -120^{\text{o}} \\ 1\angle 120^{\text{o}} \end{bmatrix}$$

3. For all other buses, define the bus type and specify the known variables.

4. Select initial voltages and angles for all buses in the same way that it is done for the Power Injection Mismatch method.

5. Calculate the current injection mismatch vector by using the initial voltages in Step 4 and the specified power injections in Step 3.

6. Apply the Newton-Raphson method solve the system of nonlinear equations for the current injection mismatch per bus (bus k) and obtained the next iteration of the bus voltages:

$$\Delta I_k^{abc} = f(V_{r,k}^{abc}, V_{m,k}^{abc}) = [0]_{3x1}$$

$$\left(V_k^{abc} = \begin{bmatrix} V_{r,k}^a + jV_{m,k}^a & V_{r,k}^b + jV_{m,k}^b & V_{r,k}^c + jV_{m,k}^c \end{bmatrix}^T \right)$$

Let us assume the first bus is the slack bus. The current injection mismatch can be discomposed into real and imaginary part and then we have the following approximation based on Taylor's series expansion:

$$0 - \Delta S = -\begin{bmatrix} I_{m,2}^{abc} \\ I_{r,2}^{abc} \\ I_{m,3}^{abc} \\ I_{r,3}^{abc} \\ \vdots \\ I_{m,n}^{abc} \\ I_{r,n}^{abc} \end{bmatrix} \approx \boldsymbol{J} \cdot \begin{bmatrix} V_{r,2}^{abc} \\ V_{m,2}^{abc} \\ V_{r,3}^{abc} \\ V_{m,3}^{abc} \\ \vdots \\ V_{r,n}^{abc} \\ V_{m,n}^{a} \end{bmatrix}$$

Where J is the proper Jacobian matrix.

Based on this approximation, the equation for one iteration (j[th] iteration) of the Newton-Raphson algorithm is written as follows:

$$-\boldsymbol{J}^{(i)} \cdot \begin{bmatrix} V_{r,2}^{abc} \\ V_{m,2}^{abc} \\ V_{r,3}^{abc} \\ V_{m,3}^{abc} \\ \vdots \\ V_{r,n}^{abc} \\ V_{m,n}^{a} \end{bmatrix}^{(i)} = \begin{bmatrix} I_{m,2}^{abc} \\ I_{r,2}^{abc} \\ I_{m,3}^{abc} \\ I_{r,3}^{abc} \\ \vdots \\ I_{m,n}^{abc} \\ I_{r,n}^{abc} \end{bmatrix}^{(i)} \tag{3. 23}$$

And then:

$$\begin{bmatrix} V_{r,2}^{abc} \\ V_{r,3}^{abc} \\ \vdots \\ V_{r,n}^{abc} \end{bmatrix}^{(j+1)} = \begin{bmatrix} V_{r,2}^{abc} \\ V_{r,3}^{abc} \\ \vdots \\ V_{r,n}^{abc} \end{bmatrix}^{(j)} + \begin{bmatrix} \Delta V_{r,2}^{abc} \\ \Delta V_{r,3}^{abc} \\ \vdots \\ \Delta V_{r,n}^{abc} \end{bmatrix}^{(j)} \tag{3. 24}$$

$$\begin{bmatrix} V_{m,2}^{abc} \\ V_{m,3}^{abc} \\ \vdots \\ V_{m,n}^{a} \end{bmatrix}^{(j+1)} = \begin{bmatrix} V_{m,2}^{abc} \\ V_{m,3}^{abc} \\ \vdots \\ V_{m,n}^{a} \end{bmatrix}^{(j)} + \begin{bmatrix} \Delta V_{m,2}^{abc} \\ \Delta V_{m,3}^{abc} \\ \vdots \\ \Delta V_{m,n}^{a} \end{bmatrix}^{(j)} \quad (3.\ 25)$$

The Jacobian matrix will have the following structure:

$$\begin{bmatrix} \left[\frac{\partial I_2}{\partial V_2}\right]^{abc} & \left[\frac{\partial I_2}{\partial V_3}\right]^{abc} & \cdots & \left[\frac{\partial I_2}{\partial V_n}\right]^{abc} \\ \left[\frac{\partial I_3}{\partial V_2}\right]^{abc} & \left[\frac{\partial I_3}{\partial V_3}\right]^{abc} & \cdots & \left[\frac{\partial I_3}{\partial V_n}\right]^{abc} \\ \vdots & \vdots & \ddots & \vdots \\ \left[\frac{\partial I_n}{\partial V_2}\right]^{abc} & \left[\frac{\partial I_n}{\partial V_3}\right]^{abc} & \cdots & \left[\frac{\partial I_n}{\partial V_n}\right]^{abc} \end{bmatrix}$$

And the submatrices will have the structure shown in the end of point 3.4.1 for all the off-diagonal submatrices. What means the value of all these elements is constant for all the iterations and equal to susceptances and conductances.

For the submatrices in the diagonal, a slight modification will be required because the demanded power in the power mismatch is a function of the voltage (Polynomial model) and this value will affect only the derivative of current injections respect to voltages of the same node. It means that most of the elements of the Jacobian will not be updated per iteration, reducing the computational load.

7. Re-calculate the current injection mismatch vector and check if the current mismatches are within a pre-specified error tolerance. If so, then the solution has been reached successfully. If not, then continue iterating until the solution is reached or stop after a maximum number of iterations.

3.5. Special treatment of constant-impedance loads

The power demand by constant impedance/admittance loads is proportional to the square of the magnitude of the voltage applied to them, as shown in equation 2.13. Therefore, they add computational load to the calculation of the objective function and the Jacobian matrix in each iteration. For this reason, in the present project they are treated as admittance values to add to the nodal admittance matrix.

Equation 2.13 applied to a load with only constant-impedance component shows that:

$$S_{load} = P_{e2} \cdot |V|^2 + jQ_{e2} \cdot |V|^2 \quad (3.\ 26)$$

The complex power consumed by any admittance is equal to:

$$S_{\text{admittance}} = V \cdot I^* = |V|^2 \cdot Y^* = G \cdot |V|^2 - jB \cdot |V|^2 \quad (3.\ 27)$$

By comparing equations 3.26 and 3.27, we can observe that given the nominal active and reactive power components of the constant impedance load, the correspondent admittance of such load, in per-unit values, is equal to:

$$G_{p.u.} = \frac{P_{e2}}{S_{base}} \quad (3.\ 28)$$

$$B_{p.u.} = -\frac{Q_{e2}}{S_{base}} \quad (3.\ 29)$$

3.6. Method selection

Both the PIM and the CIM methods are able to provide a solution for the power flow problem, however, for the current development, the Power Injection Mismatch method has been selected because of the following reasons:

- Most derivatives in the CIM method are constant for PQ nodes. However, on PV nodes, where the magnitude of the voltage is a constant, the derivatives respect to real and imaginary parts of the voltage become very complex, affecting the convergence of the method and losing the speed advantage given by having a Jacobian matrix with constant elements. In a common distribution network, there will not be many PV nodes, nevertheless in future networks with distributed renewable sources this may change, therefore the PIM method that eliminates the derivatives respect to the voltage magnitude in PV nodes have been selected as more convenient.

- In the case of delta connected loads, the power injection into a phase depends of the voltage in other phases from the same node. It derives into complex derivatives when using the CIM method.

- PIM method is a simpler extension of the traditional power flow solution methodology.

3.7. Calculation of power injections and branch quantities

Since the PIM method has been selected for this project, the magnitudes and angles of the voltage nodes are found by solving the respective equations. After that, the power injection for all nodes are found using the nodal admittance matrix as follows:

$$S^{abc} = V^{abc} \odot \left(Y^{abc} \cdot V^{abc} \right)^{*} \quad (3.\ 30)$$

Where:

- $\odot$ represents the Hadamard product
- V^{abc} is the vector of bus voltages for the three phases of all nodes
- Y^{abc} is the nodal admittance matrix of the network
- S^{abc} is the vector of power injections for the three phases of all nodes

To calculate the branch currents and the losses of the network, we have used the connection matrix and the primitive admittance matrix (both defined in section 2.3.2) as follows:

$$I_{br}^{abc} = Y_{br}^{abc} \cdot \left(C^{abc} \cdot V^{abc} \right) \quad (3.\ 31)$$

$$S_{losses}^{abc} = \left(C^{abc} \cdot V^{abc} \right) \odot \left(I_{br}^{abc} \right)^{*} \quad (3.\ 32)$$

Where:

- $\odot$ represents the Hadamard product
- I_{br}^{abc} is the vector of branch currents
- V^{abc} is the vector of bus voltages for the three phases of all nodes
- C^{abc} is the matrix of connections of the network
- Y_{br}^{abc} is the primitive admittance matrix
- S_{losses}^{abc} is the vector of power losses of the three phases of all branches

4 PETSc and its use on power flow computations

4.1. Introduction

The power flow problem consists mainly on solving the nonlinear equations of the active and reactive power mismatch or the real and imaginary part of the current mismatch.

Open-source tools to solve this problem do exist. A very good example is MATPOWER [16] that is a package of MATLAB M-files that deals with solving power-flow problems. It offers some traditional methods for solving the nonlinear system derived from the power flow problem, that is especially suitable to small cases. However, the nature of existent and future new large-scale problems justifies the search for software tools that allow to perform faster and taking advantage of their parallel nature.

One toolkit that is very suitable to handle these operations is PETSc, that stands for Portable, Extensible Toolkit for Scientific Computation [2] [3] [4].

4.2. Direct and iterative solution of linear equation systems

Given a non-singular coefficient matrix $A \in \mathbb{R}^{nxn}$ and a right-hand side vector $f \in \mathbb{R}^n$, the goal of linear solver is to find a vector $x \in \mathbb{R}^n$, such that $Ax = f$ [18].

Each iteration of the Newton-Raphson method and other nonlinear solution methods consist on solving a linear equation system. The computation time of the power flow problem vastly depends on the computation time required to solve these linear systems. Furthermore, when a system becomes larger, the common direct solution approach may not perform well. Also, the characteristics of the matrix of coefficients may affect its performance.

Given these important consideration, let us give more information about the solution of linear equation systems.

There are two main approaches to solve these systems: direct and iterative solvers.

4.2.1. Direct solvers

A direct solver consists of a method to calculate the exact solution vector $\boldsymbol{x}$ directly using methods as:

- Computation of the inverse coefficient matrix A^{-1}, after which the solution of the linear system can simply be found by calculating the vector $x = A^{-1}f$. This approach is commonly not used because computing the inverse is very costly and the solution found is usually not accurate.

- Forward and backward substitutions: Consists on solving one of the n equations for one of the unknowns and use this value to reduce the system to one with n-1 equations and n-1 equations. Then the same procedure is followed until explicit value of one unknown is found and from there, the explicit values of the other unknowns can be found.

In practice, it is generally more efficient to build a factorization of the coefficient matrix into triangular matrices, which can be used to easily derive the solution. For general matrices, the factorization of choice is the LU decomposition [18].

The LU decomposition consists of a lower triangular matrix *L*, and an upper triangular matrix *U*, such that *LU* = *A*. These factors are unique if the requirement is added that all the diagonal elements of either *L*, or of *U*, are ones [18]. Using the LU decomposition, the system of linear equations can be written as: *LUx* = *f*, and solved by consecutively solving the two linear systems:

$$Ly = f$$

$$Ux = y$$

Because *L* and *U* are triangular, these systems are quickly solved using forward and backward substitution respectively [18].

The algorithms of direct solvers lead to an exact solution in exact arithmetic. However, though the algorithms may be exact, the computers that execute them are not. Finite precision arithmetic may still introduce errors in the solution calculated by a direct solver.

During the factorization process, rounding errors may lead to substantial inaccuracies in the factors. Errors in the factors can, in turn, lead to errors in the solution vector calculated by forward and backward substitution.

Forward and backward substitution operations have a complexity degree proportional to the number of nonzero elements of the matrix of coefficients *A*. For full coefficient matrices, the complexity of the LU decomposition is proportional to the third power of the number of equations. For sparse matrix systems, special sparse methods improve on this, by exploiting the sparsity structure of the coefficient matrix.

4.2.2. Iterative solvers

We denote the sequence of iterands by:

$$\{x_k\}_{k\geq 0} \text{ where } x_k \to x^* \text{ for } k \to \infty$$

where x_0 and x* denotes the initial guess and exact solution of the linear system, respectively.

With each iterate we associate the distance to the solution or *error* vector:

$$e_k = x^* - x_k$$

To construct an iterative scheme, we assume that a non-singular matrix *M* exists and define the matrix *N* as *N* = *M* - *A*. We can then write *A* = *M* – *N.*

The linear system *Ax* = *f* can then be written as *Mx* = *Nx* + *f*. By multiplying to the left and right by M^{-1} we can define an iterative scheme:

$$u_{k+1} = M^{-1}Nu_k + M^{-1}f$$
$$u_{k+1} = M^{-1}(M - A)u_k + M^{-1}f$$
$$u_{k+1} = u_k + M^{-1}(f - Au_k)$$
$$u_{k+1} = u_k + M^{-1}r_k$$

In this way, an approximated solution that depends on the distance on the residue and the solution from the previous iteration can be defined. More information regarding iterative solvers can be found in [21].

Iterative solvers are more efficient on large sparse coefficient matrix than direct solvers. It is important to remark that the nodal admittance matrix is usually a sparse matrix, since there are not connections between all the nodes of an electric system. Distribution networks are more prone to generate sparse matrices because they are not meshed, furthermore in most of the cases they have a radial structure.

4.2.3. Krylov subspace method and preconditioning

The Krylov subspace of dimension i, belonging to A and r_0, is defined as:

$$\kappa_i(A, r_0) = span\{r_0, Ar_0, ..., A^{i-1}r_0\}$$

Krylov subspace methods are iterative linear solvers that generate iterates:

$$x_i \in x_0 + \kappa_i(A, r_0)$$

The simplest Krylov method consists of the Richardson iterations:

$$x_{i+1} = x_i + r_i$$

These methods are more complex than the basic iterative methods, but generally converge faster to a solution. Two important properties of Krylov method are the optimality property and short recurrences [18].

A Krylov method is said to have the optimality property, if in each iteration the computed iterate is the best possible approximation of the solution within current the Krylov subspace. An iterative process is said to have short recurrences if in each iteration only data from a small fixed number of previous iterations is used [18].

A special mention is required for preconditioning of the method. No Krylov subspace method can produce iterates that are better than the best approximation of the solution within the progressive Krylov subspaces, which are the iterates attained by minimal residual methods. In other words, the convergence of a Krylov subspace method is limited by the Krylov subspace. Preconditioning uses a preconditioner matrix M to change the Krylov subspace to improve convergence of the iterative solver [18]. More information regarding Krylov methods ca be found in [21].

4.3. Portable, Extensible Toolkit for Scientific Computation - PETSc

PETSc is an opensource toolkit which provides data structures and routines for the numerical solution of scientific problems that can be modeled by systems of differential equations. As the target of PETSc are large-scale problems for high-performance distributed computers, it supports sequential and parallel execution. The communication protocol that is used by its functions is MPI (Message Passing Interface) [2] [3] [4].

Even though, it was intended primarily for solving differential equations, PETSc provides a variety of equation solvers for linear and non-linear systems using Krylov subspace methods and Newton techniques that may be very useful for the power flow problem.

The main modules of PETSc that are useful for solving the power flow problem will be explored in the following section.

4.4. PETSc modules and their different options

4.4.1. The scalable linear equation solvers (KSP)

KSP is the abstract object that manages all Krylov methods. It manages the linear solvers in PETSc (even those such as direct solvers that do not use Krylov accelerators).

It is the heart of PETSc, because it provides uniform and efficient access to all the package's linear system solvers, including parallel and sequential, direct and iterative [2] [3] [4].

KSP is intended for solving nonsingular systems of the form $Ax = b$, where A denotes the matrix representation of a linear operator, b is the right-hand-side vector, and x is the solution vector. KSP uses the same calling sequence for both direct and iterative solution of a linear system [2] [3] [4].

Different linear methods are available in PETSc and can be selected by a command from the database. Some of these methods are:

- Richardson
- Chebyshev
- Conjugate gradients
- Generalized minimal residual (GMRES)
- Bi-conjugate gradient
- Stabilized bi-conjugate gradient
- Conjugate gradient squared
- Transpose free quasi-minimal residual
- Conjugate residuals
- LSQR
- Use only the preconditioner

The combination of a Krylov subspace method and a preconditioner is at the center of most modern numerical codes for the iterative solution of linear systems. The PC module includes a variety of preconditioner options:

- Jacobi (i.e. diagonal scaling preconditioning)
- Block Jacobi (which allows to parallelize the problem to solve)

- SOR (successive over relaxation, Gauss-Seidel)
- Incomplete Cholesky
- Incomplete LU
- Additive Schwarz
- Generalized Additive Schwarz
- Algebraic Multigrid
- Balancing Domain Decomposition by Constraints
- Linear solver
- Combination of preconditioners
- LU
- Cholesky
- No preconditioning
- Shell for user-defined preconditioner

4.4.2. The Scalable Nonlinear Equations Solvers (SNES)

The SNES library of PETSc provides a powerful suite of data structure-neutral numerical routines for solving large-scale nonlinear problems. It is built on top of the linear solvers and data structures [2] [3] [4].

The SNES interface is identical for the uniprocess and parallel cases; the only difference in the parallel version is that each process typically forms only its local contribution to various matrices and vectors. For solving non-linear systems, the SNES library is used which is based on Newton methods including line-search strategies, trust region approaches, pseudo-transient continuation and matrix-free variants. SNES is built on top of KSP and other data structures, and internally it employs KSP linear solvers and preconditioners [2] [3] [4].

The SNES module allows to select the type of nonlinear solver to be utilized. The different solvers are shown below:

- Newton's method
- Newton's method with trust region
- Single linearization
- Quasi-Newton method
- Nonlinear conjugate-gradient
- Nonlinear generalized minimum residual
- Anderson mixing
- Nonlinear Richardson
- Nonlinear Gauss-Seidel
- Full Approximation Scheme (nonlinear multigrid)
- Nonlinear additive Schwarz
- Combine several nonlinear solvers

The SNES module allows to select the parameters to finalize the nonlinear iterations:

- The absolute tolerance, defined as the absolute size of the residual norm.
- The relative tolerance, defined as the decrease of the residual norm relative to the norm of the right-hand side.

5 Program Implementation

5.1. Implementation approach

This chapter will provide details on the implementation of the program. It has consisted of two main steps:

1. A prototype using MATLAB was written. This prototype has intended to test the mathematical models of the different elements of the network. The validity of the code is tested using IEEE test feeders and the details of the implementation procedure and algorithms are given in section 5.2.

2. The final program has been written using C code and the PETSc library. Its validity will be tested using the same IEEE test feeders and the details of this implementation will be given in 5.3.

The results of the tests over the IEEE test feeders are provided in chapter 6.

5.2. MATLAB prototyping

This implementation has similar format to the existent power flow solver MATPOWER. It is a MATLAB Power System Simulation Package, developed by Zimmerman, Murillo-Sánchez & others [16].

Given the input information presented in section 5.2.1, the program calculates the following values:

1. The voltage magnitude and angle for each phase of all buses.
2. The active and reactive power injection into each phase of all buses.
3. The magnitude and angle of the current through each phase of each branch.
4. The active and reactive power losses through each phase of each branch.
5. The total amount of power losses existent in the network.

5.2.1. Input information

The first step for the prototype has been to determine which information from the system is required to compute the output values. The information will correspond to the four key elements of the network: buses, branches, loads and generation.

These values will be given to the program in an input MATLAB file (.m) that has a format similar to that used by MATPOWER. It has been designed in this way to make it easier to be used by researchers (many of them are used to MATPOWER). Details on the format of the input file are given in the file '*distributionCaseFormat.m'*, present in the repository for this development.

The input file consists of six fields:

- **Single-phase base power**. It is expressed in Megavolt-amperes (MVA).

- **Bus data matrix.** Each row of this matrix contains data for one bus. The columns determine the properties of the bus and are as follows:

  1. Bus ID: Identifier (Positive integer)
  2. Type
     PQ bus = 1
     PV bus = 2
     reference bus = 3
     isolated bus = 4
  3. Voltage magnitude on phase A (per unit value)
  4. Voltage magnitude on phase B (per unit value)
  5. Voltage magnitude on phase C (per unit value)
  6. Voltage angle on phase A (degrees)
  7. Voltage angle on phase B (degrees)
  8. Voltage angle on phase C (degrees)
  9. Single-phase base voltage (kV)

- **Generator data matrix.** Each row of this matrix contains data for one generator. The columns determine the properties of the generator and are as follows:

  1. Bus to which the generator/generators is/are connected
  2. Active power injection into phase A (MW)
  3. Active power injection into phase B (MW)
  4. Active power injection into phase C (MW)
  5. Reactive power injection into phase A (MVAR)
  6. Reactive power injection into phase B (MVAR)
  7. Reactive power injection into phase C (MVAR)
  8. Status, 0 is OFF, 1 is ON

- **Wye-connected load data matrix.** Each row of this matrix contains data for one load. The columns determine the properties of the load and are as follows:

  1. Bus to which the load is connected
  2. Phase-A active power load, power-constant component (MW)
  3. Phase-B active power load, power-constant component (MW)
  4. Phase-C active power load, power-constant component (MW)
  5. Phase-A reactive power load, power-constant component (MVAR)
  6. Phase-B reactive power load, power-constant component (MVAR)
  7. Phase-C reactive power load, power-constant component (MVAR)
  8. Phase-A active power load, current-constant component (MW)
  9. Phase-B active power load, current-constant component (MW)
  10. Phase-C active power load, current-constant component (MW)
  11. Phase-A reactive power load, current-constant component (MVAR)
  12. Phase-B reactive power load, current-constant component (MVAR)
  13. Phase-C reactive power load, current-constant component (MVAR)
  14. Phase-A active power load, impedance-constant component (MW)
  15. Phase-B active power load, impedance-constant component (MW)
  16. Phase-C active power load, impedance-constant component (MW)
  17. Phase-A reactive power load, impedance-constant component (MVAR)
  18. Phase-B reactive power load, impedance-constant component (MVAR)
  19. Phase-C reactive power load, impedance-constant component (MVAR)
  20. Status, 0 is OFF, 1 is ON

- **Delta-connected load data matrix.** Each row of this matrix contains data for one load. The columns determine the properties of the load and are as follows:

    1. Bus to which the load is connected
    2. Active power load between phases A and B, power-constant component (MW)
    3. Active power load between phases B and C, power-constant component (MW)
    4. Active power load between phases C and A, power-constant component (MW)
    5. Reactive power load between phases A and B, power-constant component (MVAR)
    6. Reactive power load between phases B and C, power-constant component (MVAR)
    7. Reactive power load between phases C and A, power-constant component (MVAR)
    8. Active power load between phases A and B, current-constant component (MW)
    9. Active power load between phases B and C, current-constant component (MW)
    10. Active power load between phases C and A, current-constant component (MW)
    11. Reactive power load between phases A and B, current-constant component (MVAR)
    12. Reactive power load between phases B and C, current-constant component (MVAR)
    13. Reactive power load between phases C and A, current-constant component (MVAR)
    14. Active power load between phases A and B, impedance-constant component (MW)
    15. Active power load between phases B and C, impedance-constant component (MW)
    16. Active power load between phases C and A, impedance-constant component (MW)
    17. Reactive power load between phases A and B, impedance-constant component (MVAR)
    18. Reactive power load between phases B and C, impedance-constant component (MVAR)
    19. Reactive power load between phases C and A, impedance-constant component (MVAR)
    20. Status, 0 is OFF, 1 is ON

- **Line-branch data matrix.** Each row of this matrix contains data for one branch. The columns determine the properties of the line branch and are as follows:

    1. First bus end of the line
    2. Second bus end of the line
    3. Raa, element (1,1) of the primitive resistance matrix (per unit value)
    4. Rab, element (1,2) of the primitive resistance matrix (per unit value)
    5. Rac, element (1,3) of the primitive resistance matrix (per unit value)
    6. Rbb, element (2,2) of the primitive resistance matrix (per unit value)
    7. Rbc, element (2,3) of the primitive resistance matrix (per unit value)
    8. Rcc, element (3,3) of the primitive resistance matrix (per unit value)
    9. Xaa, element (1,1) of the primitive reactance matrix (per unit value)
    10. Xab, element (1,2) of the primitive reactance matrix (per unit value)
    11. Xac, element (1,3) of the primitive reactance matrix (per unit value)
    12. Xbb, element (2,2) of the primitive reactance matrix (per unit value)
    13. Xbc, element (2,3) of the primitive reactance matrix (per unit value)
    14. Xcc, element (3,3) of the primitive reactance matrix (per unit value)
    15. Baa, element (1,1) of the primitive susceptance matrix (per unit value)
    16. Bab, element (1,2) of the primitive susceptance matrix (per unit value)
    17. Bac, element (1,3) of the primitive susceptance matrix (per unit value)
    18. Bbb, element (2,2) of the primitive susceptance matrix (per unit value)
    19. Bbc, element (2,3) of the primitive susceptance matrix (per unit value)
    20. Bcc, element (3,3) of the primitive susceptance matrix (per unit value)
    21. Status, 0 is OFF, 1 is ON

- **Transformer-branch data matrix.** Each row of this matrix contains data for one branch. The columns determine the properties of the transformer branch and are as follows:

    1. First bus end of the transformer
    2. Second bus end of the transformer
    3. Short-circuit resistance (per unit value)
    4. Short-circuit reactance (per unit value)
    5. Transformer connection (Integer number)
        = 1: YNyn0
        = 2: YNy0
        = 3: Yy0
        = 4: Dd0
        = 5: YNd1
        = 6: Yd1
        = 7: Dyn1
        = 8: Dy1
        = 9: Dd2
        = 10: Dd4
        = 11: YNd5
        = 12: Yd5
        = 13: Dyn5
        = 14: Dy5
        = 15: YNd6
        = 16: Yd6
        = 17: Dd6
        = 18: YNd7
        = 19: Yd7
        = 20: Dyn7
        = 21: Dy7
        = 22: Dd8
        = 23: Dd10
        = 24: YNd11
        = 25: Yd11
        = 26: Dyn11
        = 27: Dy11
        Otherwise: YNyn0
    6. Tap variation of the primary winding
    7. Tap variation of the secondary winding
    8. Status, 0 is OFF, 1 is ON

The transformer connection symbols used are according to the standard IEC 60076-1-2011: the star or delta winding connection are indicated by the capital letters Y or D for the high-voltage (HV) winding and small letters y or d for the low-voltage (LV) windings. If the neutral point of a star-connected winding is brought out, the indication is YN (yn). The winding connection letter for the low-voltage winding is immediately followed by its phase displacement 'clock number'.

5.2.2. Program characteristics

The MATLAB implementation has the following characteristics:

- **Data input script:** The input file is a script that generates a data structure containing all the information of the case, what makes the data lecture very simple.

- **Complex values:** MATLAB has built-in management of complex variables; therefore, all the equations can be established directly using their complex equivalents. As an example, instead of solving equations for the active and reactive power mismatch, the complex-valued apparent power mismatch equation is solved.

- **Matrix and vector approach:** Scripts and functions written in MATLAB perform better when using a matrix approach to compute solutions. Consequently, the program has been written to use matrices and vectors as inputs to all the internal functions and calculations and as much as possible, the use of for loops have been avoided.

- **Use of sparse matrices:** There are three large matrices that are part of the program: nodal admittance matrix, primitive admittance matrix and Jacobian matrix.

  The nodal admittance matrix will have as many nonzero elements as connections exist between nodes of the network, however as in any graph this is usually a very small number in comparison with the number of possible connections. It makes the use of sparse structure for the nodal admittance matrix a great tool that saves storage space and speeds the convergence time up.

  The Jacobian matrix of the system will have a sparsity pattern strongly dependent of the nodal admittance matrix. For this reason, all calculations used to compute the Jacobian matrix of the system are preserving all matrices as sparse.

  Finally, the primitive admittance matrix is inherently sparse having at most three nonzero elements per row for lines and six for power transformers.

- **Use of MATLAB nonlinear solver fsolve:** The MATPOWER power flow calculator use iterative solvers as Newton-Raphson and Gauss-Seidel [16]. However, MATLAB already has a solver for nonlinear system of equations called fsolve [19], which will be used in the current development.

  It uses different algorithms to deal with large scale problems. Among others some of these algorithms are: trust-region, trust-region dogleg, Levenberg-Marquardt.

  All these algorithms use the Jacobian matrix of the system into expressions more complex than the commonly used Newton-Raphson method. By default, fsolve chooses the trust-region dogleg algorithm. This function also can receive a user defined analytic Jacobian that can improve the performance of the algorithm.

5.2.3. Implementation sequence

The program consists of the following steps:

1. **Internal indexing and classification of buses:** Each bus has an identifier assigned in the input file; however internally new identifiers are assigned to match the index of each voltage over the voltage vector of the system. Then the indexes are classified according to bus type.

2. **Initial conditions:** The initial values of voltage magnitude and angle are taken from the input files. Then the system equations are solved having as unknowns the voltage magnitude and angle of the PQ buses and the voltage angle of the PV buses.

3. **Construction of the power injection vector:** The power injection vector is formed using equations 3.8, 3.9 and 3.10. However, the power consumed by the constant impedance loads is not added here.

4. **Construction of the admittance matrices:** The input data provides the impedance matrices per line and the short-circuit impedance and connection of the transformers. Their respective nodal admittance matrix is computed according to section 2.

   The impedance-constant delta- and wye-load values provided in the input file are then converted to admittance values according to section 3.5.

   Finally, all the admittance data is stored in three column vectors, the first and second columns contain the correspondent coordinates over the global nodal admittance matrix and the third column contains the specific admittance values. These three vectors are then used to assemble a sparse matrix to store the information in a more efficient way.

5. **Solving of the complex power mismatch equation and final calculations:** A function to compute the complex power mismatch is written and provided to the nonlinear solver of MATLAB, *fsolve*. Additionally, a function that evaluates the analytic derivatives that are part of the Jacobian matrix is provided to the function to increase its performance.

   This Jacobian function is based on the function from MATPOWER, but the derivative values correspondent to the constant-current loads (they are function of the voltage magnitude) and the derivatives correspondent to the delta loads (the power injected to each end node of each branch of the delta is function of both voltages) are added to the output matrix.

5.3. C / PETSc Implementation

The code has been implemented in C code using the PETSc libraries. There is already an example of use of PETSc on balanced power systems called PFLOW [2] [3] [4], what makes use of the DMNetwork interface and SNES libraries for solving the nonlinear electric power grid problem.

The present implementation is based on PFLOW, but it has been modified to focus on unbalanced distribution networks, according to the models developed in section 2.

5.3.1. Input information

It uses as data input, text files with the format shown in 5.2.1 and used by the MATLAB implementation. They can be MATLAB .m files or text files with the same format.

Regarding the SNES method, KSP method and PC type selected for running the program, they are indicated in a file called ‘pfoptions’, that is included in the same directory of the C / PETSc implementation.

5.3.2. Program details

The C / PETSc implementation has the following characteristics:

- **Data input and DPFLOW data structures:** In this case, the program reads the information from a text file, line by line, and then store it in a power network data structure called ‘*pfdata’.* This structure is defined in the header file (.h file) and contains other sub-

structures to represent the main components of the electrical network: generators, wye-loads, delta-load, nodes or vertexes and branches or edges.

- **Complex values management:** Even though PETSc can be configured with complex numbers, the performance of the programs is usually better with real numbers. Because of this, all the functions of the program have been written using real numbers, i.e. using real and imaginary parts of the variables or their magnitudes and angles.

- **Algorithm approach:** PETSc has matrix objects and they can be set using different patterns. However, it is usually better to let PETSc handle the assemble of matrices. In this case the only matrix manually assemble is the Jacobian matrix, however the main algorithms make use of the DMNetwork object and different loops, instead of the matrix approach used in MATLAB.

- **DMNetwork object use:** DM are abstract PETSc objects that manage an abstract grid object and its interactions with the algebraic solvers. DMNetwork is a subclass of DM that provides abstractions for representing general unstructured networks such as communication networks, power grid, computer networks, transportation networks, electrical circuits, graphs, and others.

  DMNetwork uses edges and nodes of the network to form a graph. It allows to store various void pointers on each node and edge. Void or generic pointers are special pointers that can be pointed at objects of any data type. In this way, it is possible to add void pointers to the vertex, wye-load and delta-load and generator structures on each vertex and void pointers to the branch objects on each edge.

  It is always possible to access to the information on neighbor vertices using the DMNetwork algorithms provided by PETSc. This is used to calculate the power injections from each branch connected to each node in the computation of the objective function, without using the nodal admittance matrix. Similar approach is then used for the function to compute the Jacobian matrix of the system.

  In the balance case, each distribution branch represents one edge and each bus represents one node, but in the unbalanced format, they represent 3 nodes and up to 15 edges. This bigger number of edges represented by each branch of the electrical network responds to the fact that the nodal admittance matrix of a three-phase line or transformer branch with both end nodes is accurately represented by an admittance matrix of 6x6 elements and the maximum number of possible combinations over 6 elements is 15. Three nodes are represented by each bus because all three phases will need now to be consider.

  The problem is handled by creating an equivalent balanced network that gets the same admittance matrix of the unbalanced case. In this network, each phase of each node is considered a node itself and each element of the admittance matrix is represented by a theoretical branch (even those that represent magnetic links).

5.3.3. Program sequence

1. **Program initialization:** All required variables are declared in the main function of the code. The file to be used as input is assigned as a char array to pfdata_file. PETSc is set up with the options from the file “pfoptions”.

2. **Data extraction:** Data is taken from the input file and assigned to the correspondent elements the electrical network data structure. As mention before, each bus of the network corresponds to three nodes of the graph and from the input information of each line and transformer, fifteen equivalent branches are created. The values are so that if we calculated the nodal admittance network of this new network, the same nodal admittance matrix for the original unbalanced case would be obtained. The constant impedance wye-loads add three more branches and the constant impedance delta-loads add nine more branches to the network, with their admittances being obtained according to section 3.5.

3. **DMNetwork creation:** An empty DMNetwork is created and its various components are registered: branches, buses, generators, wye-loads and delta-loads. Then, the DMNetwork fields are filled using the electrical network data structure.

4. **SNES solver:** Finally, the nonlinear solver object, SNES, is created. The DMNetwork object is passed to the solver. A function to form the objective function of the problem and other function to form the proper Jacobian matrix of the system per iteration are provided to the SNES solver. The SNES solver is set according to the options provided in the 'pfoptions' file and the problem is solved for the voltage magnitudes and angles.

5. **Calculation of other electrical variables:** After obtaining the voltage of the nodes, the currents of all the branches are determined by using the DMNetwork. This structure allows to go through all the graph calculating the voltage difference between each end nodes for then multiply this value times the admittance of the branch. The current per branch times the voltage nodes are equal to the power coming to and leaving the branch, which finally allows to calculate the losses on the branch and the power injection per node.

   As mentioned before, in the C / PETSc implementation, no matrix is used to compute these values, but the quantities per element are computed by going through all the network in a loop guided by the DMNetwork object.

6 Program Testing

6.1. Testing approach

The performance of the program, in MATLAB and PETSc is initially measured as function of two characteristics:

- Validity of the solution obtained
- Capacity to perform over large networks

6.1.1. Solution validity test

To test the validity of the solution, two distribution test feeders from the IEEE Power and Energy Society, Distribution System Analysis Subcommittee, have been selected:

- 4-bus Test Feeder: This is a very small case, especially useful because it allows to test the capability of the program to handle transformers with different connections [13]. The results of this test are presented in section 6.2.

- 13-bus Test Feeder: This case is also small, but it contains many important features that need to be tested as: constant power, constant current and constant impedance unbalanced loads, overhead and underground lines, shunt capacitors and an in-line transformer [14]. The results of this test are presented in section 6.3.

6.1.2. Performance of the program over large-scale networks

An electrical network with 452745 nodes has been constructed by creating a slightly meshed transmission network at 69 and 33 kV connecting, by means of transformers, distribution feeders at 11 kV and 416 V terminal nodes. These low voltage distribution feeders are based on the layout and power demand scales of the IEEE European Low Voltage Test Feeder from the IEEE Power and Energy Society [20].

The performance of the program over the case is measured when selecting different nonlinear and linear solvers.

6.2. IEEE distribution 4-node test feeder

This feeder consists of 4 nodes, one three-phase generator feeding a load through two distribution lines and one power transform. The case has input data for different transformer connections, voltage values, balanced and unbalanced loads and transmission line configurations [13]. Figure 12 shows the schematic diagram of the network.

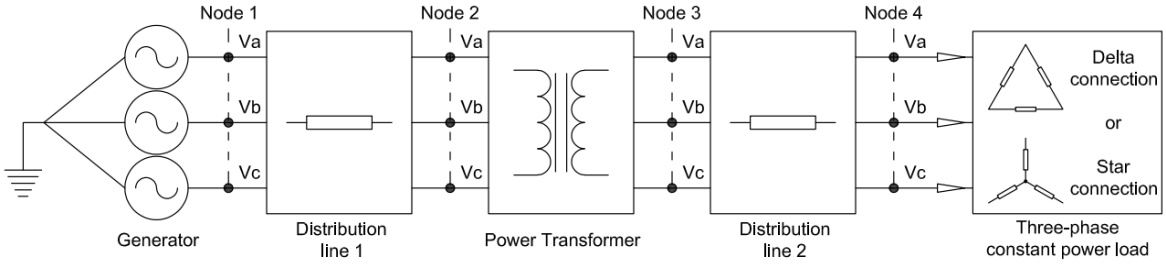


Figure 12. *Electrical diagram for the IEEE 4-node test feeder*

The input data of the case is shown in [13]. The problem has been solved for three step-down transformer connections with unbalanced load:

- Grounded star - Grounded star transformer connection with the load connected in Star
- Grounded star - Delta transformer connection with the load connected in Delta
- Delta - Grounded star transformer connection with the load connected in Star

The impedance matrices per unit of length, when carrying four or three cables respectively, are shown below:

Zy [Ω/mile]			**Zd [Ω/mile]**		
0.4576 + j1.078	0.1559 + j0.5017	0.1535 + j0.3849	0.4013 + j1.4133	0.0953 + j0.8515	0.0953 + j0.7266
0.1559 + j0.5017	0.4666 + j1.0482	0.158 + j0.4236	0.0953 + j0.8515	0.4013 + j1.4133	0.0953 + j0.7802
0.1535 + j0.3849	0.158 + j0.4236	0.4615 + j1.0651	0.0953 + j0.7266	0.0953 + j0.7802	0.4013 + j1.4133

Table 2. *Primitive impedance matrices for the IEEE 4-node test feeder*

The data for the step-down transformer is shown below:

Nominal power [kVA]	**Nominal high voltage [kV_{LL}]**	**Nominal low voltage [kV_{LL}]**	**Short-circuit resistance [%]**	**Short-circuit reactance [%]**
6,000	12.47	4.16	1.0	6.0

Table 3. *Transformer data for the IEEE 4-node test feeder*

And finally, the data corresponding to the load is as follows:

Phase	**Active power [kW]**	**Power factor**
A	1275	0.85 lag
B	1800	0.9 lag
C	2375	0.95 lag

Table 4. *Load data for the IEEE 4-node test feeder*

The settings of the MATLAB implementation for this case are as follows:

- Absolute tolerance: 1e-10

The settings of the PETSc implementation for this case are as follows:

- Nonlinear solver: Newton's method with linear search
- Absolute tolerance: 1e-10
- Relative tolerance 1e-20
- KSP type: GMRES
- Preconditioner type: LU
- Preconditioner matrix reordering: AMD

The results obtained by both implementations of the program, for these three transformer connections are shown in tables 8, 9 and 10:

STEP-DOWN TRANSFORMER WITH GROUNDED STAR – GROUNDED STAR CONNECTION					
Network variables	**Solution in [13]**	**MATLAB Implementation**		**PETSc implementation**	
		Solution	**Difference [%]**	**Solution**	**Difference [%]**
Number of iterations	-	6		6	
Execution time [seconds]	-	0.942000		0.002857	
Node 2					
Va – magnitude [volts]	7164	7163.70	0.0	7163.70	0.0
Va – angle [degrees]	-0.1	-0.14	0.0	-0.14	0.0
Vb – magnitude [volts]	7110	7110.49	0.0	7110.49	0.0
Vb – angle [degrees]	-120.2	-120.18	0.0	-120.18	0.0
Vc – magnitude [volts]	7082	7082.00	0.0	7082.00	0.0
Vc – angle [degrees]	119.3	119.26	0.0	119.26	0.0
Node 3					
Va – magnitude [volts]	2305	2305.48	0.0	2305.48	0.0
Va – angle [degrees]	-2.3	-2.26	0.0	-2.26	0.0
Vb – magnitude [volts]	2255	2254.66	0.0	2254.66	0.0
Vb – angle [degrees]	-123.6	-123.62	0.0	-123.62	0.0
Vc – magnitude [volts]	2203	2202.79	0.0	2202.79	0.0
Vc – angle [degrees]	114.8	114.79	0.0	114.79	0.0
Node 4					
Va – magnitude [volts]	2175	2174.90	0.0	2174.90	0.0
Va – angle [degrees]	-4.1	-4.12	0.0	-4.12	0.0
Vb – magnitude [volts]	1930	1929.86	0.0	1929.86	0.0
Vb – angle [degrees]	-126.8	-126.80	0.0	-126.80	0.0
Vc – magnitude [volts]	1833	1832.57	0.0	1832.57	0.0
Vc – angle [degrees]	102.8	102.84	0.0	102.84	0.0
Branch 1-2					
Ia – magnitude [volts]	230.1	230.08	0.0	230.08	0.0
Ia – angle [degrees]	-35.9	-35.91	0.0	-35.91	0.0
Ib – magnitude [volts]	345.7	345.73	0.0	345.73	0.0
Ib – angle [degrees]	-152.6	-152.64	0.0	-152.64	0.0
Ic – magnitude [volts]	455.1	455.10	0.0	455.10	0.0
Ic – angle [degrees]	84.7	84.65	0.0	84.65	0.0
Branch 3-4					
Ia – magnitude [volts]	689.7	689.69	0.0	689.69	0.0
Ia – angle [degrees]	-35.9	-35.91	0.0	-35.91	0.0
Ib – magnitude [volts]	1036	1036.35	0.0	1036.35	0.0
Ib – angle [degrees]	-152.6	-152.64	0.0	-152.64	0.0
Ic – magnitude [volts]	1364	1364.20	0.0	1364.20	0.0
Ic – angle [degrees]	84.7	84.65	0.0	84.65	0.0

Table 5. *Results comparison of MATLAB and PETSc implementation, grounded star – grounded star transformer connection*

STEP-DOWN TRANSFORMER WITH GROUNDED STAR – DELTA CONNECTION					
Network variables	**Solution in [13]**	**MATLAB Implementation**		**PETSc implementation**	
		Solution	**Difference [%]**	**Solution**	**Difference [%]**
Number of iterations	-	7		5	
Execution time [seconds]	-	0.969000		0.003118	
Node 2					
Va – magnitude [volts]	7113	7112.52	0.0	7112.52	0.0
Va – angle [degrees]	-0.2	-0.21	0.0	-0.21	0.0
Vb – magnitude [volts]	7144	7143.28	0.01	7143.28	0.01
Vb – angle [degrees]	-120.4	-120.42	0.0	-120.42	0.0
Vc – magnitude [volts]	7111	7110.08	0.01	7110.08	0.01
Vc – angle [degrees]	119.5	119.53	0.0	119.53	0.0
Node 3					
Vab – magnitude [volts]	3896	3896.28	0.0	3896.28	0.0
Vab – angle [degrees]	-2.8	-2.82	0.0	-2.82	0.0
Vbc – magnitude [volts]	3972	3972.08	0.0	3972.08	0.0
Vbc – angle [degrees]	-123.8	-123.83	0.0	-123.83	0.0
Vca – magnitude [volts]	3875	3875.04	0.0	3875.04	0.0
Vca – angle [degrees]	115.7	115.70	0.0	115.70	0.0
Node 4					
Vab – magnitude [volts]	3425	3425.38	0.0	3425.38	0.0
Vab – angle [degrees]	-5.8	-5.76	0.0	-5.76	0.0
Vbc – magnitude [volts]	3646	3646.25	0.0	3646.25	0.0
Vbc – angle [degrees]	-130.3	-130.28	0.0	-130.28	0.0
Vca – magnitude [volts]	3298	3297.61	0.0	3297.61	0.0
Vca – angle [degrees]	108.6	108.58	0.0	108.58	0.0
Branch 1-2					
Ia – magnitude [volts]	308.5	308.47	0.0	308.47	0.0
Ia – angle [degrees]	-41.5	-41.46	0.0	-41.46	0.0
Ib – magnitude [volts]	314.6	314.67	0.03	314.67	0.03
Ib – angle [degrees]	-145.5	-145.48	0.0	-145.48	0.0
Ic – magnitude [volts]	389	388.99	0.0	388.99	0.0
Ic – angle [degrees]	85.9	85.93	0.0	85.93	0.0
Branch 3-4					
Ia – magnitude [volts]	1083.8	1083.86	0.009	1083.86	0.009
Ia – angle [degrees]	-71	-71.03	0.0	-71.03	0.0
Ib – magnitude [volts]	849.9	849.92	0.0	849.92	0.0
Ib – angle [degrees]	177	176.98	0.0	176.98	0.0
Ic – magnitude [volts]	1098.7	1098.70	0.0	1098.70	0.0
Ic – angle [degrees]	63.1	63.14	0.0	63.14	0.0

Table 6. *Results comparison of MATLAB and PETSc implementation, grounded star – delta transformer connection*

STEP-DOWN TRANSFORMER WITH DELTA - GROUNDED STAR CONNECTION					
Network variables	**Solution in [13]**	**MATLAB Implementation**		**PETSc implementation**	
		Solution	**Difference [%]**	**Solution**	**Difference [%]**
Number of iterations	-	6		5	
Execution time [seconds]	-	0.951000		0.004108	
Node 2					
Vab – magnitude [volts]	12350	12350.21	0.0	12350.21	0.0
Vab – angle [degrees]	29.6	29.60	0.0	29.60	0.0
Vbc – magnitude [volts]	12314	12313.80	0.0	12313.80	0.0
Vbc – angle [degrees]	-90.4	-90.39	0.0	-90.39	0.0
Vca – magnitude [volts]	12333	12332.66	0.0	12332.66	0.0
Vca – angle [degrees]	149.8	149.75	0.0	149.75	0.0
Node 3					
Va – magnitude [volts]	2290	2290.28	0.0	2290.28	0.0
Va – angle [degrees]	-32.4	-32.40	0.0	-32.40	0.0
Vb – magnitude [volts]	2261	2261.59	0.04	2261.59	0.04
Vb – angle [degrees]	-153.8	-153.81	0.0	-153.81	0.0
Vc – magnitude [volts]	2214	2213.94	0.0	2213.94	0.0
Vc – angle [degrees]	85.2	85.18	0.0	85.18	0.0
Node 4					
Va – magnitude [volts]	2157	2156.78	0.0	2156.78	0.0
Va – angle [degrees]	-34.2	-34.24	0.0	-34.24	0.0
Vb – magnitude [volts]	1936	1936.17	0.0	1936.17	0.0
Vb – angle [degrees]	-157	-157.04	0.0	-157.04	0.0
Vc – magnitude [volts]	1849	1849.34	0.0	1849.34	0.0
Vc – angle [degrees]	73.4	73.39	0.0	73.39	0.0
Branch 1-2					
Ia – magnitude [volts]	285.7	285.65	0.0	285.65	0.0
Ia – angle [degrees]	-27.6	-27.61	0.0	-27.61	0.0
Ib – magnitude [volts]	402.7	402.69	0.0	402.69	0.0
Ib – angle [degrees]	-149.6	-149.59	0.0	-149.59	0.0
Ic – magnitude [volts]	349.1	349.15	0.0	349.15	0.0
Ic – angle [degrees]	74.4	74.35	0.0	74.35	0.0
Branch 3-4					
Ia – magnitude [volts]	695.5	695.49	0.0	695.49	0.0
Ia – angle [degrees]	-66	-66.03	0.0	-66.03	0.0
Ib – magnitude [volts]	1033	1032.97	0.0	1032.97	0.0
Ib – angle [degrees]	177.1	177.12	0.0	177.12	0.0
Ic – magnitude [volts]	1352	1351.83	0.0	1351.83	0.0
Ic – angle [degrees]	55.2	55.20	0.0	55.20	0.0

Table 7. *Results comparison of MATLAB and PETSc implementation, delta – grounded star transformer connection*

The relative difference has been computed respect to the solution shown in [13]. It is computed after rounding the solution value to the same number of significant digits of the reference solution. We can observe how it is always less than 0.1%, difference due to rounding errors introduced when computing the per-unit impedance matrices of the lines. Therefore, it is concluded that the transformer modelling has been properly performed.

Also, it is important to remark that in this case the values obtained by both implementations are basically the same. This has occurred in all the study cases, therefore for all the remaining tests, only the values for one implementation have been depicted.

The input files for these three transformer connections are included in both repositories for the MATLAB and PETSc implementation by the names of: ‘distCase_4YNyn0.m’, ‘distCase_4YNd1.m’ and ‘distCase_4Dyn1.m’.

6.3. IEEE distribution 13-node test feeder

This distribution feeder includes a voltage regulator for the voltage on node RG60, so this node is treated as the slack bus of the system. It includes spot and distributed loads, with different connections and voltage dependence, shunt capacitors, a switch and asymmetrical distribution lines and underground cables [14]. The single-line diagram of this feeder is depicted at figure 13:

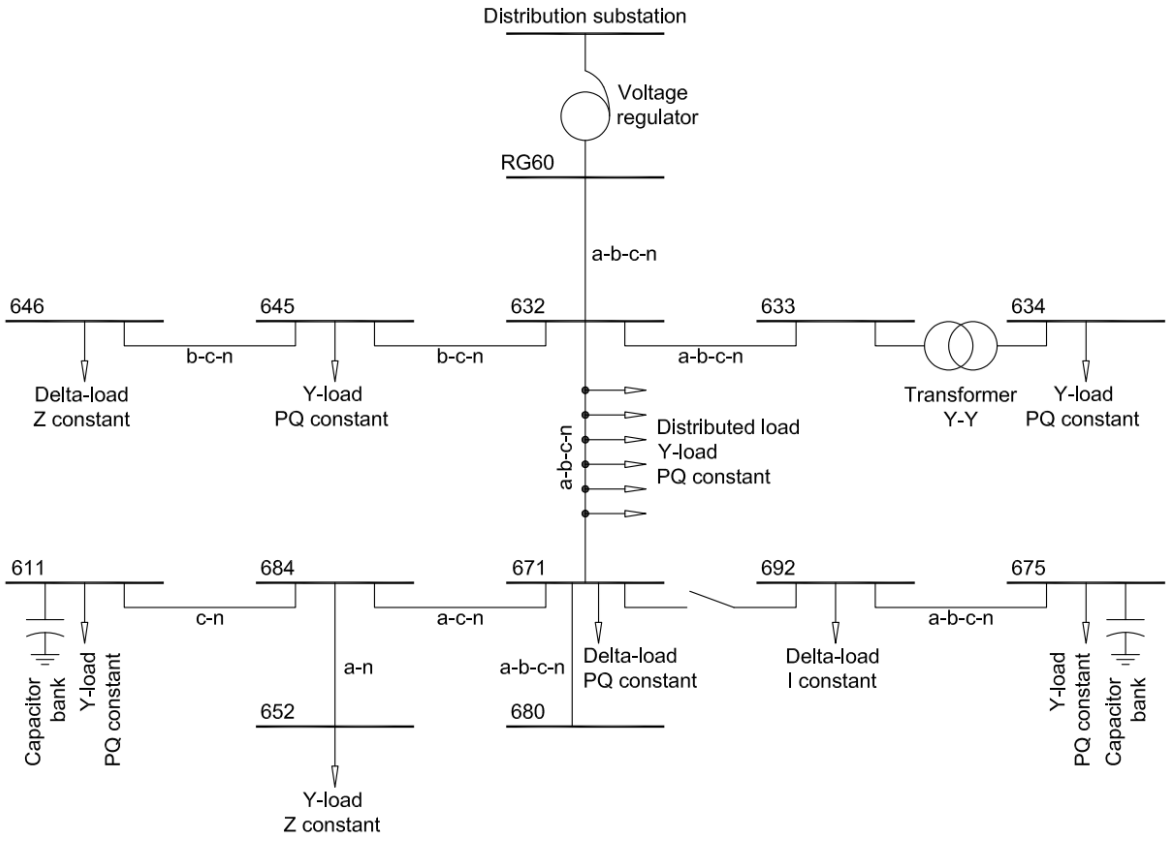


Figure 13. *Single line diagram of the 13-nodes distribution test feeder*

The data for line and cable branches is given in the following tables:

Line/Cable	1	2	3	4	5	6	7	8	9	10
Origin node	632	632	645	RG60	684	632	671	671	684	692
End node	645	633	646	632	652	671	684	680	611	675
Length [ft]	500	500	300	2000	800	2000	300	1000	300	500
Configuration	603	602	603	601	607	601	604	601	605	606

Table 8. *Line and cable data for the IEEE 13-node test feeder*

The impedance matrices corresponding to each configuration are detail in [14].

The data for the step-down YNyn0 transformer is shown below:

Nominal power [kVA]	Nominal high voltage [kV_{LL}]	Nominal low voltage [kV_{LL}]	Short-circuit resistance [%]	Short-circuit reactance [%]
500	4.16	0.48	1.1	2

Table 9. *Transformer data for the IEEE 13-node test feeder*

And finally, the data corresponding to capacitor banks and loads is as follows:

Node	Reactive power Phase A [kVAR]	Reactive power Phase B [kVAR]	Reactive power Phase C [kVAR]
675	200	200	200
611			100

Table 10. *Capacitor data for the IEEE 13-node test feeder*

Node	Load Model	Active power [kW] Phase A/AB	Reactive power [kVAR] Phase A/AB	Active power [kW] Phase B/BC	Reactive power [kVAR] Phase B/BC	Active power [kW] Phase C/CA	Reactive power [kVAR] Phase C/CA
634	Y-PQ	160	110	120	90	120	90
645	Y-PQ	0	0	170	125	0	0
646	D-Z	0	0	230	132	0	0
652	Y-Z	128	86	0	0	0	0
671	D-PQ	385	220	385	220	385	220
675	Y-PQ	485	190	68	60	290	212
692	D-I	0	0	0	0	170	151
611	Y-I	0	0	0	0	170	80

Table 11. *Spot load data for the IEEE 13-node test feeder*

Origin node	End node	Load Model	Active power [kW] Phase A	Reactive power [kVAR] Phase A	Active power [kW] Phase B	Reactive power [kVAR] Phase B	Active power [kW] Phase C	Reactive power [kVAR] Phase C
632	671	Y-PQ	160	110	120	90	120	90

Table 12. *Distributed load data for the IEEE 13-node test feeder*

It is important to remark that the distributed load has been simulated as 10 spotted loads distributed along the line connecting nodes 632 and 671.

The following table shows the results obtained with the software and the results in the publication of the test feeder, as well as the relative difference between them:

BUS VOLTAGE MAGNITUDE									
Bus ID	Phase A			Phase B			Phase C		
	Value in [14]	Solution [p.u.]	Difference [%]	Value in [14]	Solution [p.u.]	Difference [%]	Value in [14]	Solution [p.u.]	Difference [%]
RG60	1.0625	1.0625	0.00	1.0500	1.0500	0.00	1.0687	1.0687	0.00
632	1.0210	1.0210	0.00	1.0420	1.0420	0.00	1.0174	1.0175	0.00
633	1.0180	1.0180	0.00	1.0401	1.0401	0.00	1.0148	1.0148	0.00
634	0.9940	0.9940	0.00	1.0218	1.0218	0.00	0.9960	0.9960	0.00
645	-	-	-	1.0329	1.0328	0.01	1.0155	1.0155	0.00
646	-	-	-	1.0311	1.0311	0.00	1.0134	1.0134	0.00
671	0.9900	0.9900	0.00	1.0529	1.0529	0.00	0.9778	0.9778	0.00
680	0.9900	0.9900	0.00	1.0529	1.0529	0.00	0.9778	0.9778	0.00
684	0.9881	0.9881	0.00	-	-	-	0.9758	0.9758	0.00
611	-	-	-	-	-	-	0.9738	0.9738	0.00
652	0.9825	0.9825	0.00	-	-	-	-	-	-
692	0.9900	0.9900	0.00	1.0529	1.0529	0.00	0.9777	0.9778	0.01
675	0.9835	0.9835	0.00	1.0553	1.0553	0.00	0.9758	0.9759	0.01
BUS VOLTAGE ANGLE									
Bus ID	Phase A			Phase B			Phase C		
	Value in [14]	Solution [degrees]	Difference [%]	Value in [14]	Solution [degrees]	Difference [%]	Value in [14]	Solution [degrees]	Difference [%]
RG60	0.00	0.00	0.00	-120.00	-120.00	0.00	120.00	120.00	0.00
632	-2.49	-2.49	0.00	-121.72	-121.72	0.00	117.83	117.83	0.00
633	-2.56	-2.55	0.00	-121.77	-121.77	0.00	117.82	117.82	0.00
634	-3.23	-3.23	0.00	-122.22	-122.22	0.00	117.34	117.35	0.01
645	-	-	-	-121.90	-121.90	0.00	117.86	117.86	0.00
646	-	-	-	-121.98	-121.98	0.00	117.90	117.90	0.00
671	-5.30	-5.30	0.00	-122.34	-122.34	0.00	116.02	116.03	0.01
680	-5.30	-5.30	0.00	-122.34	-122.34	0.00	116.02	116.03	0.01
684	-5.32	-5.32	0.00	-	-	-	115.92	115.92	0.00
611	-	-	-	-	-	-	115.78	115.78	0.00
652	-5.25	-5.24	0.19	-	-	-	-	-	-
692	-5.31	-5.30	0.19	-122.34	-122.34	0.00	116.02	116.02	0.00
675	-5.56	-5.55	0.18	-122.52	-122.52	0.00	116.03	116.04	0.01

Table 13. *Comparison of bus voltage results from MATLAB/PETSc implementation and 13-node test feeder publication*

BRANCH CURRENT MAGNITUDE										
From	To	Phase A			Phase B			Phase C		
		Value in [14]	Solution [Amps]	Diff. [%]	Value in [14]	Solution [Amps]	Diff. [%]	Value in [14]	Solution [Amps]	Diff. [%]
RG60	632	558.40	558.38	0.004	414.87	414.85	0.005	586.60	586.53	0.011
632	645	-	-	-	143.02	143.02	0.000	65.21	65.21	0.000
632	633	81.33	81.33	0.000	61.12	61.12	0.000	62.70	62.70	0.000
645	646	-	-	-	65.21	65.21	0.000	65.21	65.21	0.000
632	671	478.29	478.27	0.004	215.12	215.10	0.009	475.50	475.45	0.011
684	652	63.07	63.07	0.000	-	-	-	-	-	-
671	684	63.07	63.07	0.000	-	-	-	71.15	71.15	0.000
671	680	0.00	0.00	0.000	0.00	0.00	0.000	0.00	0.00	0.000
684	611	-	-	-	-	-	-	71.15	71.15	0.000
692	675	205.33	205.33	0.000	69.61	69.60	0.001	124.07	124.06	0.008
HV side transf		81.33	81.33	0.000	61.12	61.12	0.000	62.71	62.70	0.016
LV side transf		704.83	704.83	0.000	529.73	529.74	0.002	543.45	543.44	0.002
BRANCH CURRENT ANGLE										
From	To	Phase A			Phase B			Phase C		
		Value in [14]	Solution [degs]	Diff. [%]	Value in [14]	Solution [degs]	Diff. [%]	Value in [14]	Solution [degs]	Diff. [%]
RG60	632	-28.58	-28.57	0.035	-140.91	-140.91	0.000	93.59	93.60	0.011
632	645	-	-	-	-142.66	-142.66	0.000	57.83	57.83	0.000
632	633	-37.74	-37.74	0.000	-159.09	-159.09	0.000	80.48	80.48	0.000
645	646	-	-	-	-122.17	-122.17	0.000	57.83	57.83	0.000
632	671	-27.03	-27.02	0.037	-134.66	-134.65	0.007	99.90	99.92	0.020
684	652	-39.12	-39.13	0.026	-	-	-	-	-	-
671	684	-39.12	-39.12	0.000	-	-	-	121.62	121.62	0.000
671	680	0.00	0.00	0.000	0.00	0.00	0.000	0.00	0.00	0.000
684	611	-	-	-	-	-	-	121.61	121.62	0.008
692	675	-5.15	-5.14		-55.19	-55.19	0.000	111.79	111.80	0.009
HV side transf		-37.74	-37.74	0.000	-159.09	-159.09	0.000	80.47	80.48	0.012
LV side transf		-37.74	-37.74	0.000	-159.09	-159.09	0.000	80.47	80.48	0.012

Table 14. *Comparison of branch current results from MATLAB/PETSc implementation and 13-node test feeder publication*

It is observed how in most cases the values are exactly the same considering the decimal numbers provided and the greatest difference is the 0.2%, caused by modelling differences for the switch impedance and the distributed load characteristics. However, the precision of the results is very good and we consider the modelling have been well performed and included into the solver.

The settings of the MATLAB implementation for this case are as follows:

- Absolute tolerance: 1e-10

The settings of the PETSc implementation for this case are as follows:

- Nonlinear solver: Newton's method with linear search
- Absolute tolerance: 1e-10
- Relative tolerance 1e-20
- KSP type: GMRES
- Preconditioner type: LU
- Preconditioner matrix reordering: AMD

Regarding the convergence of the solvers, table 16 summarizes the different execution times of the programs and the number iterations for each implementation:

Implementation	Number of iterations	Data preparation time [s]	Nonlinear solver convergence time [s]
MATLAB	5	0.112000	0.938000
C / PETSc	5	0.00071550	0.0040359

Table 15. *Iterations and execution times MATLAB and C / PETSc implementations when solving the 13-node test feeder*

The following figure will depict the residual norm of the objective function vs the iteration counter on the MATLAB implementation:

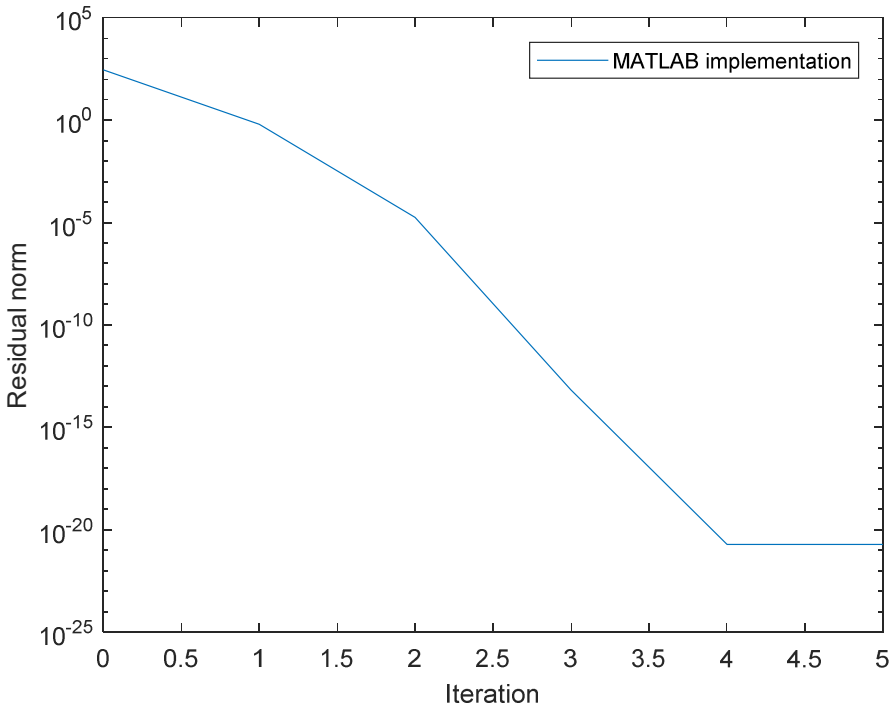


Figure 14. *Residual norm vs iteration number for MATLAB implementation when solving the 13-node test feeder*

The following figure will depict the residual norm of the objective function vs the iteration counter on the C / PETSc implementation:

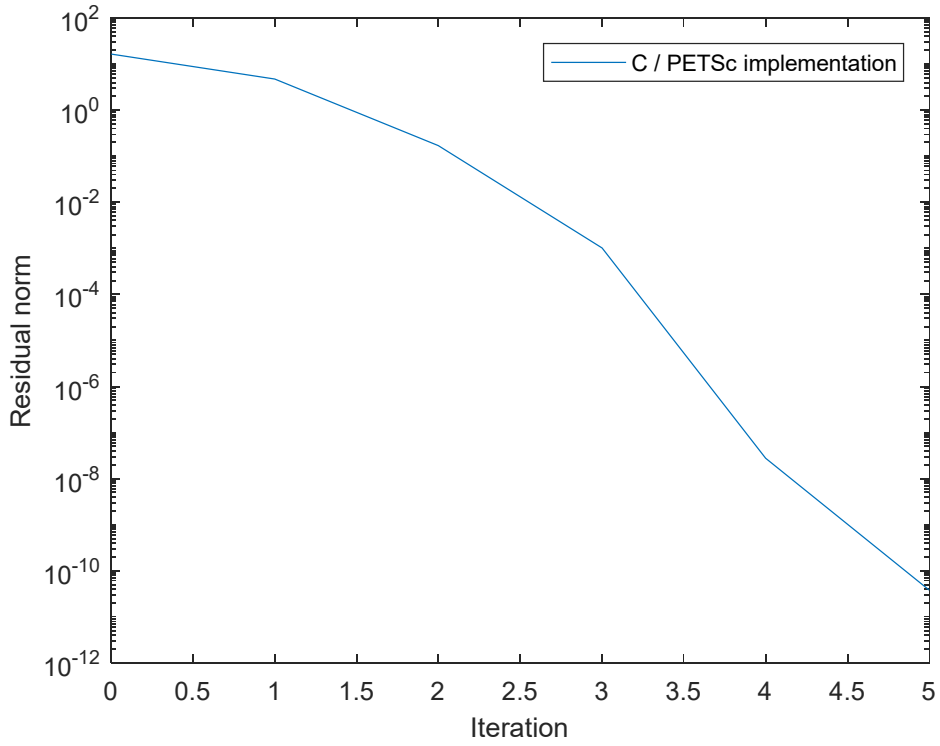


Figure 15. *Residual norm vs iteration number for C / PETSc implementation when solving the 13-node test feeder*

The tolerance for both programs was set to $1x10^{-10}$. However, it is worth pointing out that the MATLAB implementation uses the function *fsolve* that evaluates the norm of the vector of first-order optimal values, that are defined differently for each of its sub-algorithms. Instead, the PETSc implementation measures the norm 2 of the objective function to compare against the tolerance given to the program.

The input file for this case is included in the repositories for the MATLAB and PETSc implementation by the name of: 'distCase_13nodes.m'.

6.4. Performance analysis of the application of the program to a case with 452745 nodes

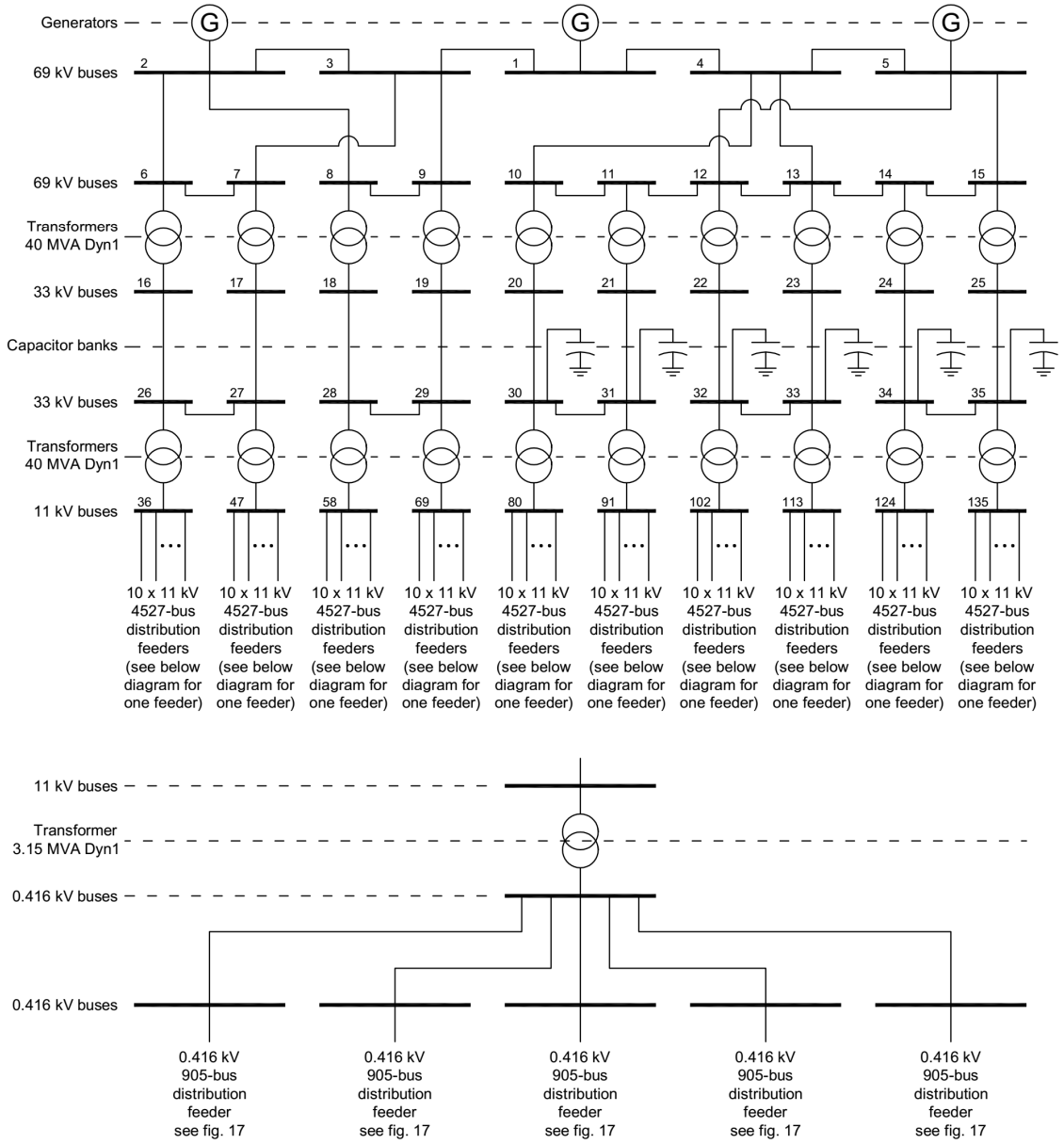


Figure 16. *One-line diagram of the 452745-bus electrical network*

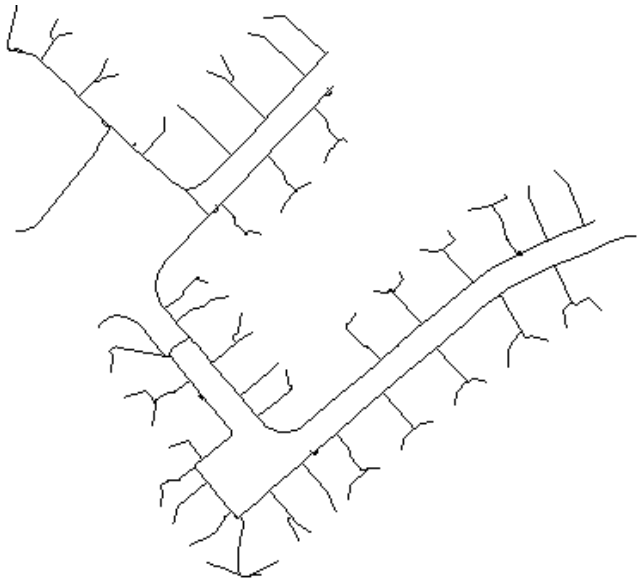


Figure 17. *One-line diagram of the 905-bus distribution feeder (based on European low voltage test feeder [20])*

We see can above, the single-line diagram for the test network that will be used for this performance test. It consists downstream of a 905-bus distribution feeder whose layout is based on the IEEE European low voltage test feeder [20], without neither the power transformer nor the generator. Five of these feeders are connected to a power transformer that takes the voltage level from 416 V to 11 kV. One hundred similar feeders are generated, and then the loads of the network, initially based on the European test feeder, are randomly re-distributed along the phases as single-phase, two-phase or three-phase loads.

These one hundred feeders are then connected to the transmission network shown in figure 16. The nodes sum up to a total number of 452745 nodes.

The input file for this case is also included in the repositories for the MATLAB and PETSc implementation by the name of: 'distCase_452745nodes.m'. The characteristics of the main elements of this network are resumed in the following tables:

Transformer Connection	Nominal power [MVA]	Nominal high voltage [kV_{LL}]	Nominal low voltage [kV_{LL}]	Short-circuit resistance [%]	Short-circuit reactance [%]	Secondary winding tap [%]
Dyn1	40	69	33	0.507	12.7	100
Dyn1	40	33	11	0.507	12.7	100
Dyn1	3.15	11	0.416	0.7	7	102.5

Table 16. *Transformer data for the 452745-bus electrical network*

The data corresponding to capacitor banks is as follows:

Nodes	Reactive power Phase A [kVAR]	Reactive power Phase B [kVAR]	Reactive power Phase C [kVAR]
30, 31, 32, 33, 34, 35	80	80	80

Table 17. *Capacitor data for the 452745-bus electrical network*

Load model	Number of loads	Minimum active power per phase [kW]	Maximum active power per phase [kW]	Minimum reactive power per phase [kVAR]	Maximum reactive power per phase [kVAR]	Network total active power [kW]	Network total reactive power [kVAR]
Y-PQ – 1PH	3879	0.4480	1.4720	0.1473	0.4838	9653.07	3172.81
Y-PQ – 2PH	11757	0.2240	0.7360	0.0736	0.2419	9576.70	3147.71
Y-PQ – 3PH	11864	0.1493	0.4907	0.0491	0.1613	3146.22	1034.11

Table 18. *Load data for the 452745-bus electrical network*

The case was solved with an absolute tolerance of 1e-10 for the MATLAB implementation and with an absolute tolerance of 1e-10 and a relative tolerance of 1e-20 for the C / PETSc implementation.

Figures 18 and 19 shows the voltage magnitudes and angles obtained for the buses at nominal voltages 33 and 69 kV:

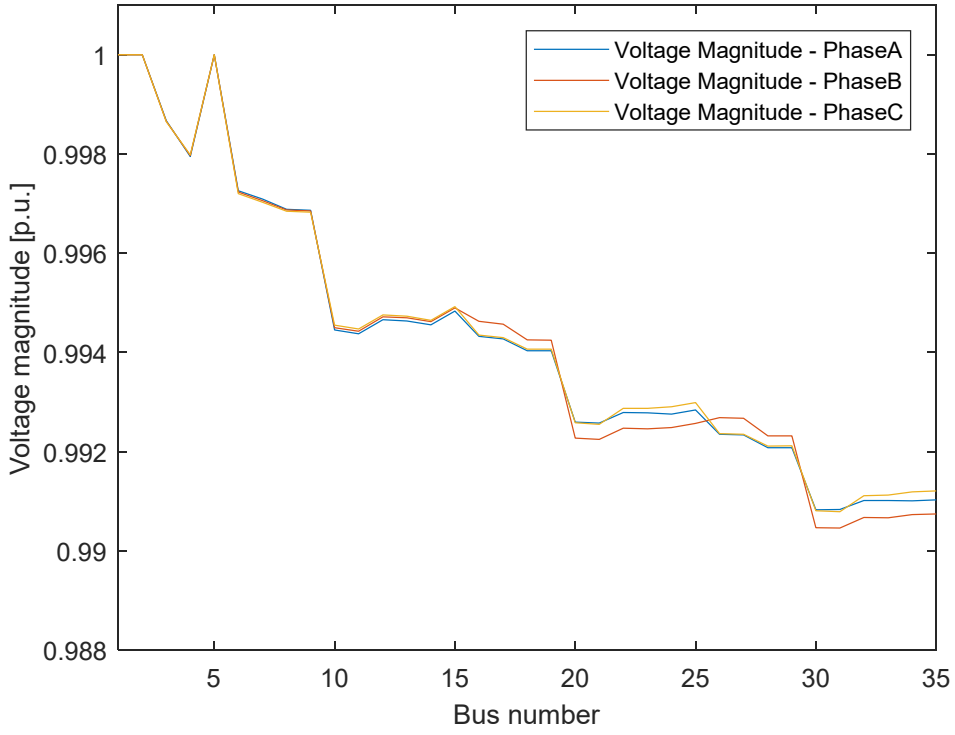


Figure 18. *Voltage magnitude of 69 and 33 kV buses of the 452745-bus electrical network*

In the figure 18, it can be observed how the voltage magnitudes of the different phases of the buses in the transmission system are very close one to each other, even though the load in the distribution section was randomly distributed along the phases.

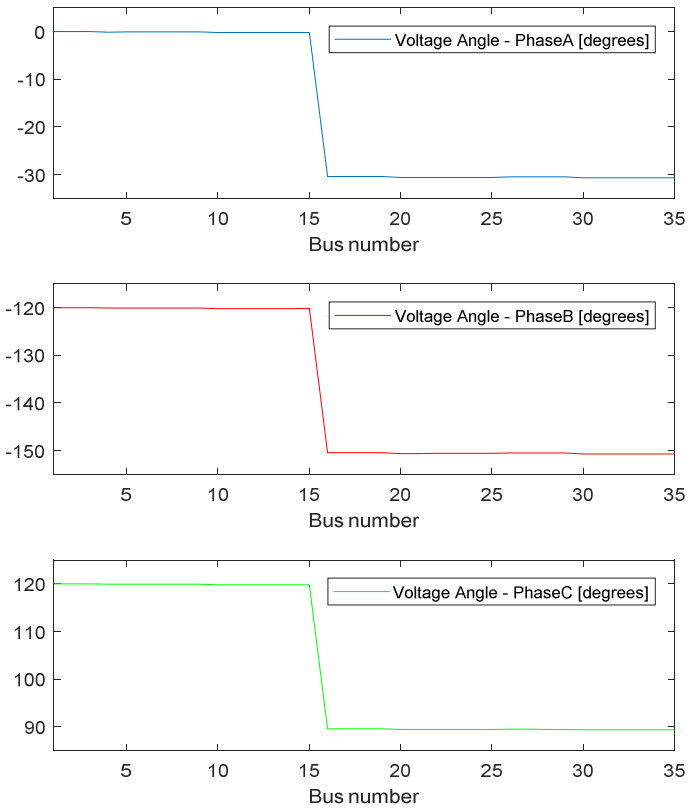


Figure 19. *Voltage angle of 69 and 33 kV buses of the 452745-bus electrical network*

In the figure 19, it can be observed the different voltage angles per phase and how they are shifted 30 degrees after the Dyn1 transformers from 69 kV to 33 kV.

At distribution voltage levels, the voltage magnitudes per phase can be considerably different. This is depicted in figure 20. The first bus shown in this figure is in the secondary winding of one 11 kV to 416 V transformer. On these transformers one tab at 102.5% is being selected in the secondary winding to increase the voltage magnitude in the feeder, which effect is clearly observed. For that branch, the power consumption in phase A is 16.7 kW, in phase B it is 14.2 kW and in phase C it is 13.8 kW, what explains why the voltage goes the lowest on phase A.

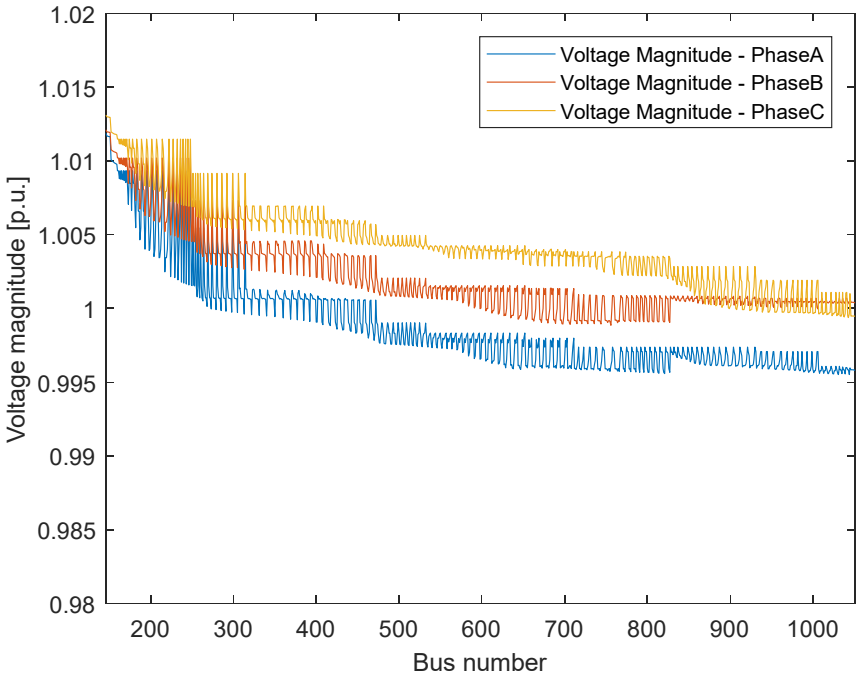


Figure 20. *Voltage magnitude of 416 V buses at one distribution branch of the 452745-bus electrical network*

Regarding the convergence of the solver, table 19 summarizes the number of iterations and the execution times for the MATLAB implementation.

Implementation	Number of iterations	Data preparation time [s]	Nonlinear solver convergence time [s]
MATLAB	7	35.215	148.673

Table 19. *Iterations and time of convergence of MATLAB implementation when solving the 452745-bus electrical network*

The following figure will depict the residual norm of the objective function vs the iteration counter on the MATLAB implementation:

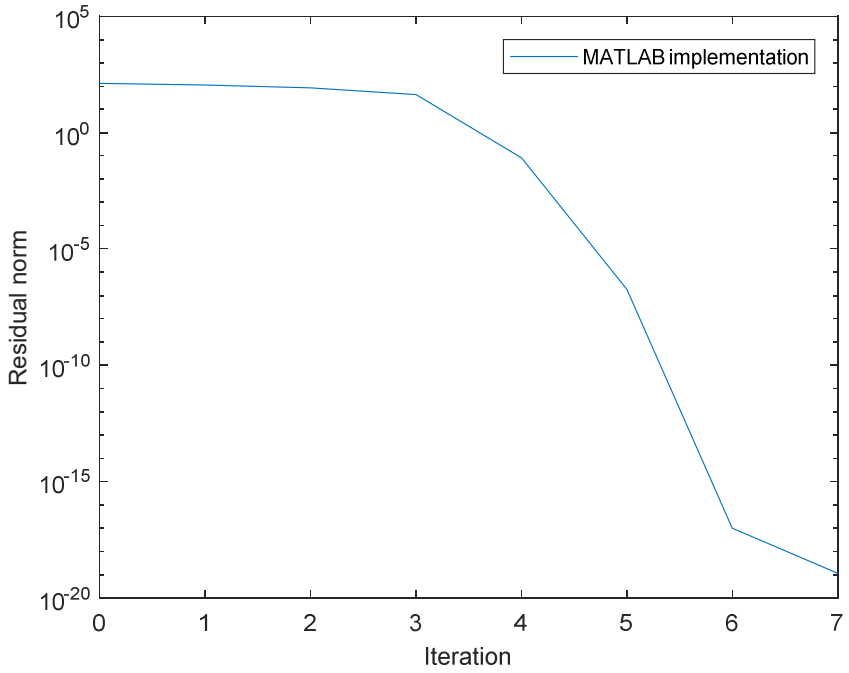


Figure 21. *Residual norm vs iteration number for MATLAB implementation when solving the 452745-bus electrical network*

In the case of the PETSc implementation, the settings were modified six times, in order to see how it affects the execution times. The following settings remained constant for all the tests:

- Nonlinear solver: Newton's method with linear search
- Absolute tolerance: 1e-10
- Relative tolerance 1e-20
- Preconditioner type: LU

The following settings were modified:

1. KSP type: GMRES; Preconditioner matrix reordering: AMD
2. KSP type: GMRES; Preconditioner matrix reordering: none
3. KSP type: Stabilized bi-conjugate gradient; Preconditioner matrix reordering: AMD
4. KSP type: Stabilized bi-conjugate gradient; Preconditioner matrix reordering: none
5. KSP type: Preconditioner only; Preconditioner matrix reordering: AMD
6. KSP type: Preconditioner only; Preconditioner matrix reordering: none

Table 20 summarizes the number of iterations and execution times obtained for the different tests:

Implementation	Number of iterations	Data preparation time [s]	Nonlinear solver convergence time [s]
C / PETSc test 1	4	5.2771	34.993
C / PETSc test 2	4	5.2744	38.761
C / PETSc test 3	4	5.3227	35.794
C / PETSc test 4	4	5.3924	39.848
C / PETSc test 5	4	5.3286	34.497
C / PETSc test 6	4	5.2991	38.055

Table 20. *Iterations and time of convergence of different PETSc implementations when solving the 452745-bus electrical network*

Table 20 allows to see that for this particular case, the convergence times with different linear solvers are very similar, however it is also clearly noticed that the reordering of the preconditioner matrices improve the performance of all the linear solvers.

The following figure will depict the residual norm of the objective function vs the iteration counter on the C / PETSc implementations. It is almost the same for all the tests, since it is related to the convergence of the nonlinear solver and not to the convergence of the linear solutions per iteration:

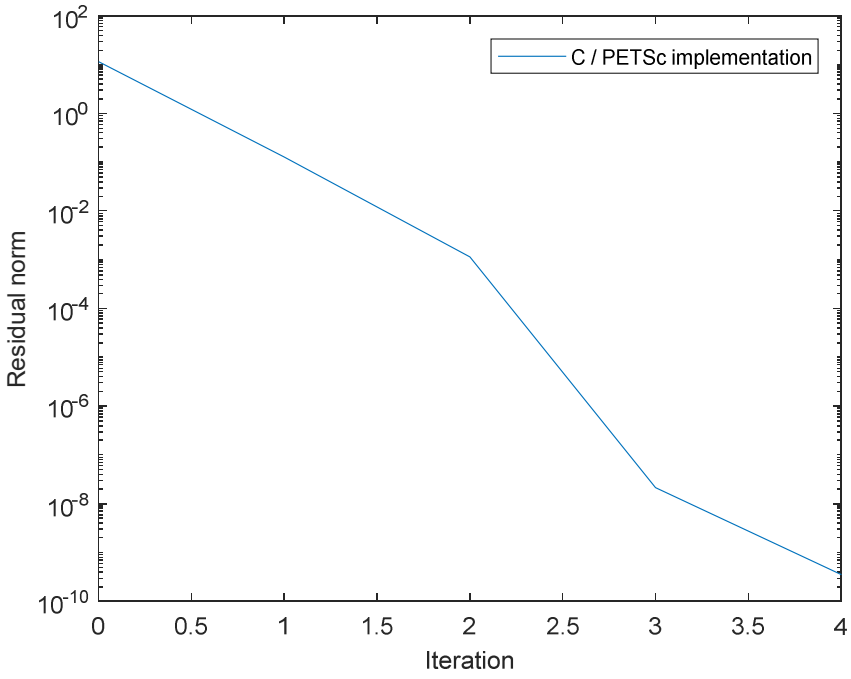


Figure 22. *Residual norm vs iteration number for PETSc implementations when solving the 452745-bus electrical network*

7 Conclusions and Recommendations

7.1. Conclusions

- Typical distribution network elements have been modelled under the phase-frame, using the nodal admittance matrix.

- A software that can handle unbalanced three-phase power networks has been developed using the MATLAB environment and C code with the support of the PETSc library for the solution of large linear and nonlinear system of equations.

- The program has been able to solve the IEEE 4-bus test feeder and the IEEE 13-bus test feeder with high accuracy, proving the proper modelling of the element of the electrical network.

- A network with 3x452745 buses has been solved with a short convergence time, which shows that it is able to calculate power flow over large-scale distribution networks.

7.2. Recommendations

- This software allows to solve electrical networks over the phase-frame, therefore a short-circuit calculator without the use of symmetrical components might be implemented. A performance comparison between both approaches would be needed to confirm what is the best approach for this problem.

- Since a power flow is already available for large distribution networks, optimal power flow calculations can be implemented as the next step. In future networks, where energy trading between consumers and the network will exist, optimal power flow calculations in low voltage networks will also be required.

Bibliography

[1] J. Arrillaga and C.P. Arnold, “Computer Analysis of Power Systems”, 1990, published by John Wiley and Sons Ltd.

[2] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. “PETSc Web page: *http://www.mcs.anl.gov/petsc*”, 2016.

[3] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. “PETSc User’s Manual”, 2016, published by Argonne National Laboratory.

[4] Satish Balay and William D. Gropp and Lois Curfman McInnes and Barry F. Smith. “Efficient Management of Parallelism in Object Oriented Numerical Software Libraries”, 1997, published by Birkhäuser press.

[5] William H. Kersting, “Distribution System Modeling and Analysis”, 2012, published by Taylor & Francis Group, LLC.

[6] Abdelhay A. Sallam and Om P. Malik, “Electric Distribution Systems”, 2011, published by John Wiley & Sons.

[7] Thomas Allen Short, “Electric Power Distribution Handbook”, 2004, published by Taylor & Francis Group, LLC.

[8] Juan M Gers, “Distribution System Analysis and Automation”, 2014, published by The Institution of Engineering and Technology (IET).

[9] Anthony J. Pansini, “Electrical Distribution Engineering”, 2007, Published by The Fairmont Press, Inc.

[10] John Grainger and William Stevenson, “Power System Analysis”, 1994, published by Search Results McGraw-Hill Education.

[11] Paulo A. N. Garcia, Jose Luiz R. Pereira, Member, Sandoval Carneiro, Vander M. da Costa and Nelson Martins, “Three-Phase Power Flow Calculations Using the Current Injection Method”, IEEE Transactions on Power Systems, Vol. 15, No. 2, May 2000.

[12] IEEE Distribution Planning Working Group Report, “Radial distribution test feeders”, IEEE Transactions on Power Systems, August 1991, Volume 6, Number 3.

[13] IEEE Distribution Planning Working Group Report, “IEEE 4 Node Test Feeder”, IEEE Transactions on Power Systems, Revised Sept. 19, 2006.

[14] R. Dugan: “IEEE 13 Node Test Feeder”, Results of the test feeder, released by IEEE PES Distribution System Analysis Subcommittee.

[15] Nikos Hatziargyriou, “Microgrids Architectures and Control”, 2014, published by John Wiley & Sons.

[16] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education," Power Systems, IEEE Transactions on, vol. 26, no. 1, pp. 12-19, Feb. 2011. (Digital Object Identifier: 10.1109/TPWRS.2010.2051168)

[17] Cédric Josz, Stéphane Fliscounakis, Jean Maeght, Patrick Panciatici, “AC power flow data in MATPOWER and QCQP format: iTesla, RTE snapshots, and PEGASE”.

[18] Reijer Idema, “Newton-Krylov Methods in Power Flow and Contingency Analysis”, 2012, Dissertation at Delft University of Technology.

[19] MathWorks, MATLAB team. “MATLAB documentation: fsolve function”. Retrieved from *https://www.mathworks.com/help/optim/ug/fsolve.html*.

[20] IEEE Distribution Planning Working Group Report, “The IEEE European Low Voltage Test Feeder”, IEEE Transactions on Power Systems, Revised May 19, 2015.

[21] Saad, Y., “Iterative methods for sparse linear systems, 2nd edition”. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2003.