COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

STEADY-STATE ANALYSIS OF POWER NETWORKS BASED ON DROOP-CONTROLLED INVERTERS

Autor: Minerva Carballo Palacio

Director: Alejandro Domínguez-García

University of Illinois at Urbana-Champaign
Champaign, IL
Mayo 2022

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título Steady-state Analysys of Power Networks based on Droop-controlled

Inverters

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2021-2022 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada

de otros documentos está debidamente referenciada.

Fdo.: Minerva Carballo Palacio Fecha: 23/ 05/ 2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Alejandro Domínguez García Fecha: 24/05/2022

A. D. Domínguez

COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

STEADY-STATE ANALYSIS OF POWER NETWORKS BASED ON DROOP-CONTROLLED INVERTERS

Autor: Minerva Carballo Palacio

Director: Alejandro Domínguez-García

University of Illinois at Urbana-Champaign
Champaign, IL
Mayo 2022

Abstract

Steady-state analysis of bulk power networks is typically performed by solving the so-called power flow problem. This problem can be solved through two different approaches: either by considering a single generation bus to take all the slack (single slack bus) or by considering a distributed slack bus (all generation buses contribute to cover the slack). In this work, I will focus on a distributed slack bus formulation of the power flow problem for the steady-state analysis.

The generator buses active power output is modelled using an active power setpoint modulated by the power imbalance between total generation and net demand, whereas the gen-buses reactive power injected into the grid is usually considered as an unknown variable. The load buses active (reactive) power injection is represented by the corresponding active (reactive) power demand preceded by a negative sign. Then, the power flow problem is solved for all buses' angles and load-buses' voltages, since the gen-buses' voltages are set to a constant setpoint value (normally about 1 p.u.). A gen-bus angle must be fixed to a certain value (usually 0 rad without loss of generality).

However, the approach in which we are working on over this project differs significantly from the one explained above. The reactive power injected by the generators is no longer a simple variable but depends on two terms: a reactive power setpoint and a voltage mismatch allocated by another participation factor. Moreover, the gen-buses' voltages are not fixed to a setpoint anymore, so we need to solve the model for them as well. Therefore, the resulting unknowns finally include all buses' not only angles but also voltages (apart from the angle which we need to fix to 0 rad), and complexity considerably increases.

To address the power flow problem described above, we will be working on the implementation of Newton-Raphson in an adequate software platform. Python will be the chosen programming language to code the mentioned algorithm, as well as several more Quasi-Newton methods based on simultaneous and/or non-simultaneous updates of the state variables.

Subject Keywords: Power Flow Problem; Newton-Raphson; Quasi-Newton-Raphson; Voltage Droop Control; Distributed Slack Bus; Grid-Forming Inverters.

Acknowledgments

I would like to begin by sincerely thanking my advisor, Prof. Alejandro Domínguez-García, for offering me the opportunity to work on this project, and for assisting me in completing my Senior Thesis. I also wish to acknowledge the great support and patience I have received from Olaolu Ajala during all the project. I would also like to thank my family and friends, as without their encouragement I would not have been able to finish the project. Finally, I would also like to express my gratitude to the University of Illinois at Urbana-Champaign, as well as to Universidad Pontificia de Comillas ICAI for making this exchange year possible.

ANÁLISIS EN ESTADO ESTACIONARIO DE REDES ELÉCTRICAS BASADAS EN INVERSORES DE DROOP CONTROL

Autor: Carballo Palacio, Minerva.
Director: Domínguez-García, Alejandro.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Se implementarán los métodos de Newton-Raphson y Quasi-Newton-Raphson, de forma genérica, a través de Python, para resolver el problema del Flujo de Cargas adaptado a todo sistema eléctrico. Las ecuaciones del flujo de cargas incluyen la utilización de fuentes de generación que requieren inversores Grid-Forming, por lo que se introduce un control de la tensión basado en Droop Control. En primer lugar, se desarrolla el método de Newton-Raphson tradicional, incluyendo la actualización del Droop Control y partiendo de los resultados obtenidos, se implementa el método de Quasi-Newton-Raphson realizando simplificaciones sobre el método anterior, como la utilización de líneas de transmisión cortas y sin pérdidas.

Palabras clave: Problema de Flujo de Cargas; Newton-Raphson; Quasi-Newton-Raphson; Droop Control de Tensión; Slack distribuido; Inversores Grid-Forming.

1. Introducción

Hoy en día, nuestra sociedad depende en gran medida en la energía eléctrica. Por esto, se necesita un estudio en profundidad del funcionamiento y comportamiento de la red eléctrica, para lo que se necesita resolver el problema del flujo de cargas.

En cuanto a la resolución del flujo de cargas, hay que tener en cuenta que este modelo que describe los flujos de potencia activa y reactiva por la red eléctrica no es lineal [1], por lo que hay que recurrir a la utilización de métodos numéricos para alcanzar una solución. Los métodos utilizados en este proyecto son el método de Newton-Raphson y el método de Quasi-Newton-Raphson [2].

La solución de estos métodos permite conocer la tensión de cada bus del sistema, es decir, el fasor completo de la tensión, que se divide en módulo y ángulo. Estos resultados se introducen en las ecuaciones del flujo de cargas, obteniendo la potencia activa y reactiva inyectadas en cada bus del sistema eléctrico estudiado, además, todo esto permite conocer el comportamiento del sistema en estado estacionario.

Por otra parte, la transición de generadores síncronos a formas de generación que requieren el uso de inversores grid-forming ha modificado el funcionamiento de la red eléctrica [3]. Por tanto, es necesario introducir cambios en las ecuaciones tradicionales del flujo de cargas para reflejar el comportamiento de los inversores.

Teniendo estas modificaciones de las ecuaciones en cuenta, para conocer el comportamiento en estado estacionario de un sistema eléctrico se necesita resolver el problema del flujo de cargas actualizado, utilizando los métodos de Newton-Raphson y Quasi-Newton-Raphson.

2. Definición del proyecto

El proyecto se centra en la implementación de los métodos de Newton-Raphson y Quasi-Newton-Raphson para resolver el problema del flujo de cargas con las ecuaciones actualizadas, teniendo en cuenta las modificaciones introducidas por el uso de inversores grid-forming.

Para llevar esto a cabo, en primer lugar, se realizará la formulación de las ecuaciones del flujo de cargas con las modificaciones convenientes. Se continúa con la implementación de un código que permita leer los datos del sistema eléctrico que se va a estudiar. Estos datos permiten la generación de la matriz de admitancias que define el sistema, así como parámetros utilizados en las ecuaciones de flujo de cargas, como son los factores de participación tanto de potencia activa como reactiva de cada generador.

Para la implementación de cada método, se crea un código para la construcción del vector mismatch, que contiene el mismatch de potencia activa de cada bus, así como un código diferente para el cálculo de la matriz Jacobiana. Además, se genera otro código que actualiza los valores del módulo de la tensión, los ángulos y la variable slack. Por último, se crea un código principal que utiliza todos los códigos anteriores para la resolución del problema del flujo de cargas estableciendo un número máximo de iteraciones, así como un parámetro para detener las iteraciones cuando sea alcanzado.

Estos cinco códigos se implementan para cada método, ya que el planteamiento de la construcción de la matriz Jacobiana y el vector mismatch es diferente para el método de Newton-Raphson y el de Quasi-Newton-Raphson, así como la forma de lectura de los datos y la construcción de las ecuaciones del flujo de cargas.

3. Descripción del modelo/sistema/herramienta

Para comprobar que los métodos implementados alcanzan una solución correcta, se creará un sistema eléctrico de prueba con 5 buses, de los cuales, 3 generadores y 2 cargas. Este sistema se muestra en la Ilustración 1.

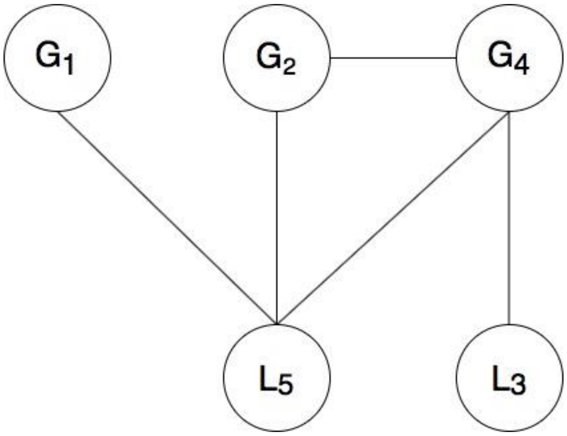
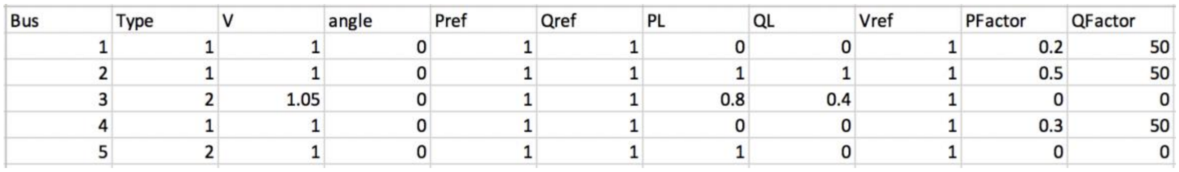


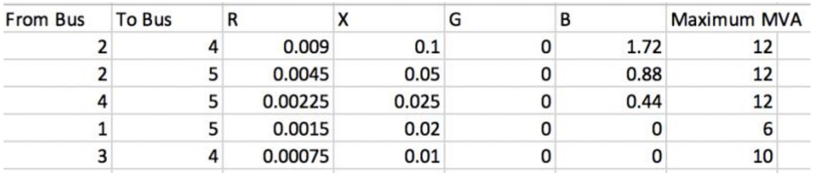
Ilustración 1 - Esquema del sistema eléctrico de prueba

Además, se creará un archivo Excel conteniendo los datos necesarios para construir la matriz de admitancias, así como parámetros necesarios en la implementación de las ecuaciones del flujo de cargas. Estas hojas de Excel se muestran en la Ilustración 2 e Ilustración 3.



Bus	Type	V	angle	Pref	Qref	PL	QL	Vref	PFactor	QFactor
1	1	1	0	1	1	0	0	1	0.2	50
2	1	1	0	1	1	1	1	1	0.5	50
3	2	1.05	0	1	1	0.8	0.4	1	0	0
4	1	1	0	1	1	0	0	1	0.3	50
5	2	1	0	1	1	1	0	1	0	0

Ilustración 2 – Datos principales del sistema eléctrico de prueba



From Bus	To Bus	R	X	G	B	Maximum MVA
2	4	0.009	0.1	0	1.72	12
2	5	0.0045	0.05	0	0.88	12
4	5	0.00225	0.025	0	0.44	12
1	5	0.0015	0.02	0	0	6
3	4	0.00075	0.01	0	0	10

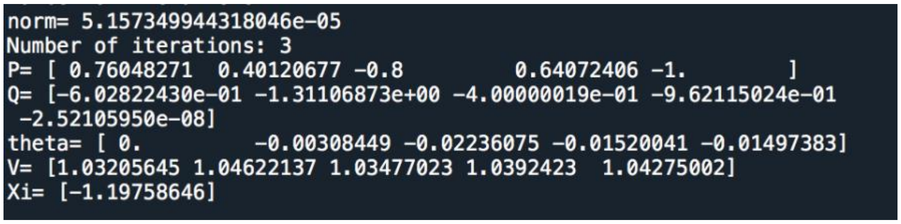
Ilustración 3 – Datos principales de las líneas de transmisión del sistema eléctrico de prueba

4. Resultados

- La matriz de admitancias obtenida para el sistema eléctrico de prueba es la presentada a continuación:

$$\mathrm{Y} = \begin{bmatrix} 3.72902424 - 49.72032318j & 0+0j & 0+0j & 0+0j & -3.72902424 + 49.72032318j \\ 0+0j & 2.67830572 - 28.45895248j & 0+0j & -0.89276857 + 9.91965083j & -1.78553715 + 19.83930166j \\ 0+0j & 0+0j & 7.45804848 - 99.44064636j & -7.45804848 + 99.44064636j & 0+0j \\ 0+0j & -0.89276857 + 9.91965083j & -7.45804848 + 99.44064636j & 11.92189135 - 147.95890051j & -3.5710743 + 39.67860331j \\ -3.72902424 + 49.72032318 & -1.78553715 + 19.83930166j & 0+0j & -3.5710743 + 39.67860331j & 9.08563569 - 108.57822815j]] \end{bmatrix}$$

- A partir del método de Newton-Raphson aplicado al sistema de prueba se obtuvieron los siguientes resultados, recogidos en la Ilustración 4.



```
norm= 5.157349944318046e-05
Number of iterations: 3
P= [ 0.76048271  0.40120677 -0.8         0.64072406 -1.        ]
Q= [-6.02822430e-01 -1.31106873e+00 -4.00000019e-01 -9.62115024e-01
 -2.52105950e-08]
theta= [ 0.         -0.00308449 -0.02236075 -0.01520041 -0.01497383]
V= [1.03205645 1.04622137 1.03477023 1.0392423  1.04275002]
Xi= [-1.19758646]
```

Ilustración 4 – Resultados método Newton-Raphson

Estos resultados aparecen en un orden concreto según el bus al que pertenecen, el orden es el siguiente:

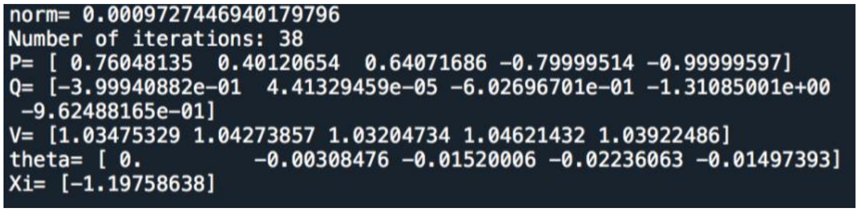
$$\mathrm{P} = [\mathrm{P}_1 \quad \mathrm{P}_2 \quad \mathrm{P}_3 \quad \mathrm{P}_4 \quad \mathrm{P}_5]$$

$$Q = [Q_1 \quad Q_2 \quad Q_3 \quad Q_4 \quad Q_5]$$

$$\theta = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5]$$

$$V = [V_1 \quad V_2 \quad V_3 \quad V_4 \quad V_5]$$

- A partir del método de Quasi-Newton-Raphson aplicado al sistema de prueba se obtuvieron los siguientes resultados, recogidos en la Ilustración 5.



```
norm= 0.0009727446940179796
Number of iterations: 38
P= [ 0.76048135  0.40120654  0.64071686 -0.79999514 -0.99999597]
Q= [-3.99940882e-01  4.41329459e-05 -6.02696701e-01 -1.31085001e+00
 -9.62488165e-01]
V= [1.03475329 1.04273857 1.03204734 1.04621432 1.03922486]
theta= [ 0.         -0.00308476 -0.01520006 -0.02236063 -0.01497393]
Xi= [-1.19758638]
```

Ilustración 5 – Resultados método Quasi-Newton-Raphson

Si se comparan directamente estos resultados con los del método Newton-Raphson, éstos no coinciden. Esto se debe a que los resultados en este método se ofrecen en un orden diferente, debido a reordenaciones en la matriz Jacobiana y en el vector mismatch. El orden utilizado es el siguiente:

$$P = [P_1 \quad P_2 \quad P_4 \quad P_3 \quad P_5]$$

$$Q = [Q_3 \quad Q_5 \quad Q_1 \quad Q_2 \quad Q_4]$$

$$\theta = [\theta_1 \quad \theta_2 \quad \theta_4 \quad \theta_3 \quad \theta_5]$$

$$V = [V_3 \quad V_5 \quad V_1 \quad V_2 \quad V_4]$$

5. Conclusiones

El proyecto pretende demostrar que utilizando dos métodos distintos, como son el método de Newton-Raphson y el de Quasi-Newton-Raphson, se consiguen los mismos resultados al resolver el flujo de cargas. Para ello, se introduce la Tabla 1 comparando los resultados obtenidos para cada método.

Tabla 1 Comparación de resultados

Variables	**Newton-Raphson**	**Quasi-Newton-Raphson**
P_1 [p.u]	0.76048271	0.76048135
P_2[p.u]	0.40120677	0.40120654
P_3[p.u]	-0.8	-0.79999514
P_4[p.u]	0.64072406	0.64071686
P_5[p.u]	-1	-0.99999597
Q_1[p.u]	-0.60282243	-0.602696701
Q_2[p.u]	-1.31106873	-1.31085001
Q_3[p.u]	-0.4	-0.399940882
Q_4[p.u]	-0.962115024	-0.962488165
Q_5[p.u]	-0	-0
θ_1[Rad]	0	0
θ_2[Rad]	-0.00308449	-0.00308476
θ_3[Rad]	-0.02236075	-0.02236063
θ_4[Rad]	-0.01520041	-0.01520006
θ_5[Rad]	-0.01497383	-0.01497393
V_1[p.u]	1.03205645	1.03204734
V_2[p.u]	1.04622137	1.04621432
V_3[p.u]	1.03477023	1.03475329
V_4[p.u]	1.0392423	1.03922486
V_5[p.u]	1.04275002	1.04273857
ξ[p.u]	-1.19758646	-1.19758638
Número de iteraciones	3	38
norm	5.1573499443180e-05	0.0009727446940179796

Se puede observar que los resultados obtenidos son muy similares, pero no idénticos. Ésta diferencia entre ellos se podría reducir disminuyendo la constante que se ha utilizado como umbral para detener el proceso iterativo de ambos métodos.

Por otra parte, se demuestra que la matriz Jacobiana utilizada en el método de Quasi-Newton-Raphson (Ilustración 6) se puede dividir a su vez en varias matrices más pequeñas cumpliendo lo siguiente:

$$\begin{bmatrix} -\tilde{\alpha} & \tilde{J}(\tilde{x}^{\nu}) & 0_{(2n-m)\times m} \\ 0_m & 0_{m\times(2n-m-1)} & F(\tilde{x}^{\nu}) \end{bmatrix} \begin{bmatrix} \Delta\xi^{\nu} \\ \widetilde{\Delta x}^{\nu} \\ \widetilde{\Delta v}^{\nu} \end{bmatrix} = - \begin{bmatrix} \tilde{f}(\xi^{\nu}, \tilde{x}^{\nu}, \tilde{v}^{\nu}) \\ \tilde{f}(\tilde{x}^{\nu}, \tilde{v}^{\nu}) \end{bmatrix}$$

$$\tilde{J}(\tilde{x}^{\nu})\widetilde{\Delta x}^{\nu} = -\tilde{f}(\xi^{\nu}, \tilde{x}^{\nu}, \tilde{v}^{\nu}) + \tilde{\alpha}\Delta\xi^{\nu}$$

$$F(\tilde{x}^{\nu})\widetilde{\Delta v}^{\nu} = -\tilde{f}(\tilde{x}^{\nu}, \tilde{v}^{\nu})$$

$$= \overset{-\tilde{\alpha}\qquad\qquad \mathcal{J}(\tilde{x}^{\nu})\qquad\qquad D(x^{\nu})}{\underset{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad F(\tilde{x}^{\nu})}{\begin{bmatrix}
-\alpha_1 & \left.\frac{\partial p_1}{\partial \theta_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_1}{\partial \theta_4}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_1}{\partial \theta_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_1}{\partial \theta_5}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_1}{\partial V_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_1}{\partial V_5}\right|_{\theta^{v},v^{v}} & 0 & 0 & 0 \\
-\alpha_2 & \left.\frac{\partial p_2}{\partial \theta_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_2}{\partial \theta_4}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_2}{\partial \theta_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_2}{\partial \theta_5}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_2}{\partial V_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_2}{\partial V_5}\right|_{\theta^{v},v^{v}} & 0 & 0 & 0 \\
-\alpha_4 & \left.\frac{\partial p_4}{\partial \theta_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_4}{\partial \theta_4}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_4}{\partial \theta_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_4}{\partial \theta_5}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_3}{\partial V_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_4}{\partial V_5}\right|_{\theta^{v},v^{v}} & 0 & 0 & 0 \\
0 & \left.\frac{\partial p_3}{\partial \theta_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_3}{\partial \theta_4}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_3}{\partial \theta_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_3}{\partial \theta_5}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_3}{\partial V_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_3}{\partial V_5}\right|_{\theta^{v},v^{v}} & 0 & 0 & 0 \\
0 & \left.\frac{\partial p_5}{\partial \theta_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_5}{\partial \theta_4}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_5}{\partial \theta_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_5}{\partial \theta_5}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_5}{\partial V_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial p_5}{\partial V_5}\right|_{\theta^{v},v^{v}} & 0 & 0 & 0 \\
0 & \left.\frac{\partial q_3}{\partial \theta_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_3}{\partial \theta_4}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_3}{\partial \theta_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_3}{\partial \theta_5}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_3}{\partial V_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_3}{\partial V_5}\right|_{\theta^{v},v^{v}} & 0 & 0 & 0 \\
0 & \left.\frac{\partial q_5}{\partial \theta_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_5}{\partial \theta_4}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_5}{\partial \theta_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_5}{\partial \theta_5}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_5}{\partial V_3}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_5}{\partial V_5}\right|_{\theta^{v},v^{v}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \left.\frac{\partial q_1}{\partial V_1}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_1}{\partial V_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_1}{\partial V_4}\right|_{\theta^{v},v^{v}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \left.\frac{\partial q_2}{\partial V_1}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_2}{\partial V_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_2}{\partial V_4}\right|_{\theta^{v},v^{v}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \left.\frac{\partial q_4}{\partial V_1}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_4}{\partial V_2}\right|_{\theta^{v},v^{v}} & \left.\frac{\partial q_4}{\partial V_4}\right|_{\theta^{v},v^{v}}
\end{bmatrix}}}$$

Ilustración 6 – Matriz Jacobiana método Quasi-Newton-Raphson

6. Referencias

[1] A. D. Domínguez-García, “Power Flow,” class notes for ECE 476, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2021.

[2] A. D. Domínguez-García, “Numerical Solution of PF,” class notes for ECE 476, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2021.

[3] O. Ajala, A. Domínguez-García, N. Baeckeland, S. Dhople, “Uncovering the Kuramoto Model from Full-order Models of Grid-forming Inverter-based Power Networks,” *IEEE Conference on Decision and Control*, December 2021.

STEADY-STATE ANALYSIS OF POWER NETWORKS BASED ON DROOP-CONTROLLED INVERTERS

Author: Carballo Palacio, Minerva.
Supervisor: Domínguez-García, Alejandro.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

Newton-Raphson and Quasi-Newton-Raphson methods will be implemented generically, using Python to solve the Power Flow problem adapted to any electrical system. The power flow balance equations include the use of generation sources that require Grid-Forming inverters, so a Droop Control based voltage control is introduced. First, the traditional Newton-Raphson method is developed, including the Droop Control update, and based on the results obtained, the Quasi-Newton-Raphson method is implemented, making simplifications on the previous method, such as the use of short and lossless transmission lines.

Subject Keywords: Power Flow Problem; Newton-Raphson; Quasi-Newton-Raphson; Voltage Droop Control; Distributed Slack Bus; Grid-Forming Inverters.

1. Introduction

Nowadays, our society relies heavily on electric power. For this reason, an in-depth study of the operation and behavior of the electrical network is needed, for which the power flow problem must be solved.

Regarding the resolution of the power flow, it must be considered that this model describing the active and reactive power flows through the electrical network is nonlinear [1]. So, it is necessary to resort to the use of numerical methods to reach a solution. The methods used in this project are the Newton-Raphson method and the Quasi-Newton-Raphson method [2].

The solution of these methods allows to know the voltage of each bus of the system, i. e., the complete phasor of the voltage, which is divided into module and angle. These results are introduced in the power flow equations, obtaining the active and reactive power injected in each bus of the electrical system studied. In addition, all this permits to determine the behavior of the system in steady state.

On the other hand, the transition from synchronous generators to sources of generation that require the use of grid-forming inverters has modified the operation of the power grid [3]. Therefore, it is necessary to introduce changes in the traditional power flow equations to reflect the behavior of these inverters.

Considering these modifications of the equations, in order to understand the steady state behavior of an electrical system, it is necessary to solve the updated power flow problem, using the Newton-Raphson and Quasi-Newton-Raphson methods.

2. Project Definition

The project is focused on the implementation of the Newton-Raphson and Quasi-Newton-Raphson methods to solve the power flow problem with the updated equations, bearing in mind the modifications introduced by the use of grid-forming inverters.

To accomplish this, first of all, the formulation of the power flow equations will be performed with the appropriate modifications. This is followed by the implementation of a code that allows reading the data of the electrical system to be studied. These data enable the generation of the admittance matrix that defines the system, as well as the parameters used in the power flow equations, such as the active and reactive power participation factors of each generator.

For the implementation of each method, a code is created for the construction of the mismatch vector, which contains the active power mismatch of each bus, as well as a different code for the calculation of the Jacobian matrix. In addition, another code is generated to update the values of the voltage module, the angles and the slack variable. Finally, a main code is created that uses all the previous codes for solving the power flow problem by setting a maximum number of iterations, as well as a parameter to stop the iterations when it is reached.

These five codes are implemented for each method, since the approach to the construction of the Jacobian matrix and the mismatch vector is different for the Newton-Raphson and the Quasi-

Newton-Raphson method, as well as the reading of the data and the construction of the power flow equations.

3. General Description of the System

To verify that the implemented methods achieve a correct solution, a test electrical system will be created with 5 buses, including 3 generators and 2 loads. This system is shown in Figure 1.

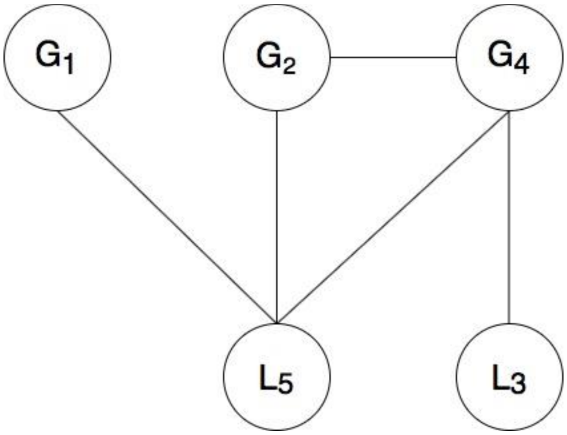
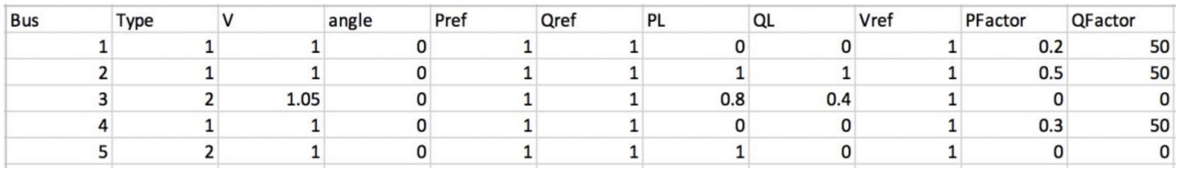


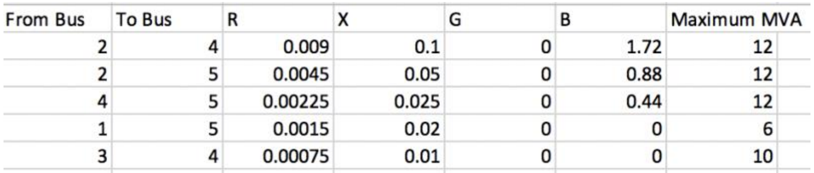
Figure 2 – Test Electrical Grid

In addition, an Excel file will be created containing the data necessary to construct the admittance matrix, as well as parameters needed in the implementation of the power flow equations. These Excel sheets are shown in Figure 2 and Figure 3.



Bus	Type	V	angle	Pref	Qref	PL	QL	Vref	PFactor	QFactor
1	1	1	0	1	1	0	0	1	0.2	50
2	1	1	0	1	1	1	1	1	0.5	50
3	2	1.05	0	1	1	0.8	0.4	1	0	0
4	1	1	0	1	1	0	0	1	0.3	50
5	2	1	0	1	1	1	0	1	0	0

Figure 2 – Bus main data of the power network studied



From Bus	To Bus	R	X	G	B	Maximum MVA
2	4	0.009	0.1	0	1.72	12
2	5	0.0045	0.05	0	0.88	12
4	5	0.00225	0.025	0	0.44	12
1	5	0.0015	0.02	0	0	6
3	4	0.00075	0.01	0	0	10

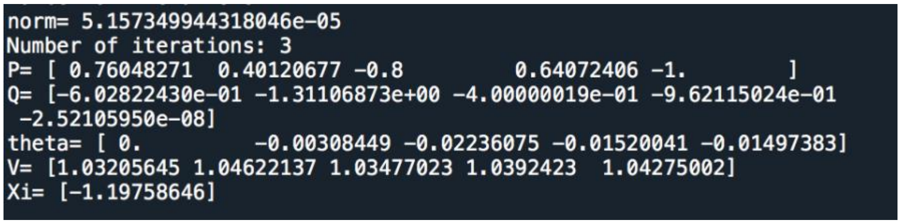
Figure 3 – Transmission lines main data of the power network studied

4. Results

- The admittance matrix obtained for the test electrical system is presented below:

$$\mathrm{Y} = \begin{bmatrix} 3.72902424\ -49.72032318j & 0+0j & 0+0j & 0+0j & -3.72902424\ +49.72032318j \\ 0+0j & 2.67830572\ -28.45895248j & 0+0j & -0.89276857\ +9.91965083j & -1.78553715\ +19.83930166j \\ 0+0j & 0+0j & 7.45804848\ -99.44064636j & -7.45804848\ +99.44064636j & 0+0j \\ 0+0j & -0.89276857\ +9.91965083j & -7.45804848\ +99.44064636j & 11.92189135-147.95890051j & -3.5710743\ +39.67860331j \\ -3.72902424\ +49.72032318 & -1.78553715\ +19.83930166j & 0+0j & -3.5710743\ +39.67860331j & 9.08563569-108.5782815j]] \end{bmatrix}$$

- Based on the Newton-Raphson method applied to the test system, the following results were obtained, as shown in Figure 4.



```
norm= 5.157349944318046e-05
Number of iterations: 3
P= [ 0.76048271  0.40120677 -0.8         0.64072406 -1.        ]
Q= [-6.02822430e-01 -1.31106873e+00 -4.00000019e-01 -9.62115024e-01
 -2.52105950e-08]
theta= [ 0.         -0.00308449 -0.02236075 -0.01520041 -0.01497383]
V= [1.03205645 1.04622137 1.03477023 1.0392423  1.04275002]
Xi= [-1.19758646]
```

Figure 4 – Results obtained for Newton-Raphson method

These results are listed in a specific order according to the bus they belong to, the order is as follows:

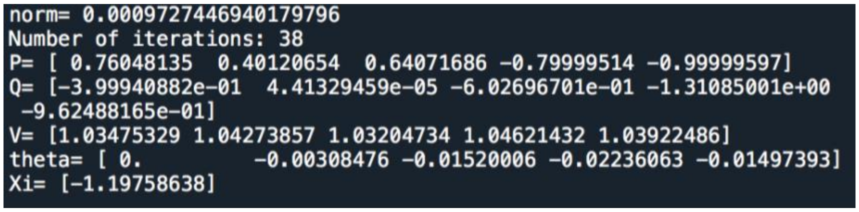
$$\mathrm{P} = [\mathrm{P}_1 \quad \mathrm{P}_2 \quad \mathrm{P}_3 \quad \mathrm{P}_4 \quad \mathrm{P}_5]$$

$$\mathrm{Q} = [\mathrm{Q}_1 \quad \mathrm{Q}_2 \quad \mathrm{Q}_3 \quad \mathrm{Q}_4 \quad \mathrm{Q}_5]$$

$$\theta = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5]$$

$$\mathrm{V} = [\mathrm{V}_1 \quad \mathrm{V}_2 \quad \mathrm{V}_3 \quad \mathrm{V}_4 \quad \mathrm{V}_5]$$

- The following results were obtained from the Quasi-Newton-Raphson method applied to the test system, as shown in Figure 5.



```
norm= 0.0009727446940179796
Number of iterations: 38
P= [ 0.76048135  0.40120654  0.64071686 -0.79999514 -0.99999597]
Q= [-3.99940882e-01  4.41329459e-05 -6.02696701e-01 -1.31085001e+00
 -9.62488165e-01]
V= [1.03475329 1.04273857 1.03204734 1.04621432 1.03922486]
theta= [ 0.         -0.00308476 -0.01520006 -0.02236063 -0.01497393]
Xi= [-1.19758638]
```

Figure 5 – Results obtained for Quasi-Newton-Raphson method

If these results are directly compared with those of the Newton-Raphson method, they do not coincide. This is due to the fact that the results in this method are given in a different order, as a result of rearrangements in the Jacobian matrix and in the mismatch vector. The order used is as follows:

$$P = [P_1 \quad P_2 \quad P_4 \quad P_3 \quad P_5]$$

$$Q = [Q_3 \quad Q_5 \quad Q_1 \quad Q_2 \quad Q_4]$$

$$\theta = [\theta_1 \quad \theta_2 \quad \theta_4 \quad \theta_3 \quad \theta_5]$$

$$V = [V_3 \quad V_5 \quad V_1 \quad V_2 \quad V_4]$$

5. Conclusion

The project aims to demonstrate that using two different methods, such as the Newton-Raphson method and the Quasi-Newton-Raphson method, the same results are obtained when solving the power flow. For this purpose, Table 1 is introduced comparing the results obtained for each method.

Table 1 Comparison of results obtained

Variables	Newton-Raphson	Quasi-Newton-Raphson
P_1 [p.u]	0.76048271	0.76048135
P_2[p.u]	0.40120677	0.40120654
P_3[p.u]	-0.8	-0.79999514
P_4[p.u]	0.64072406	0.64071686
P_5[p.u]	-1	-0.99999597
Q_1[p.u]	-0.60282243	-0.602696701
Q_2[p.u]	-1.31106873	-1.31085001
Q_3[p.u]	-0.4	-0.399940882
Q_4[p.u]	-0.962115024	-0.962488165
Q_5[p.u]	-0	-0

θ_1[Rad]	0	0
θ_2[Rad]	-0.00308449	-0.00308476
θ_3[Rad]	-0.02236075	-0.02236063
θ_4[Rad]	-0.01520041	-0.01520006
θ_5[Rad]	-0.01497383	-0.01497393
V_1[p.u]	1.03205645	1.03204734
V_2[p.u]	1.04622137	1.04621432
V_3[p.u]	1.03477023	1.03475329
V_4[p.u]	1.0392423	1.03922486
V_5[p.u]	1.04275002	1.04273857
ξ[p.u]	-1.19758646	-1.19758638
Number of iterations	3	38
norm	5.1573499443180e-05	0.0009727446940179796

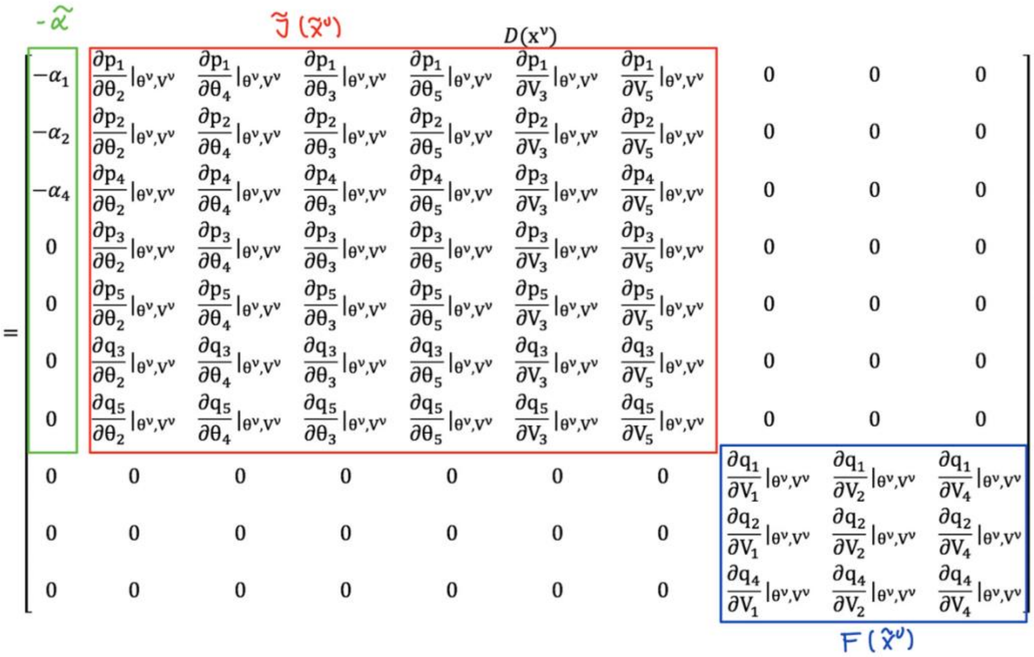
It can be seen that the results obtained are very similar, but not identical. This difference between them could be reduced by decreasing the constant that has been used as a threshold to stop the iterative process of both methods.

On the other hand, it is shown that the Jacobian matrix used in the Quasi-Newton-Raphson method (Figure 6) can be divided in turn into various smaller matrices fulfilling the following:

$$\begin{bmatrix} -\tilde{\alpha} & \tilde{\mathrm{J}}(\tilde{\mathrm{x}}^{\nu}) & 0_{(2n-m)\times m} \\ 0_m & 0_{m\times(2n-m-1)} & \mathrm{F}(\tilde{\mathrm{x}}^{\nu}) \end{bmatrix} \begin{bmatrix} \Delta\xi^{\nu} \\ \widetilde{\Delta\mathrm{x}}^{\nu} \\ \widetilde{\Delta\mathrm{v}}^{\nu} \end{bmatrix} = -\begin{bmatrix} \tilde{\mathrm{f}}(\xi^{\nu}, \tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu}) \\ \tilde{\mathrm{f}}(\tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu}) \end{bmatrix}$$

$$\tilde{\mathrm{J}}(\tilde{\mathrm{x}}^{\nu})\widetilde{\Delta\mathrm{x}}^{\nu} = -\tilde{\mathrm{f}}(\xi^{\nu}, \tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu}) + \tilde{\alpha}\Delta\xi^{\nu}$$

$$\mathrm{F}(\tilde{\mathrm{x}}^{\nu})\widetilde{\Delta\mathrm{v}}^{\nu} = -\tilde{\mathrm{f}}(\tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu})$$



$$
\begin{array}{cc}
 & -\tilde{\alpha} \qquad\qquad \mathfrak{J}(\tilde{x}^{\nu}) \qquad\qquad D(x^{\nu}) \\
= & \left[\begin{array}{c|cccccc|ccc}
-\alpha_1 & \frac{\partial p_1}{\partial \theta_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_1}{\partial \theta_4}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_1}{\partial \theta_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_1}{\partial \theta_5}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_1}{\partial V_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_1}{\partial V_5}\big|_{\theta^\nu,v^\nu} & 0 & 0 & 0 \\
-\alpha_2 & \frac{\partial p_2}{\partial \theta_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_2}{\partial \theta_4}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_2}{\partial \theta_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_2}{\partial \theta_5}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_2}{\partial V_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_2}{\partial V_5}\big|_{\theta^\nu,v^\nu} & 0 & 0 & 0 \\
-\alpha_4 & \frac{\partial p_4}{\partial \theta_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_4}{\partial \theta_4}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_4}{\partial \theta_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_4}{\partial \theta_5}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_3}{\partial V_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_4}{\partial V_5}\big|_{\theta^\nu,v^\nu} & 0 & 0 & 0 \\
0 & \frac{\partial p_3}{\partial \theta_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_3}{\partial \theta_4}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_3}{\partial \theta_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_3}{\partial \theta_5}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_3}{\partial V_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_3}{\partial V_5}\big|_{\theta^\nu,v^\nu} & 0 & 0 & 0 \\
0 & \frac{\partial p_5}{\partial \theta_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_5}{\partial \theta_4}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_5}{\partial \theta_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_5}{\partial \theta_5}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_5}{\partial V_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial p_5}{\partial V_5}\big|_{\theta^\nu,v^\nu} & 0 & 0 & 0 \\
0 & \frac{\partial q_3}{\partial \theta_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_3}{\partial \theta_4}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_3}{\partial \theta_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_3}{\partial \theta_5}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_3}{\partial V_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_3}{\partial V_5}\big|_{\theta^\nu,v^\nu} & 0 & 0 & 0 \\
0 & \frac{\partial q_5}{\partial \theta_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_5}{\partial \theta_4}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_5}{\partial \theta_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_5}{\partial \theta_5}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_5}{\partial V_3}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_5}{\partial V_5}\big|_{\theta^\nu,v^\nu} & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial q_1}{\partial V_1}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_1}{\partial V_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_1}{\partial V_4}\big|_{\theta^\nu,v^\nu} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial q_2}{\partial V_1}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_2}{\partial V_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_2}{\partial V_4}\big|_{\theta^\nu,v^\nu} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial q_4}{\partial V_1}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_4}{\partial V_2}\big|_{\theta^\nu,v^\nu} & \frac{\partial q_4}{\partial V_4}\big|_{\theta^\nu,v^\nu}
\end{array}\right] \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad F(\tilde{x}^{\nu})
\end{array}
$$

Figure 6 – Jacobian Matrix for Quasi-Newton-Raphson Method

6. References

[1] A. D. Domínguez-García, “Power Flow,” class notes for ECE 476, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2021.

[2] A. D. Domínguez-García, “Numerical Solution of PF,” class notes for ECE 476, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2021.

[3] O. Ajala, A. Domínguez-García, N. Baeckeland, S. Dhople, “Uncovering the Kuramoto Model from Full-order Models of Grid-forming Inverter-based Power Networks,” *IEEE Conference on Decision and Control*, December 2021.

Contents

1. Introduction

There is no question that our society is highly reliant on electric power. Therefore, it is necessary to have a good understanding of the operation of the electrical network, which can be accomplished via the solution of the power flow problem. It is important to mention that the applications of power flow problem are essential instrument for the analysis, planning and design of electrical systems, as well as their operation and control.

The relations between the currents injected into buses and the voltages at all buses are linear [1]. But the power flow model, which describes the active and reactive power flow through the electrical grid, is nonlinear [2]. This makes it difficult to achieve an analytical solution to the power flow equations, so it is necessary to resort to numerical computation methods. The methods to be discussed in this thesis are Newton-Raphson method and Quasi-Newton-Raphson method [3].

The objective of the power flow problem is to obtain the bus voltages, i.e., the complete phasor which will be separated in voltage magnitude and voltage phase angle. With these values, the active and reactive power injected in each bus can be calculated and it is possible as well to determine the behavior of the system under steady-state conditions. This is actually required to plan the extension of the existing system and to establish contingency plans in case of failure of network elements.

Moreover, the changeover from synchronous power generators to grid-forming inverter-interfaced generating resources has modified the operational characteristics of the power network [4]. These new generating resources need control systems that are different than those traditionally used, so it is necessary to modify the traditional power flow model, by introducing the particular models governing the steady-state behavior of the grid-forming inverters [4].

My involvement in this project is the implementation of Newton-Raphson method and Quasi-Newton-Raphson method using Python as programming language. In order to achieve it, the first step consisted of the formulation of the updated power flow equations, followed by the development of the Newton-Raphson method with Python, and based on the results obtained with this method, the Quasi-Newton-Raphson method was implemented.

The code developed for both methods is valid for any electrical system in which buses are differentiated into generators and loads depending on their behavior, which will be described later. For analysis purposes, certain specific data are necessary, such as the initial value of the voltage and angle of each bus, reference active and reactive power for generator buses, reference voltage for generator buses, active and reactive power demanded by each bus and finally some participation factor in the active and reactive power of each generator bus. Besides the relationship between the buses, provided through the transmission lines data, that allows the generation of the admittance matrix, which will also be discussed later.

Regarding numerical methods, the Newton Raphson is the most commonly used. It is also the most complete and consequently the most complex to solve. Therefore, simplifications are made by making assumptions about the electrical system, such as short and lossless lines. This leads to the development of Quasi-Newton methods, which are slightly simpler but require a larger number of iterations, so they are slower in finding a suitable solution. Thus, the same solution is obtained using any of the methods described and implemented in this project.

2. Literature Review: The Power Flow Problem

2.1 The Power Flow Problem

A concise summary of the formulation of the power flow problem is presented in this section. It is essential to start with the network admittance matrix, which represents the most fundamental relation in the power flow, following with the standard power flow problem and therefore, finishing with the power flow problem used during the development of the project.

2.1.1 The Admittance Matrix

In order to design a model of the terminal behavior of a power network it is necessary to establish the relations between the currents injected into the terminals, also referred to as buses, and the voltages at all buses [1]. To do so, Kirchhoff's Current Law is applied to each bus and a vector for the voltages ($\overline{\mathrm{V}}$) and another for the currents injected ($\overline{\mathrm{I}}$) is constructed. Then, the admittance matrix ($\overline{\mathrm{Y}}$) can be defined, which is described by the following relation [1]:

$$\overline{\mathrm{I}}_{\mathrm{i}} = \sum_{k=1}^{n} \overline{\mathrm{Y}}_{\mathrm{i,k}} \cdot \overline{\mathrm{V}}_{\mathrm{k}},\ \mathrm{i}=1,\ 2,\ \ldots,\ \mathrm{n} \qquad (1)$$

This equation corresponds to a general system with n buses, being $\overline{\mathrm{I}}_{\mathrm{i}}$ the phasor associated with the current injected into the system via bus i, and $\overline{\mathrm{V}}_{\mathrm{i}}$ is the phasor associated with the voltage at bus i [1]. The elements of the matrix that correspond to $\overline{\mathrm{Y}}_{\mathrm{i,k}}$, i ≠ k, represent the series admittances of all the components connected between bus i and bus k, and the elements of the matrix that correspond to $\overline{\mathrm{Y}}_{\mathrm{i,i}}$ represent the series admittance of all the components connected to bus i, half of the shunt admittance of all the transmission lines connect to bus i, and also the shunt admittance of transformers whose secondary side is connected to bus i [1]. Thus, the admittance matrix defines the relationship between the voltage and the current for each bus, becoming a complex and symmetric matrix that can be expressed as follows [1]:

$$\overline{Y} = G + jB, \tag{2}$$

where G is the real part of the matrix, also known as the conductance matrix, and B is the imaginary part of the matrix, also referred to as the susceptance matrix [1].

2.1.2 The Standard Power Flow Problem: Single Slack Bus

Consider a power system with n buses. Assuming that the first m buses are generators and the remainder n-m buses are loads. $\overline{V}_i = V_i e^{j\theta_i}$, where V_i denotes the voltage magnitude of the phasor and θ_i denotes the phase angle, and $\overline{I}_i = I_i e^{j\gamma_i}$, where I_i denotes the current magnitude of the phasor and γ_i denotes the phase angle. Let $P_i^D$and Q_i^D denote the active and reactive power consumed by the load buses, let V_i^{ref} denote the reference value passed to the voltage regulator of the generator buses, and let P_i^{ref} denote the active power reference command for the generator buses [2]. Then, the formulation of the power flow problem can be developed.

The complex power for each bus is defined by the next equation [2]:

$$\overline{S}_i = P_i + jQ_i = \overline{V}_i \cdot \overline{I}_i^* = \overline{V}_i \cdot \left(\sum_{k=1}^{n} \overline{Y}_{i,k} \cdot \overline{V}_k\right)^* \tag{3}$$

Then, using polar coordinates for $\overline{V}_i$, cartesian coordinates for $\overline{Y}_{i,k}$ and after operating and separating real and imaginary parts, we obtain [2]:

$$P_i = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \cos(\theta_i - \theta_k) + B_{i,k} \cdot \sin(\theta_i - \theta_k) \tag{4}$$

$$Q_i = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \sin(\theta_i - \theta_k) - B_{i,k} \cdot \cos(\theta_i - \theta_k), \tag{5}$$

which represent the power flow balance equations.

The power flow formulation assumes the use of synchronous generators and grid-forming inverters, where voltages are controlled by a voltage regulator and then they are fixed to reference voltages, V_i^{ref}. It also assumes that the active power generated is equal to the sum of the individual active power references P_i^{ref} and a term that allocates the mismatch between the active power demanded and the sum of the reference commands of the active power (P_i^{ref}) according to a coefficient α_i , also known as participation factor [5].

The participation factors (whose sum is equal to 1: $\sum_{i=1}^{m} \alpha_i = 1$) depend on the droop coefficient of the generator speed governor and the droop coefficient of the inverter-interfaced generator frequency droop control [5]. Let ξ denote a slack variable, which represents the mismatch between the active power demanded and the sum of the individual active power references. The standard power flow problem assumes that the first generator picks all the slack, which is known as the slack bus, so $\alpha_1 = 1$ and $\alpha_i = 0$, i = 2, …, m [2]. Then, the equation for the active power flow for the generator buses is defined as follows [2]:

$$P_i = P_i^{ref} + \alpha_i \cdot \xi = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \cos(\theta_i - \theta_k) + B_{i,k} \cdot \sin(\theta_i - \theta_k)), \qquad (6)$$

where $V_i = V_i^{ref}$, $\alpha_1 = 1$, $\alpha_i = 0$, i = 2, …, m and because of phase ambiguity, the angle of the first generator will be fixed to 0, i.e., $\theta_1 = 0$ [2]. These generator buses are also called PV buses due to having fixed the voltage magnitude and their active power injection as it is defined in equation (6).

For the load terminals, the power injections for each bus can be described as follows [2]:

$$P_i = -P_i^D \qquad (7)$$

$$Q_i = -Q_i^D \qquad (8)$$

These load buses are also known as PQ buses because of their fixed active and reactive power injections as described in equations (7) and (8). Then, the complete standard power flow model can be obtained [2]:

$$P_i^{ref} + \alpha_i \cdot \xi = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \cos(\theta_i - \theta_k) + B_{i,k} \cdot \sin(\theta_i - \theta_k)), \ i = 1, 2, \dots, m \quad (9)$$

$$Q_i = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \sin(\theta_i - \theta_k) - B_{i,k} \cdot \cos(\theta_i - \theta_k)), \ i = 1, 2, \dots, m, \quad (10)$$

$$-P_i^D = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \cos(\theta_i - \theta_k) + B_{i,k} \cdot \sin(\theta_i - \theta_k)), \ i = m+1, m+\ 2, \dots, n, \quad (11)$$

$$-Q_i^D = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \sin(\theta_i - \theta_k) - B_{i,k} \cdot \cos(\theta_i - \theta_k)), \ i = m+1, m+2, \dots, n, \quad (12)$$

where $V_i = V_i^{ref}$ for $i = 1, 2, \dots, m$, $\alpha_1 = 1$, $\alpha_i = 0$, i = 2, …, m and $\theta_1 = 0$.

This system is constituted by a set of 2n equations that needs to be solved for the unknowns: bus voltage magnitudes at PQ buses (n-m unknowns), bus voltage phase angles at all buses except for bus 1 (n-1 unknowns), the slack variable (1 unknowns) and the reactive power injections at PV buses (m unknowns), which represent 2n unknown variables that coincide with the total number of equations [2].

To solve for the unknown variables, the equations that should be considered are (9) for PV buses, (11) and (12), where $V_i = V_i^{ref}$ for $i = \ 2, \dots, m$, $\alpha_i = 0$, i = 2, …, m and $\theta_1 = 0$, which results in a closed set of 2n-m-1 equations. Note that the active power equation of the generator that takes over the complete slack is not taken into account, since it is no longer a generators bus, but the slack bus, so it will be calculated separately, since all the variables of this equation are fixed. For this reduced system of equations, the number of unknown variables is 2n-m-1: θ_i, i = 2, …, n (n-1 unknowns) and V_i, i = m+1, m+2, …, n (n-m unknowns). Thus, equation (10) will be solved to obtain all other unknown variables (Q_i, i= 1, 2, …, m) [2].

2.1.3 Distributed Slack Bus

A more realistic model for depicting a power system´s behavior is considering that the slack is distributed between all the generator buses. The power system taken into account is the one presented in Section 2.1.2, with certain additions and modifications.

Since the slack is considered distributed between the generator buses, the participation factors are defined between 0 and 1. Thus, the active power for the generator buses follows equation (6), being also known as PV buses as in the standard problem. The load buses are defined according to the standard power flow problem equations (7) and (8), so they are described as PQ buses. Therefore, the power balance equations will be (9), (10), (11) and (12), where $V_i = V_i^{ref}$ for $i = 1, 2, \dots, m$ and $\theta_1 = 0$ [2].

The resulting system is formed by a set of 2n equations that must be solved for the unknowns: bus voltage magnitudes at PQ buses (n-m unknowns), bus voltage phase angles at all buses except for bus 1 (n-1 unknowns), the slack variable (1 unknowns) and the reactive power injections at PV buses (m unknowns), which represent 2n unknown variables that coincide with the total number of equations [2].

In order to solve for the unknown variables, the equations that should be solved jointly are equations (9), (11) and (12) where $V_i = V_i^{ref}$ for $i = 1, 2, \dots, m$ and $\theta_1 = 0$. The number of unknown variables is 2n-m: ξ (1 unknowns), θ_i, i = 2, …, n (n-1 unknowns) and V_i, i = m+1, m+2, …, n (n-m unknowns), which matches the number of equations of the selected closed set of equations, i.e, (9), (11) and (12). Finally, (10) will be solved to obtain all other unknown variables (Q_i, i= 1, 2, …, m) [2].

2.1.4 Distributed Slack Bus and Voltage Droop Control

The emerging application of power generating sources based on grid-forming inverters has introduced changes in electric power systems [4]. These different characteristics affect the active and reactive power injections due to the droop control, and can be described using the Kuramoto Model for a network of nonlinear n coupled oscillators, which is represented as follows [4]:

$$\dot{\delta}_i = \omega_i - \omega_0 - \sum_{j=1}^{n} a_{ij} \sin\left(\delta_i - \delta_j\right), i = 1, 2, \ldots, n, \tag{13}$$

where ω_i is the natural frequency of the oscillator i, ω_0 is the nominal frequency of the oscillators, $a_{ij} \geq 0$ represents the coupling strength between oscillator i and j ($a_{ij} = a_{ji}$) and δ_i denotes the phase of oscillator i relative to a reference frame rotating at frequency ω_0 [4].

The droop control dynamics can be represented by [4]:

$$\omega = \omega_0 + \frac{1}{\mathrm{df}} \mathbb{e}_1^T \mathrm{T}\left(\psi - \frac{\pi}{2}\right)(s^* - s_m), \tag{14}$$

$$e^* = e_0 + \frac{1}{\mathrm{dv}} \mathbb{e}_2^T \mathrm{T}\left(\psi - \frac{\pi}{2}\right)(s^* - s_m), \tag{15}$$

$$\frac{1}{\omega_c}\dot{s}_m = -s_m + s, \tag{16}$$

where $\mathbb{e}_i$ is the standard basis vector with one in the i-th position, df and dv are the droop coefficients for frequency and voltage respectively, T is representing: $\mathrm{T}(x) = [\cos(x), \sin(x)]^T$, ψ is the rotation angle which in steady − state is $\frac{\pi}{2}$, $s^* = [p^*, q^*]$ denoting the reference commands for active and reactive power, $s_m = [p_m, q_m]$ representing the measured values for active and reactive power, e^* is the voltage magnitude reference, e_0

is the nominal voltage of the inverter, and ω_c is the cut-off frequency of a low-pass measurement filter [4].

Simplifying (14) and (15) to return to steady-state analysis and considering that P and Q are not decoupled, the following equations are developed [6]:

$$\omega = \omega_0 - m_p(P_G - Q_G), \tag{17}$$

$$V = V_0 - n_q(P_G + Q_G), \tag{18}$$

where m_p and n_q denote the coefficients of frequency and voltage droop respectively, and P_G and Q_G represent the power injections. However, the assumption that active and reactive power injections are decoupled will be considered [6]:

$$P_G = P_{ref} + \frac{(\omega_{ref} - \omega)}{m_p}, \tag{19}$$

$$Q_G = Q_{ref} + \frac{(V_{ref} - V)}{n_q}, \tag{20}$$

m_p denotes the inverse of the active participation factor α, and n_q is the inverse of the reactive participation factor called β.

For this project, the power system to be considered is also the one explained in Section 2.1.2, with the addition of Q_i^{ref} that denotes the reactive power reference command for the generator buses. Furthermore, voltages of generator buses are no longer fixed to reference voltages, V_i^{ref}, thus, this generator buses will no further be handled as PV buses.

We still use ξ to denote the slack variable, and the angle of the first generator will be fixed to 0, i.e., $\theta_1 = 0$. Thus, the active and reactive power equations for the generator buses are defined introducing (17) and (18):

$$P_i = P_i^{ref} + \alpha_i \cdot \xi \tag{21}$$

$$Q_i = Q_i^{ref} + \beta_i \cdot (V_i^{ref} - V_i) \tag{22}$$

These power injections for the generator buses are already fixed, so they change from being PV buses to PQ buses. The power injections for the load buses are the same as in the standard power flow following equations (7) and (8), thus they are still PQ buses. To facilitate the difference between all PQ buses, they will be separated into generator and load buses. Then, the complete and updated power flow equations are as follows [2] and [6]:

$$P_i^{ref} + \alpha_i \cdot \xi = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \cos(\theta_i - \theta_k) + B_{i,k} \cdot \sin(\theta_i - \theta_k)),\ \ i = 1, 2, \dots, m, \tag{23}$$

$$Q_i^{ref} + \beta_i \cdot (V_i^{ref} - V_i) = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \sin(\theta_i - \theta_k) - B_{i,k} \cdot \cos(\theta_i - \theta_k)),\ \ i = 1, 2, \dots, m, \tag{24}$$

$$-P_i^D = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \cos(\theta_i - \theta_k) + B_{i,k} \cdot \sin(\theta_i - \theta_k)),\ \ i = m+1, m+\ 2, \dots, n, \tag{25}$$

$$-Q_i^D = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \sin(\theta_i - \theta_k) - B_{i,k} \cdot \cos(\theta_i - \theta_k)),\ \ i = m+1, m+2, \dots, n, \tag{26}$$

where $\theta_1 = 0$.

The previous system of equations is composed of 2n equations that needs to be solved for the unknowns: bus voltage magnitudes at all buses (n unknowns), bus voltage phase angles at all buses except for bus 1 (n-1 unknowns), and the slack variable (1 unknowns), which make 2n unknown variables that is equal to the total number of equations. In order to solve for the unknown variables, the equations needed are all those described above, with no possibility of reducing them to a smaller closed set of equations.

2.2 Numerical Solution of the Power Flow Problem

The power flow problem equations for either of the above cases do not have an easy to identify analytical solution, so there is the necessity of using numerical computation methods. In this section, the methods utilized during the project will be described.

2.2.1 Newton-Raphson's Method for the Standard Power Flow

Recall the n-bus system setting discussed in Section 2.1.2. The standard power flow problem needs to be solved for ξ, θ_i, for all buses but the first generator, V_i, for PQ buses and Q_i, for PV buses. Then, given $P_i^{ref}$and V_i^{ref} for PV buses, $\theta_1 = 0$, P_i^D and Q_i^D for PQ buses, $\alpha_1 = 1$ and $\alpha_i = 0$, for PV buses, the objective is to solve for the unknowns [2].

The equations that must be taken into account are the active power for PV buses (9), and the active (11) and reactive (12) power equations for PQ buses [2]. This fashion, the reduced system can be solved for the unknown variables involved θ_i, i = 2, …, n and V_i, i = m+1, m+2, …, n.

First, the active and reactive power equations can be defined as follows [3]:

$$p_i(\theta, V) = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \cos(\theta_i - \theta_k) + B_{i,k} \cdot \sin(\theta_i - \theta_k) \tag{27}$$

$$q_i(\theta, V) = \sum_{k=1}^{n} V_i \cdot V_k \cdot (G_{i,k} \cdot \sin(\theta_i - \theta_k) - B_{i,k} \cdot \cos(\theta_i - \theta_k) \tag{28}$$

The active and reactive power balance equations (9), (11) and (12) must be rewritten using (27) and (28) as described [3]:

$$p_i(\theta, V) - P_i^{ref} - \alpha_i \cdot \xi = 0, i = 2\ldots, m \tag{29}$$

$$p_i(\theta, V) + P_i^D = 0, i = m+1\ldots, n \tag{30}$$

$$q_i(\theta, V) + Q_i^D = 0, i= m+1, \ldots, n, \tag{31}$$

where $\theta_1 = 0$, $V_i = V_i^{ref}$ for $i = 1, 2, \ldots, m$, $\alpha_1 = 1$ and $\alpha_i = 0, i = 2, \ldots, m$.

Let x be a vector containing the following unknowns [3]:

$$x = \begin{bmatrix} \theta \\ V \end{bmatrix},$$

where $\theta = [\theta_2 \quad \ldots \quad \theta_n]^T$ and $V = [V_{m+1} \quad \ldots \quad V_n]^T$. Subsequently, Newton´s method iteration for (29), (30) and (31) is [3]:

$$x^{\nu+1} = x^{\nu} - \left(J(x^{\nu})\right)^{-1} \cdot f(x^{\nu}), \tag{32}$$

where $J(x^{\nu})$ is the Jacobian matrix built as follows [3]:

$$J(x^{\nu}) = \begin{bmatrix} \frac{\partial p_2}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_2}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_2}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_2}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_m}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_m}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_m}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_m}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \frac{\partial p_{m+1}}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_{m+1}}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_{m+1}}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_{m+1}}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_n}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_n}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_n}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_n}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \frac{\partial q_{m+1}}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_{m+1}}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_{m+1}}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_{m+1}}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial q_n}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_n}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_n}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_n}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \end{bmatrix} \tag{33}$$

$$=: \begin{bmatrix} H(x^{\nu}) & N(x^{\nu}) \\ K(x^{\nu}) & L(x^{\nu}) \end{bmatrix},$$

where $H(x^{\nu}) = \frac{\partial p_i}{\partial \theta_j}|_{\theta^{\nu},V^{\nu}}$, i = 2, …, n, j = 2, …, n, $N(x^{\nu}) = \frac{\partial p_i}{\partial V_j}|_{\theta^{\nu},V^{\nu}}$, i = 2, …, n, , j = m+1, …, n, $K(x^{\nu}) = \frac{\partial q_i}{\partial \theta_j}|_{\theta^{\nu},V^{\nu}}$, i = m+1, …, n, j = 2, …, n and $L(x^{\nu}) = \frac{\partial q_i}{\partial V_j}|_{\theta^{\nu},V^{\nu}}$ i = m+1, …, n, j = m+1, …, n. Moreover, $f(x^{\nu})$ corresponds to [3]:D

$$f(x^{\nu}) = \begin{bmatrix} p_2(\theta^{\nu}, V^{\nu}) - P_2^{ref} \\ \vdots \\ p_m(\theta^{\nu}, V^{\nu}) - P_m^{ref} \\ p_{m+1}(\theta^{\nu}, V^{\nu}) + P_{m+1}^{D} \\ \vdots \\ p_n(\theta^{\nu}, V^{\nu}) + P_n^{D} \\ q_{m+1}(\theta^{\nu}, V^{\nu}) + Q_{m+1}^{D} \\ \vdots \\ q_n(\theta^{\nu}, V^{\nu}) + Q_n^{D} \end{bmatrix} \tag{34}$$

The solution of the power flow problem will depend on the choice of the initial values for the unknowns to start with iterations. These values are selected to comply with the values that the system would take in normal conditions, which match the so-called flat voltage profile ($\theta_i = 0$ and $V_i = 0$) for all the buses [3]. Then, the initial vector $x^0 = [\theta^0 \quad V^0]^T$ where $\theta_i^0 = 0$ for i = 2, …, n and $V_i^0 = 1$ for i = m+1, …, n. While iterating, the inverse of the Jacobian matrix is not calculated, since it is computationally expensive. In turn, the equation to be solved is the following [3]:

$$J(x^{\nu}) \cdot \Delta x^{\nu} = -f(x^{\nu}), \tag{35}$$

to obtain the change in x^{ν}, i.e., Δx^{ν}. Consequently, the updates of the solution will be calculated as follows [3]:

$$\mathrm{x}^{\nu+1} = \mathrm{x}^{\nu} + \Delta x^{\nu} \tag{36}$$

The algorithm finishes when $||\mathrm{x}^{\nu+1} - \mathrm{x}^{\nu}|| < \varepsilon$ for some ε small [3], approximately $1\mathrm{e}^{-3}$.

If the slack is distributed as for the system in Section 2.1.3, the iteration method is the same as the one described above with slight differences. For this case, the participation factors are defined between 0 and 1, thus, equations (29), (30) and (31) comply with $\theta_1 = 0$ and $V_i = V_i^{ref}$ for $i = 1, 2, \dots, m$ [2]. The vector x containing the unknown variables will be defined [3]:

$$x = \begin{bmatrix} \xi \\ \theta \\ V \end{bmatrix},$$

where $\theta = [\theta_2 \quad \dots \quad \theta_n]^T$ and $V = [V_{m+1} \quad \dots \quad V_n]^T$. In addition, the Jacobian matrix is quite different [3]:

$$J(x^{\nu}) = \begin{bmatrix} \frac{\partial p_1}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_1}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_1}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_1}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_1}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_m}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_m}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_m}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_m}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_m}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \frac{\partial p_{m+1}}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_{m+1}}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_{m+1}}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_{m+1}}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_{m+1}}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_n}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_n}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_n}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_n}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_n}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \frac{\partial q_{m+1}}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_{m+1}}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_{m+1}}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_{m+1}}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_{m+1}}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial q_n}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_n}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_n}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_n}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_n}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \end{bmatrix} \tag{37}$$

For PV buses $\frac{\partial p_i}{\partial \xi}|_{\theta^{\nu},V^{\nu}} = -\alpha_i$, i = 1, 2, …, m and for PQ $\frac{\partial p_i}{\partial \xi}|_{\theta^{\nu},V^{\nu}} = 0$, i = m+1, …, n and $\frac{\partial q_i}{\partial \xi}|_{\theta^{\nu},V^{\nu}} = 0$, i = m+1, …, n [3]. And $f(x^{\nu})$ contains an additional column for the active power equation for the first generator bus [3]:

$$f(x^{\nu}) = \begin{bmatrix} p_1(\theta^{\nu}, V^{\nu}) - P_1^{ref} - \alpha_1 \cdot \xi \\ \vdots \\ p_m(\theta^{\nu}, V^{\nu}) - P_m^{ref} - \alpha_m \cdot \xi \\ p_{m+1}(\theta^{\nu}, V^{\nu}) + P_{m+1}^{D} \\ \vdots \\ p_n(\theta^{\nu}, V^{\nu}) + P_n^{D} \\ q_{m+1}(\theta^{\nu}, V^{\nu}) + Q_{m+1}^{D} \\ \vdots \\ q_n(\theta^{\nu}, V^{\nu}) + Q_n^{D} \end{bmatrix} \tag{38}$$

The initial values chosen for the first iteration must follow the flat-voltage profile as above, additionally, the slack variable ξ initial point will be set to 0 considering that the losses are typically small [3]. Then, the initial vector $x^0 = [\xi^0 \quad \theta^0 \quad V^0]^T$, where $\xi^0 = 0$, $\theta_i^0 = 0$ for i = 2, …, n and $V_i^0 = 1$ for i = m+1, …, n [3]. The remaining resolution method is the same as described previously using (35) and (36).

2.2.2 Newton-Raphson's Method: Distributed Slack Bus and Voltage Droop Control

For this project, the power system to be studied is the one presented in Section 2.1.4. This updated system needs to be solved for the slack ξ, bus voltage phase angles at all buses but bus 1 (θ_i for i = 2, …, n) and bus voltage magnitude at all buses (V_i for i = 1, …, n). As the system of equations composed by (23), (24), (25) and (26) cannot be reduced to a smaller closed set of equations, all those equations will be involved in the iteration process. Let (27) and (28) define the active and reactive power balance equations. Then, (23), (24), (25) and (26) can be rewritten as the following [3], [6]:

$$\mathrm{p_i}(\theta, \mathrm{V}) - \mathrm{P_i^{ref}} - \alpha_\mathrm{i} \cdot \xi = 0,\ \mathrm{i} = 1\ldots,\mathrm{m} \tag{39}$$

$$\mathrm{q_i}(\theta, \mathrm{V}) - \mathrm{Q_i^{ref}} - \beta_i \cdot \left(\mathrm{V_i^{ref}} - \mathrm{V_i}\right) = 0,\ \mathrm{i} = 1\ldots,\mathrm{m} \tag{40}$$

$$\mathrm{p_i}(\theta, \mathrm{V}) + \mathrm{P_i^D} = 0,\ \mathrm{i} = \mathrm{m}{+}1\ldots,\mathrm{n} \tag{41}$$

$$\mathrm{q_i}(\theta, \mathrm{V}) + \mathrm{Q_i^D} = 0,\ \mathrm{i} = \mathrm{m}{+}1\ldots,\mathrm{n} \tag{42}$$

where $\theta_1 = 0$. Let x contain the unknown variables [3]:

$$x = \begin{bmatrix} \xi \\ \theta \\ \mathrm{V} \end{bmatrix},$$

where $\theta = [\theta_2 \quad \ldots \quad \theta_n]^T$ and $V = [V_1 \quad \ldots \quad V_n]^T$. The Jacobian matrix can be defined as follows [3], [6]:

$$J(x^{\nu}) = \begin{bmatrix}
\frac{\partial p_1}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_1}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_1}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_1}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_1}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial p_m}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_m}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_m}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_m}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_m}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\
\frac{\partial p_{m+1}}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_{m+1}}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_{m+1}}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_{m+1}}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_{m+1}}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial p_n}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_n}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_n}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_n}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_n}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\
\frac{\partial q_1}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_1}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_1}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_1}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_1}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial q_m}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_m}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_m}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_m}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_m}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\
\frac{\partial q_{m+1}}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_{m+1}}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_{m+1}}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_{m+1}}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_{m+1}}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial q_n}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_n}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_n}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_n}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_n}{\partial V_n}|_{\theta^{\nu},V^{\nu}}
\end{bmatrix} \tag{43}$$

For generator buses $\frac{\partial p_i}{\partial \xi}|_{\theta^{\nu},V^{\nu}} = -\alpha_i$ and $\frac{\partial q_i}{\partial \xi}|_{\theta^{\nu},V^{\nu}} = 0$, $i = 1, \ldots, m$, and for load buses $\frac{\partial p_i}{\partial \xi}|_{\theta^{\nu},V^{\nu}} = 0$ and $\frac{\partial q_i}{\partial \xi}|_{\theta^{\nu},V^{\nu}} = 0$, $i = m+1, \ldots, n$ [3]. In this case, $f(x^{\nu})$ will consist of all the active and reactive power equations for all the buses [3], [4]:

$$f(x^{\nu}) = \begin{bmatrix}
p_1(\theta^{\nu},\ V^{\nu}) - P_1^{ref} - \alpha_1 \cdot \xi \\
\vdots \\
p_m(\theta^{\nu},\ V^{\nu}) - P_m^{ref} - \alpha_m \cdot \xi \\
p_{m+1}(\theta^{\nu},\ V^{\nu}) + P_{m+1}^{D} \\
\vdots \\
p_n(\theta^{\nu},\ V^{\nu}) + P_n^{D} \\
q_1(\theta^{\nu},\ V^{\nu}) - Q_1^{ref} - \beta_1 \cdot \left(V_1^{ref} - V_1\right) \\
\vdots \\
q_m(\theta^{\nu},\ V^{\nu}) - Q_m^{ref} - \beta_m \cdot \left(V_m^{ref} - V_m\right) \\
q_{m+1}(\theta^{\nu},\ V^{\nu}) + Q_{m+1}^{D} \\
\vdots \\
q_n(\theta^{\nu},\ V^{\nu}) + Q_n^{D}
\end{bmatrix} \tag{44}$$

The initial values will be chosen as in the previous section, following the flat-voltage profile [3]. Then, the initial vector $x^0 = [\xi^0 \quad \theta^0 \quad V^0]^T$, where $\xi^0 = 0$, $\theta_i^0 = 0$ for i = 2, …, n and $V_i^0 = 1$ for i = m+1, …, n [3]. The rest of the resolution method is the same as described above using (35) and (36).

2.2.3 Quasi-Newton-Raphson's Method for the Updated Power Flow

In this section, the power system considered is the one explained in Section 2.1.4, and the resolution method utilized is the one described in Section 2.2.2. However, there exist slight differences with respect to Newton-Raphson´s method in order to demonstrate that the Jacobian matrix from the standard power flow problem (33) with some additions and some simplifications can be used to solve this power flow problem. The equations that need to be solved are (23), (24), (25) and (26), and must be considered as described in (39), (40), (41) and (42) where $\theta_1 = 0$.

The main differences with Newton-Raphson´s method are the rearrangement of the mismatch function and the Jacobian matrix, and the adjustment of several sections of the matrix to cero. The reorganization of the mismatch consists of placing first the active power equation for generator buses, then that for the loads, followed by the reactive power equation for load buses, and finally that for the generators. Thus, the mismatch is represented as follows [3], [6]:

$$f(x^\nu) = \begin{bmatrix} p_1(\theta^\nu, V^\nu) - P_1^{ref} - \alpha_1 \cdot \xi \\ \vdots \\ p_m(\theta^\nu, V^\nu) - P_m^{ref} - \alpha_m \cdot \xi \\ p_{m+1}(\theta^\nu, V^\nu) + P_{m+1}^D \\ \vdots \\ p_n(\theta^\nu, V^\nu) + P_n^D \\ q_{m+1}(\theta^\nu, V^\nu) + Q_{m+1}^D \\ \vdots \\ q_n(\theta^\nu, V^\nu) + Q_n^D \\ q_1(\theta^\nu, V^\nu) - Q_1^{ref} - \beta_1 \cdot (V_1^{ref} - V_1) \\ \vdots \\ q_m(\theta^\nu, V^\nu) - Q_m^{ref} - \beta_m \cdot (V_m^{ref} - V_m) \end{bmatrix} \qquad (45)$$

The resulting Jacobian matrix called $D(x^v)$ is reordered following the rearrangement of the mismatch, and also reorganizing the derivatives with respect to V: first the load ones and following the generators. The resulting Jacobian matrix is the following [3], [6]:

$$D(x^v) = \begin{bmatrix}
\frac{\partial p_1}{\partial \xi}|_{\theta^v,v^v} & \frac{\partial p_1}{\partial \theta_2}|_{\theta^v,v^v} & \cdots & \frac{\partial p_1}{\partial \theta_m}|_{\theta^v,v^v} & \frac{\partial p_1}{\partial \theta_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_1}{\partial \theta_n}|_{\theta^v,v^v} & \frac{\partial p_1}{\partial V_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_1}{\partial V_n}|_{\theta^v,v^v} & \frac{\partial p_1}{\partial V_1}|_{\theta^v,v^v} & \cdots & \frac{\partial p_1}{\partial V_m}|_{\theta^v,v^v} \\
\frac{\partial p_2}{\partial \xi}|_{\theta^v,v^v} & \frac{\partial p_2}{\partial \theta_2}|_{\theta^v,v^v} & \cdots & \frac{\partial p_2}{\partial \theta_m}|_{\theta^v,v^v} & \frac{\partial p_2}{\partial \theta_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_2}{\partial \theta_n}|_{\theta^v,v^v} & \frac{\partial p_2}{\partial V_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_2}{\partial V_n}|_{\theta^v,v^v} & \frac{\partial p_2}{\partial V_1}|_{\theta^v,v^v} & \cdots & \frac{\partial p_2}{\partial V_m}|_{\theta^v,v^v} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial p_m}{\partial \xi}|_{\theta^v,v^v} & \frac{\partial p_m}{\partial \theta_2}|_{\theta^v,v^v} & \cdots & \frac{\partial p_m}{\partial \theta_m}|_{\theta^v,v^v} & \frac{\partial p_m}{\partial \theta_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_m}{\partial \theta_n}|_{\theta^v,v^v} & \frac{\partial p_m}{\partial V_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_m}{\partial V_n}|_{\theta^v,v^v} & \frac{\partial p_m}{\partial V_1}|_{\theta^v,v^v} & \cdots & \frac{\partial p_m}{\partial V_m}|_{\theta^v,v^v} \\
\frac{\partial p_{m+1}}{\partial \xi}|_{\theta^v,v^v} & \frac{\partial p_{m+1}}{\partial \theta_2}|_{\theta^v,v^v} & \cdots & \frac{\partial p_{m+1}}{\partial \theta_m}|_{\theta^v,v^v} \cdots & \frac{\partial p_{m+1}}{\partial \theta_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_{m+1}}{\partial \theta_n}|_{\theta^v,v^v} & \frac{\partial p_{m+1}}{\partial V_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_{m+1}}{\partial V_n}|_{\theta^v,v^v} & \frac{\partial p_{m+1}}{\partial V_1}|_{\theta^v,v^v} & \cdots & \frac{\partial p_{m+1}}{\partial V_m}|_{\theta^v,v^v} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial p_n}{\partial \xi}|_{\theta^v,v^v} & \frac{\partial p_n}{\partial \theta_2}|_{\theta^v,v^v} & \cdots & \frac{\partial p_n}{\partial \theta_m}|_{\theta^v,v^v} & \frac{\partial p_n}{\partial \theta_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_n}{\partial \theta_n}|_{\theta^v,v^v} & \frac{\partial p_n}{\partial V_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial p_n}{\partial V_n}|_{\theta^v,v^v} & \frac{\partial p_n}{\partial V_1}|_{\theta^v,v^v} & \cdots & \frac{\partial p_n}{\partial V_m}|_{\theta^v,v^v} \\
\frac{\partial q_{m+1}}{\partial \xi}|_{\theta^v,v^v} & \frac{\partial q_{m+1}}{\partial \theta_2}|_{\theta^v,v^v} & \cdots & \frac{\partial q_{m+1}}{\partial \theta_m}|_{\theta^v,v^v} & \frac{\partial q_{m+1}}{\partial \theta_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial q_{m+1}}{\partial \theta_n}|_{\theta^v,v^v} & \frac{\partial q_{m+1}}{\partial V_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial q_{m+1}}{\partial V_n}|_{\theta^v,v^v} & \frac{\partial q_{m+1}}{\partial V_1}|_{\theta^v,v^v} & \cdots & \frac{\partial q_{m+1}}{\partial V_m}|_{\theta^v,v^v} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial q_n}{\partial \xi}|_{\theta^v,v^v} & \frac{\partial q_n}{\partial \theta_2}|_{\theta^v,v^v} & \cdots & \frac{\partial q_n}{\partial \theta_m}|_{\theta^v,v^v} & \frac{\partial q_n}{\partial \theta_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial q_n}{\partial \theta_n}|_{\theta^v,v^v} & \frac{\partial q_n}{\partial V_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial q_n}{\partial V_n}|_{\theta^v,v^v} & \frac{\partial q_n}{\partial V_1}|_{\theta^v,v^v} & \cdots & \frac{\partial q_n}{\partial V_m}|_{\theta^v,v^v} \\
\frac{\partial q_1}{\partial \xi}|_{\theta^v,v^v} & \frac{\partial q_1}{\partial \theta_2}|_{\theta^v,v^v} & \cdots & \frac{\partial q_1}{\partial \theta_m}|_{\theta^v,v^v} & \frac{\partial q_1}{\partial \theta_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial q_1}{\partial \theta_n}|_{\theta^v,v^v} & \frac{\partial q_1}{\partial V_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial q_1}{\partial V_n}|_{\theta^v,v^v} & \frac{\partial q_1}{\partial V_1}|_{\theta^v,v^v} & \cdots & \frac{\partial q_1}{\partial V_m}|_{\theta^v,v^v} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial q_m}{\partial \xi}|_{\theta^v,v^v} & \frac{\partial q_m}{\partial \theta_2}|_{\theta^v,v^v} & \cdots & \frac{\partial q_m}{\partial \theta_m}|_{\theta^v,v^v} & \frac{\partial q_m}{\partial \theta_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial q_m}{\partial \theta_n}|_{\theta^v,v^v} & \frac{\partial q_m}{\partial V_{m+1}}|_{\theta^v,v^v} & \cdots & \frac{\partial q_m}{\partial V_n}|_{\theta^v,v^v} & \frac{\partial q_m}{\partial V_1}|_{\theta^v,v^v} & \cdots & \frac{\partial q_m}{\partial V_m}|_{\theta^v,v^v}
\end{bmatrix} \tag{46}$$

Taking all this into account, the vector containing the unknown variables is defined [3]:

$$x = \begin{bmatrix} \xi \\ \theta \\ V \end{bmatrix},$$

$\theta = [\theta_2 \ \ldots \ \theta_{m}, \ \theta_{m+1} \ \ldots \ \theta_n]^T$ and $V = [V_{m+1} \ \ldots \ V_n, \ V_1 \ \ldots \ V_m]^T$. So, the results obtained from this iterative method are also rearranged. The phase angles obtained are first generators´ angles followed by loads´ angles, and the voltage magnitudes achieved are first loads' voltages followed by generators' voltages. This matrix is the same as (43) but in a different order, so the result obtained using it must be the same as the one with the Newton-Raphson´s method with the new order.

To accomplish Quasi-Newton-Raphson’s method some sections of $D(x^{\nu})$ must be set to cero: first, the derivatives with respect to ξ for all buses but the generators active power, the derivative of the reactive power for the generator buses but the ones with respect to the voltage of the generator buses, the derivative of the active power with respect to the voltage of the generator buses and the derivative of the reactive power of the load buses with respect to the voltage of the generator buses. Then, the resulting Jacobian matrix is described [3]:

$$D(x^{\nu}) = \begin{bmatrix}
\frac{\partial p_1}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_1}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_1}{\partial \theta_m}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_1}{\partial \theta_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_1}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_1}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_1}{\partial V_n}|_{\theta^{\nu},V^{\nu}} & 0 & \cdots & 0 \\
\frac{\partial p_2}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_2}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_2}{\partial \theta_m}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_2}{\partial \theta_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_2}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_2}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_2}{\partial V_n}|_{\theta^{\nu},V^{\nu}} & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial p_m}{\partial \xi}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_m}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_m}{\partial \theta_m}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_m}{\partial \theta_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_m}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_m}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_m}{\partial V_n}|_{\theta^{\nu},V^{\nu}} & 0 & \cdots & 0 \\
0 & \frac{\partial p_{m+1}}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_{m+1}}{\partial \theta_m}|_{\theta^{\nu},V^{\nu}} \cdots & \frac{\partial p_{m+1}}{\partial \theta_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_{m+1}}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_{m+1}}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_{m+1}}{\partial V_n}|_{\theta^{\nu},V^{\nu}} & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \frac{\partial p_n}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_n}{\partial \theta_m}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_n}{\partial \theta_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_n}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_n}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_n}{\partial V_n}|_{\theta^{\nu},V^{\nu}} & 0 & \cdots & 0 \\
0 & \frac{\partial q_{m+1}}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_{m+1}}{\partial \theta_m}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_{m+1}}{\partial \theta_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_{m+1}}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_{m+1}}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_{m+1}}{\partial V_n}|_{\theta^{\nu},V^{\nu}} & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \frac{\partial q_n}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_n}{\partial \theta_m}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_n}{\partial \theta_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_n}{\partial \theta_n}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_n}{\partial V_{m+1}}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_n}{\partial V_n}|_{\theta^{\nu},V^{\nu}} & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \frac{\partial q_1}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_1}{\partial V_m}|_{\theta^{\nu},V^{\nu}} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \frac{\partial q_m}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_m}{\partial V_m}|_{\theta^{\nu},V^{\nu}}
\end{bmatrix} \tag{47}$$

It can be observed that this matrix can be divided as follows: $-\tilde{\alpha}$ is a column vector of size (2n-m) containing de derivatives of the active power with respect of the slack and the derivatives of the reactive power of the load buses with respect to the slack. $\tilde{J}(\tilde{x}^{\nu})$ is a (2n-m) x (2n-m-1) matrix composed of the Jacobian matrix of the standard power flow problem (33) with an added row above containing the derivatives of the active power of the first generator with respect to the angles and to the loads’ voltages. $F(\tilde{x}^{\nu})$ is a m x m matrix containing the derivatives of the reactive power of the generator buses with respect to the generators’ voltage. Then, Newton-Raphson’s method can be rearranged as follows:

$$\begin{bmatrix} -\tilde{\alpha} & \tilde{\mathrm{J}}(\tilde{\mathrm{x}}^{\nu}) & 0_{(2n-m)\times m} \\ 0_m & 0_{m\times(2n-m-1)} & \mathrm{F}(\tilde{\mathrm{x}}^{\nu}) \end{bmatrix} \begin{bmatrix} \Delta\xi^{\nu} \\ \widetilde{\Delta \mathrm{x}}^{\nu} \\ \widetilde{\Delta \mathrm{v}}^{\nu} \end{bmatrix} = - \begin{bmatrix} \tilde{\mathrm{f}}(\xi^{\nu}, \tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu}) \\ \tilde{\mathrm{f}}(\tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu}) \end{bmatrix} \tag{48}$$

Moreover, Equation (35) can be divided in two:

$$\tilde{\mathrm{J}}(\tilde{\mathrm{x}}^{\nu})\widetilde{\Delta \mathrm{x}}^{\nu} = -\tilde{\mathrm{f}}(\xi^{\nu}, \tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu}) + \tilde{\alpha}\Delta\xi^{\nu} \tag{49}$$

$$\mathrm{F}(\tilde{\mathrm{x}}^{\nu})\widetilde{\Delta \mathrm{v}}^{\nu} = -\tilde{\mathrm{f}}(\tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu}) \tag{50}$$

where ξ^{ν} represents the slack variable, $\tilde{\mathrm{x}}^{\nu}$is a vector containing all the angles but the one of the first generator $\tilde{\mathrm{x}}^{\nu} = [\theta_2 \quad \dots \quad \theta_{m,} \quad \theta_{m+1} \quad \dots \quad \theta_n \quad V_{m+1} \quad \dots \quad V_n]^{T}$, which coincide with the unknown variables of the reduced set of equations of the standard power flow problem, and $\tilde{\mathrm{v}}^{\nu}$ that is a vector containing the voltages of the generator buses, $\tilde{\mathrm{v}}^{\nu} = [V_1 \quad \dots \quad V_m]^{T}$, that were fixed variables in the standard power flow problem but in this case must be solved. $\tilde{\mathrm{f}}(\xi^{\nu}, \tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu})$ is the first part of the mismatch vector (45), containing the mismatch of the active power of generator and load buses, followed by the mismatch of the reactive power of load buses. $\tilde{\mathrm{f}}(\tilde{\mathrm{x}}^{\nu}, \tilde{\mathrm{v}}^{\nu})$is the final part of the mismatch vector (45), consisting of the mismatch of the reactive power of generator buses.

Finally, Equation (36) can be rewritten:

$$\begin{bmatrix} \xi^{\nu+1} \\ \tilde{\mathrm{x}}^{\nu+1} \\ \tilde{\mathrm{v}}^{\nu+1} \end{bmatrix} = \begin{bmatrix} \xi^{\nu} \\ \tilde{\mathrm{x}}^{\nu} \\ \tilde{\mathrm{v}}^{\nu} \end{bmatrix} + \begin{bmatrix} \Delta\xi^{\nu} \\ \widetilde{\Delta \mathrm{x}}^{\nu} \\ \widetilde{\Delta \mathrm{v}}^{\nu} \end{bmatrix} \tag{51}$$

The iterative process finishes when $\|\mathrm{x}^{\nu+1} - \mathrm{x}^{\nu}\| < \varepsilon$, being ε very small. Considering: $\mathrm{x}^{\nu+1} = \begin{bmatrix} \xi^{\nu+1} \\ \tilde{\mathrm{x}}^{\nu+1} \\ \tilde{\mathrm{v}}^{\nu+1} \end{bmatrix}$, $\mathrm{x}^{\nu} = \begin{bmatrix} \xi^{\nu} \\ \tilde{\mathrm{x}}^{\nu} \\ \tilde{\mathrm{v}}^{\nu} \end{bmatrix}$. In addition, the initial values that will determine the results obtained, must follow the flat-voltage profile: $\tilde{\mathrm{x}}^{0} = [0 \quad \dots \quad 0, \quad 0 \quad \dots \quad 0 \quad 1 \quad \dots \quad 1]^{\mathrm{T}}$, $\tilde{\mathrm{v}}^{0} = [1 \quad \dots \quad 1]^{\mathrm{T}}$ and $\xi^{0} = 0$.

3. Description of Research Results

3.1 Power System Studied

The code implemented using Python is applicable to any power system, but in order to test the codes, an electrical system was created consisting of 5 buses, 3 generators (Type 1) and 2 loads (Type 2), as shown in Figure 6.

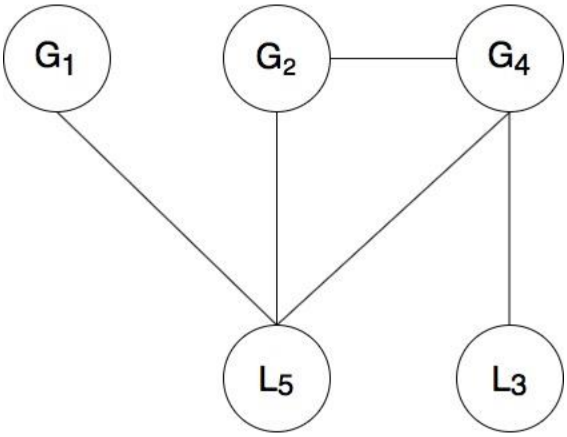
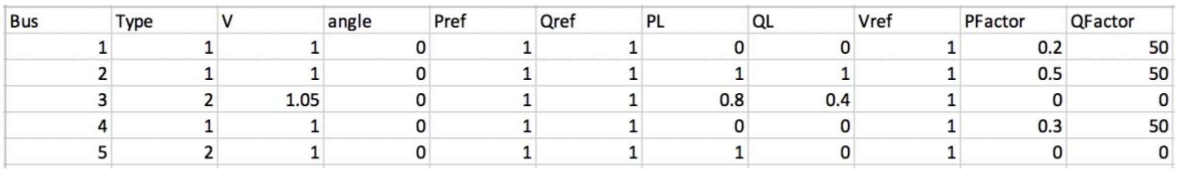
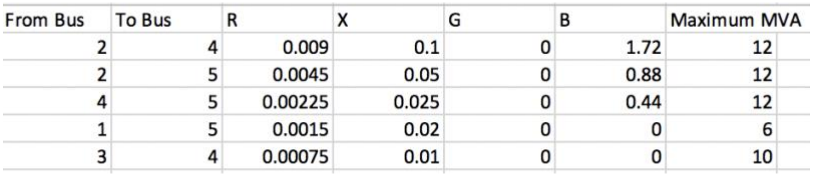


Figure 6 – Graph describing the topology of the power network studied



Bus	Type	V	angle	Pref	Qref	PL	QL	Vref	PFactor	QFactor
1	1	1	0	1	1	0	0	1	0.2	50
2	1	1	0	1	1	1	1	1	0.5	50
3	2	1.05	0	1	1	0.8	0.4	1	0	0
4	1	1	0	1	1	0	0	1	0.3	50
5	2	1	0	1	1	1	0	1	0	0

Figure 7 – Bus main data of the power network studied



From Bus	To Bus	R	X	G	B	Maximum MVA
2	4	0.009	0.1	0	1.72	12
2	5	0.0045	0.05	0	0.88	12
4	5	0.00225	0.025	0	0.44	12
1	5	0.0015	0.02	0	0	6
3	4	0.00075	0.01	0	0	10

Figure 8 – Transmission lines main data of the power network studied

The main data concerning the system in Figure 1 are described in Figure 7 and Figure 8.

3.2 Newton-Raphson's Method

Five separate codes, that can be seen in Appendix A, were developed for the implementation of the Newton-Raphson method. The first one to read the data of the case study power network, the next calculates the mismatch vector, then the following calculates the Jacobian matrix, the next solves the power flow problem and updates the results for each iteration, and the last one is the main that links all the previous codes.

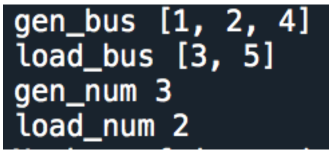
3.2.1 read_files.py

The read_files.py code is designed for reading the data of the power system studied, see Appendix A Section A.1. So, the function prepare_case is defined in order to read the power system parameters contained in the csv files shown in Figure 7 and Figure 8. This function returns two dictionaries containing the main variables and constants that will be used during the calculations.

Within this function, the admittance matrix is created using the information of the csv file containing transmission line parameters and topology information from Figure 8. The admittance matrix obtained is as follows:

$$Y = \begin{bmatrix} 3.72902424 - 49.72032318j & 0 + 0j & 0 + 0j & 0 + 0j & -3.72902424 + 49.72032318j \\ 0 + 0j & 2.67830572 - 28.45895248j & 0 + 0j & -0.89276857 + 9.91965083j & -1.78553715 + 19.83930166j \\ 0 + 0j & 0 + 0j & 7.45804848 - 99.44064636j & -7.45804848 + 99.44064636j & 0 + 0j \\ 0 + 0j & -0.89276857 + 9.91965083j & -7.45804848 + 99.44064636j & 11.92189135 - 147.95890051j & -3.5710743 + 39.67860331j \\ -3.72902424 + 49.72032318 & -1.78553715 + 19.83930166j & 0 + 0j & -3.5710743 + 39.67860331j & 9.08563569 - 108.57822815j]] \end{bmatrix}$$

Next, buses are divided in generator buses if they are described as Type 1 and load buses if they are described as Type 2. A list containing the generator buses (gen_bus) and another one (load_bus) containing the load buses are created, as well as two constants (gen_num and load_num) with the number of generator and load buses, as shown in Figure 4:



```
gen_bus [1, 2, 4]
load_bus [3, 5]
gen_num 3
load_num 2
```

Figure 9 – Generator and Load vectors and constants

These constants, and lists will be used in the calculation of the mismatch and the Jacobian matrix, in order to define the valued of the data sheet that must be used. Then, two dictionaries are defined: system (containing constants concerning the system architecture, such as gen_num and load_num, see Appendix A, section A.1), and pf (containing the variables that participate in the power balance equations, such as the active and reactive power injections, voltage magnitudes initialized to 1 or voltage phase angles initialized to 0, see Appendix A, Section A.1). Finally, the main data from Figure 7 is assigned to its respective list contained in the pf dictionary.

3.2.2 mismatch.py

The mismatch.py code, see Appendix A Section A.2, calculates the mismatch vector defined in section 2.2.2 in equation (44). In order to achieve this, the function calc_mismatch is defined, using the dictionaries pf and system returns the values of dP and dQ from pf which respectively represent the active and reactive parts of (44). For a simpler code, the order of the mismatch vector was changed, using the predetermined order from Figure 7

(Bus 1, Bus 2,…, Bus 5), regardless of whether the buses are generators or loads. This will just change the order of the resulting voltage magnitudes and voltage phase angles to this exact order, if the Jacobian matrix arrangement is consequent. Then, the mismatch vector is as follows:

$$f(x^{\nu}) = \left[\begin{array}{c} p_1(\theta^{\nu},\, V^{\nu}) - P_1^{ref} - \alpha_1 \cdot \xi \\ p_2(\theta^{\nu},\, V^{\nu}) - P_2^{ref} - \alpha_2 \cdot \xi \\ p_3(\theta^{\nu},\, V^{\nu}) + P_3^{D} \\ p_4(\theta^{\nu},\, V^{\nu}) - P_4^{ref} - \alpha_4 \cdot \xi \\ p_5(\theta^{\nu},\, V^{\nu}) + P_5^{D} \\ \hline q_1(\theta^{\nu},\, V^{\nu}) - Q_1^{ref} - \beta_1 \cdot \left(V_1^{ref} - V_1\right) \\ q_2(\theta^{\nu},\, V^{\nu}) - Q_2^{ref} - \beta_2 \cdot \left(V_2^{ref} - V_2\right) \\ q_3(\theta^{\nu},\, V^{\nu}) + Q_3^{D} \\ q_4(\theta^{\nu},\, V^{\nu}) - Q_4^{ref} - \beta_4 \cdot \left(V_4^{ref} - V_4\right) \\ q_5(\theta^{\nu},\, V^{\nu}) + Q_5^{D} \end{array}\right] \begin{array}{c} \\ \\ \\ dP \\ \\ \\ \\ dQ \\ \\ \\ \end{array} \tag{53}$$

For the calculation of the mismatch, two for loops are used in order to calculate the active (dP) and reactive (dQ) mismatch using equations (39), (40) for generator buses, and (41) and (42) for load buses. Finally, vectors dP and dQ are concatenated to calculate the complete mismatch vector.

3.2.3 jacobian.py

The jacobian.py code, see Appendix A Section A.3, computes the Jacobian matrix defined in section 2.2.2 in equation (43), but reordering it in order to be consistent with the order of the mismatch. Thus, the Jacobian matrix for the proposed system is:

$$J(x^\nu) = \begin{bmatrix} -\alpha_1 & \frac{\partial p_1}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_1}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_1}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \\ -\alpha_2 & \frac{\partial p_2}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_2}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_2}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \\ 0 & \frac{\partial p_3}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_3}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_3}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \\ -\alpha_4 & \frac{\partial p_4}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_4}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_4}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_4}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \\ 0 & \frac{\partial p_5}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_5}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial p_5}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \\ 0 & \frac{\partial q_1}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_1}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_1}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_1}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \\ 0 & \frac{\partial q_2}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_2}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_2}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \\ 0 & \frac{\partial q_3}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_3}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_3}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \\ 0 & \frac{\partial q_4}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_4}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_4}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_4}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \\ 0 & \frac{\partial q_5}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_5}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \cdots & \frac{\partial q_5}{\partial V_5}\Big|_{\theta^\nu,V^\nu} \end{bmatrix} \tag{54}$$

This Jacobian matrix is divided in 4 smaller matrixes, that become easier to compute using a nested loop:

$$J(x^{\nu}) = \left[\begin{array}{cccc|ccc}
-\alpha_1 & \frac{\partial p_1}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \overset{H}{\cdots} & \frac{\partial p_1}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_1}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \overset{N}{\cdots} & \frac{\partial p_1}{\partial V_5}|_{\theta^{\nu},V^{\nu}} \\
-\alpha_2 & \frac{\partial p_2}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_2}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_2}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_2}{\partial V_5}|_{\theta^{\nu},V^{\nu}} \\
0 & \frac{\partial p_3}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_3}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_3}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_3}{\partial V_5}|_{\theta^{\nu},V^{\nu}} \\
-\alpha_4 & \frac{\partial p_4}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_4}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_4}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_4}{\partial V_5}|_{\theta^{\nu},V^{\nu}} \\
0 & \frac{\partial p_5}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_5}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial p_5}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial p_5}{\partial V_5}|_{\theta^{\nu},V^{\nu}} \\
\hline
0 & \frac{\partial q_1}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_1}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_1}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_1}{\partial V_5}|_{\theta^{\nu},V^{\nu}} \\
0 & \frac{\partial q_2}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_2}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_3}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_2}{\partial V_5}|_{\theta^{\nu},V^{\nu}} \\
0 & \frac{\partial q_3}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_3}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_3}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_3}{\partial V_5}|_{\theta^{\nu},V^{\nu}} \\
0 & \frac{\partial q_4}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_4}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_4}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \cdots & \frac{\partial q_4}{\partial V_n}|_{\theta^{\nu},V^{\nu}} \\
0 & \frac{\partial q_5}{\partial \theta_2}|_{\theta^{\nu},V^{\nu}} & \underset{K}{\cdots} & \frac{\partial q_5}{\partial \theta_5}|_{\theta^{\nu},V^{\nu}} & \frac{\partial q_5}{\partial V_1}|_{\theta^{\nu},V^{\nu}} & \underset{L}{\cdots} & \frac{\partial q_5}{\partial V_n}|_{\theta^{\nu},V^{\nu}}
\end{array}\right]$$

Figure 10 – Jacobian matrix divided in H, N, K and L as used in the code

Finally, matrixes H, N, K and L are placed together to conform the Jacobian matrix, and a vector containing the participation factors is created in order to replace the first column of the Jacobian matrix with the derivative of the active and reactive power with respect to the slack variable, that was not computed in the nested loop.

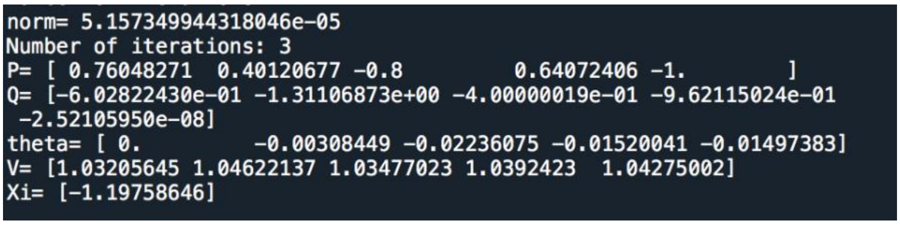
3.2.4 pf_solver.py

The pf_solver.py code, see Appendix A Section A.4, performs the Newton Raphson method using equations (35) and (36). To address this, a function nr_routine is developed, in order to compute equation (35) and establish a variable epsilon, to stop the iterative process when $||x^{\nu+1} - x^{\nu}|| <$ epsilon. Thereafter, the values of the slack variables, the voltage phase angles, and the voltage magnitudes are updated.

To update the active and reactive power injections for each bus, the function final_pf is defined, using a for loop in order to compute equations (27) and (28) with the updated values of the unknowns calculated previously.

3.2.5 Results: main.py

The main.py code, see Appendix A Section A.5, unifies the codes of sections 3.2.1, 3.2.2 and 3.2.3 and 3.2.4, establishing epsilon to $1e^{-3}$and the maximum number of iterations allowed to 30. To finish, the values of the active and reactive power injection for all buses, the voltage phase angles, voltage magnitudes and slack variables are printed to show the results in Figure 11:



```
norm= 5.157349944318046e-05
Number of iterations: 3
P= [ 0.76048271  0.40120677 -0.8         0.64072406 -1.        ]
Q= [-6.02822430e-01 -1.31106873e+00 -4.00000019e-01 -9.62115024e-01
 -2.52105950e-08]
theta= [ 0.         -0.00308449 -0.02236075 -0.01520041 -0.01497383]
V= [1.03205645 1.04622137 1.03477023 1.0392423  1.04275002]
Xi= [-1.19758646]
```

Figure 11 – Results obtained for Newton-Raphson method

The results obtained are represented in the following order to match the arrangement of the mismatch vector and the Jacobian matrix:

$$\mathrm{P} = [\mathrm{P}_1 \quad \mathrm{P}_2 \quad \mathrm{P}_3 \quad \mathrm{P}_4 \quad \mathrm{P}_5] \tag{55}$$

$$\mathrm{Q} = [\mathrm{Q}_1 \quad \mathrm{Q}_2 \quad \mathrm{Q}_3 \quad \mathrm{Q}_4 \quad \mathrm{Q}_5] \tag{56}$$

$$\theta = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5] \tag{57}$$

$$\mathrm{V} = [\mathrm{V}_1 \quad \mathrm{V}_2 \quad \mathrm{V}_3 \quad \mathrm{V}_4 \quad \mathrm{V}_5] \tag{58}$$

3.3 Quasi-Newton-Raphson's Method

Five different codes, that can be seen in Appendix B, based on the ones presented in section 3.2 were developed for the implementation of the Quasi-Newton-Raphson approach. The first one to read the data of the case study power network, the next calculates the mismatch vector, then the following calculates the Jacobian matrix, the next solves the power flow problem and updates the results for each iteration, and the last one is the main that links all the previous codes.

3.3.1 read_files_Quasi.py

The read_files_Quasi.py code is based on read_files.py from Section 3.2.1. It has been developed for reading the data of the power system studied, see Appendix B Section B.1. Therefore, the function prepare_case_quasi is defined in order to perform the same function as the function prepare_case from Section 3.2.1.

The main difference between this section and Section 3.2.1 can be found in the dictionary pf, where vectors dP is substituted by dPg, i.e., the part of dP consisting only of the generator buses and dPl, i.e., the part of dP consisting only of the load buses, and vector dQ is substituted by dQg, i.e., the part of dQ consisting only of the generator buses and dQl, i.e., the part of dQ consisting only of the load buses. In addition, while assigning the data to its corresponding vectors, the order of vectors P and Q is changed to:

$$P = [P_1 \quad P_2 \quad P_4 \quad P_3 \quad P_5] \tag{59}$$

$$Q = [Q_3 \quad Q_5 \quad Q_1 \quad Q_2 \quad Q_4] \tag{60}$$

This change will determine the order in which the results are obtained, to be explained in Section 3.3.2.

3.3.2 mismatch_Quasi.py

The mismatch_Quasi.py code is founded on mismatch.py from Section 3.2.2. It computes the mismatch vector represented in Section 2.2.3 in equation (45) in order to execute Quasi-Newton-Raphson method, see Appendix B Section B.2. Thus, the function calc_mismatch_quasi is developed to fulfill the same function as calc_mismatch from Section 3.2.2.

The first difference with respect to Section 3.2.2 is the construction of a matrix called y containing the admittance matrix but rearranged in the order required for the implementation of the method, so its element can be introduced in equations (39), (40), (41) and (42), according to the order that vectors theta and V will follow that are explained in Section 3.3.3.

Moreover, this mismatch vector is different than in Section 3.2.2, following the structure of (45) and divided as follows:

$$
f(x^{\nu}) = \left[\begin{array}{c} p_1(\theta^{\nu},\ V^{\nu}) - P_1^{ref} - \alpha_1 \cdot \xi \\ p_2(\theta^{\nu},\ V^{\nu}) - P_2^{ref} - \alpha_2 \cdot \xi \\ p_4(\theta^{\nu},\ V^{\nu}) - P_4^{ref} - \alpha_4 \cdot \\ \hline p_3(\theta^{\nu},\ V^{\nu}) + P_3^{D} \\ p_5(\theta^{\nu},\ V^{\nu}) + P_5^{D} \\ \hline q_3(\theta^{\nu},\ V^{\nu}) + Q_3^{D} \\ q_5(\theta^{\nu},\ V^{\nu}) + Q_5^{D} \\ \hline q_1(\theta^{\nu},\ V^{\nu}) - Q_1^{ref} - \beta_1 \cdot \left(V_1^{ref} - V_1\right) \\ q_2(\theta^{\nu},\ V^{\nu}) - Q_2^{ref} - \beta_2 \cdot \left(V_2^{ref} - V_2\right) \\ q_4(\theta^{\nu},\ V^{\nu}) - Q_4^{ref} - \beta_4 \cdot \left(V_4^{ref} - V_4\right) \end{array}\right] \begin{array}{l} \\ \mathrm{dPg} \\ \\ \mathrm{dPl} \\ \\ \mathrm{dQl} \\ \\ \\ \mathrm{dQg} \\ \\ \end{array} \tag{61}
$$

Therefore, the new vectors introduced in Section 3.2.1 are used to compute the complete mismatch vector concatenating them.

3.3.3 jacobian_Quasi.py

The jacobian_Quasi.py code is based on jacobian.py from Section 3.2.3. It calculates the Jacobian matrix defined in section 2.2.3 in equation (46), see Appendix B Section B.3, Thus, the Jacobian matrix for the proposed system, without setting some terms to zero, is:

$$
=\left[\begin{array}{ccc|cc|cc|ccc}
 & \text{H} & & \text{N} & J(x^v) & \text{D} & & & \text{K} & \\
-\alpha_1 & \frac{\partial p_1}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial p_1}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial p_1}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial p_1}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial p_1}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial p_1}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial p_1}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial p_1}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial p_1}{\partial V_4}\Big|_{\theta^v,V^v} \\
-\alpha_2 & \frac{\partial p_2}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial p_2}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial p_2}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial p_2}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial p_2}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial p_2}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial p_2}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial p_2}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial p_2}{\partial V_4}\Big|_{\theta^v,V^v} \\
-\alpha_4 & \frac{\partial p_4}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial p_4}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial p_4}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial p_4}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial p_3}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial p_4}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial p_4}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial p_4}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial p_4}{\partial V_4}\Big|_{\theta^v,V^v} \\
\hline
0 & \frac{\partial p_3}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial p_3}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial p_3}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial p_3}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial p_3}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial p_3}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial p_3}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial p_3}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial p_3}{\partial V_4}\Big|_{\theta^v,V^v} \\
 & \text{A} & & \text{C} & & \text{S} & & & \text{M} & \\
0 & \frac{\partial p_5}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial p_5}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial p_5}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial p_5}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial p_5}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial p_5}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial p_5}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial p_5}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial p_5}{\partial V_4}\Big|_{\theta^v,V^v} \\
\hline
0 & \frac{\partial q_3}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial q_3}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial q_3}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial q_3}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial q_3}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial q_3}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial q_3}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial q_3}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial q_3}{\partial V_4}\Big|_{\theta^v,V^v} \\
 & \text{W} & & \text{Z} & & \text{U} & & & \text{T} & \\
0 & \frac{\partial q_5}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial q_5}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial q_5}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial q_5}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial q_5}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial q_5}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial q_5}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial q_5}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial q_5}{\partial V_4}\Big|_{\theta^v,V^v} \\
\hline
0 & \frac{\partial q_1}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial q_1}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial q_1}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial q_1}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial q_1}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial q_1}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial q_1}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial q_1}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial q_1}{\partial V_4}\Big|_{\theta^v,V^v} \\
0 & \frac{\partial q_2}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial q_2}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial q_2}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial q_2}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial q_2}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial q_2}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial q_2}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial q_2}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial q_2}{\partial V_4}\Big|_{\theta^v,V^v} \\
0 & \frac{\partial q_4}{\partial \theta_2}\Big|_{\theta^v,V^v} & \frac{\partial q_4}{\partial \theta_4}\Big|_{\theta^v,V^v} & \frac{\partial q_4}{\partial \theta_3}\Big|_{\theta^v,V^v} & \frac{\partial q_4}{\partial \theta_5}\Big|_{\theta^v,V^v} & \frac{\partial q_4}{\partial V_3}\Big|_{\theta^v,V^v} & \frac{\partial q_4}{\partial V_5}\Big|_{\theta^v,V^v} & \frac{\partial q_4}{\partial V_1}\Big|_{\theta^v,V^v} & \frac{\partial q_4}{\partial V_2}\Big|_{\theta^v,V^v} & \frac{\partial q_4}{\partial V_4}\Big|_{\theta^v,V^v} \\
 & \text{O} & & \text{Q} & & \text{E} & & & \text{L} &
\end{array}\right]
$$

Figure 12 – Jacobian matrix divided in 16 smaller matrixes

The Jacobian matrix is divided in 16 smaller matrixes due to the fact that it is simpler to code the computation and arrangement of this matrix using 8 for nested loops. Furthermore, for the calculation of this matrix, it is necessary to rearrange the admittance matrix as described in Section 3.3.2. Subsequently, matrixes K, M, T, Q, E and O are set to 0 in order to accomplish matrix (47) that in this case is:

$$D(\mathrm{x}^{\nu})$$

$$= \begin{bmatrix}
-\alpha_1 & \frac{\partial p_1}{\partial \theta_2}|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial \theta_4}|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial \theta_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial \theta_5}|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial V_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial V_5}|_{\theta^\nu,V^\nu} & 0 & 0 & 0 \\
-\alpha_2 & \frac{\partial p_2}{\partial \theta_2}|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial \theta_4}|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial \theta_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial \theta_5}|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial V_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial V_5}|_{\theta^\nu,V^\nu} & 0 & 0 & 0 \\
-\alpha_4 & \frac{\partial p_4}{\partial \theta_2}|_{\theta^\nu,V^\nu} & \frac{\partial p_4}{\partial \theta_4}|_{\theta^\nu,V^\nu} & \frac{\partial p_4}{\partial \theta_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_4}{\partial \theta_5}|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial V_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_4}{\partial V_5}|_{\theta^\nu,V^\nu} & 0 & 0 & 0 \\
0 & \frac{\partial p_3}{\partial \theta_2}|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial \theta_4}|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial \theta_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial \theta_5}|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial V_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial V_5}|_{\theta^\nu,V^\nu} & 0 & 0 & 0 \\
0 & \frac{\partial p_5}{\partial \theta_2}|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial \theta_4}|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial \theta_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial \theta_5}|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial V_3}|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial V_5}|_{\theta^\nu,V^\nu} & 0 & 0 & 0 \\
0 & \frac{\partial q_3}{\partial \theta_2}|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial \theta_4}|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial \theta_3}|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial \theta_5}|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial V_3}|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial V_5}|_{\theta^\nu,V^\nu} & 0 & 0 & 0 \\
0 & \frac{\partial q_5}{\partial \theta_2}|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial \theta_4}|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial \theta_3}|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial \theta_5}|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial V_3}|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial V_5}|_{\theta^\nu,V^\nu} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial q_1}{\partial V_1}|_{\theta^\nu,V^\nu} & \frac{\partial q_1}{\partial V_2}|_{\theta^\nu,V^\nu} & \frac{\partial q_1}{\partial V_4}|_{\theta^\nu,V^\nu} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial q_2}{\partial V_1}|_{\theta^\nu,V^\nu} & \frac{\partial q_2}{\partial V_2}|_{\theta^\nu,V^\nu} & \frac{\partial q_2}{\partial V_4}|_{\theta^\nu,V^\nu} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial q_4}{\partial V_1}|_{\theta^\nu,V^\nu} & \frac{\partial q_4}{\partial V_2}|_{\theta^\nu,V^\nu} & \frac{\partial q_4}{\partial V_4}|_{\theta^\nu,V^\nu}
\end{bmatrix} \quad (62)$$

It can be observed that the voltage magnitudes and voltage phase angles are ordered also differently than in Newton-Raphson method, which is defined as follows:

$$\theta = [\theta_1 \quad \theta_2 \quad \theta_4 \quad \theta_3 \quad \theta_5] \qquad (63)$$

$$V = [V_3 \quad V_5 \quad V_1 \quad V_2 \quad V_4] \qquad (64)$$

So, the vectors containing these results are obtained in this specific order, which leads to the necessity of rearranging the admittance matrix as explained in Section 3.3.2.

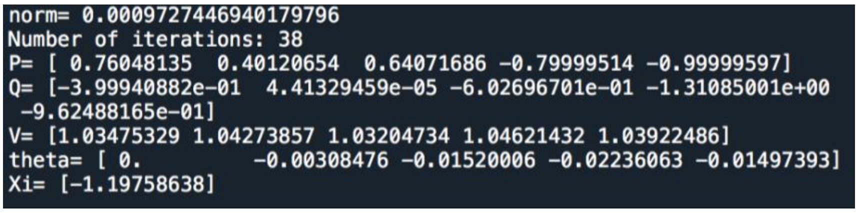
3.3.4 pf_solver_Quasi.py

The pf_solver_Quasi.py code is based on pf_solver.py from Section 3.2.4. It addresses Quasi-Newton Raphson method using equations (35) and (36), see Appendix B section B.4, to perform it, the function nr_routine_quasi is developed such as in Section 3.2.4. Thereafter, the values of the slack variables, the voltage phase angles, and the voltage magnitudes are updated.

To update the active and reactive power injections for each bus, the function final_pf_quasi is constructed as in Section 3.2.4 but taking into account the new order of vector for the active and reactive power (59), (60).

3.3.5 Results: main_Quasi.py

The main_Quasi.py is founded on main.py from Section 3.2.5. It consolidates the codes of Sections 3.3.1, 3.3.2, 3.3.3 and 3.3.4, establishing epsilon to $1e^{-3}$and the maximum number of iterations allowed to 100. The result of the active and reactive power injection for all buses, the voltage phase angles, voltage magnitudes and slack variables are printed to show the results in Figure 13:



```
norm= 0.0009727446940179796
Number of iterations: 38
P= [ 0.76048135  0.40120654  0.64071686 -0.79999514 -0.99999597]
Q= [-3.99940882e-01  4.41329459e-05 -6.02696701e-01 -1.31085001e+00
 -9.62488165e-01]
V= [1.03475329 1.04273857 1.03204734 1.04621432 1.03922486]
theta= [ 0.         -0.00308476 -0.01520006 -0.02236063 -0.01497393]
Xi= [-1.19758638]
```

Figure 13 – Results obtained for Quasi-Newton-Raphson method

The results obtained are represented in the order described by (59), (60), (63) and (64).

4. Conclusion

The main objective of this thesis was the implementation of the Newton-Raphson and Quasi-Newton-Raphson methods using Python as programming language, introducing significant changes in the power flow equations due to the use of the distributed slack and the use of generation sources based on grid-forming inverters.

In order to verify that the code developed generically for any type of electrical system is valid, it was used with a 5-bus system, obtaining the following results:

Variables	**Newton-Raphson**	**Quasi-Newton-Raphson**
P_1 [p.u]	0.76048271	0.76048135
P_2[p.u]	0.40120677	0.40120654
P_3[p.u]	-0.8	-0.79999514
P_4[p.u]	0.64072406	0.64071686
P_5[p.u]	-1	-0.99999597
Q_1[p.u]	-0.60282243	-0.602696701
Q_2[p.u]	-1.31106873	-1.31085001
Q_3[p.u]	-0.4	-0.399940882
Q_4[p.u]	-0.962115024	-0.962488165
Q_5[p.u]	-0	-0
θ_1[Rad]	0	0
θ_2[Rad]	-0.00308449	-0.00308476
θ_3[Rad]	-0.02236075	-0.02236063
θ_4[Rad]	-0.01520041	-0.01520006
θ_5[Rad]	-0.01497383	-0.01497393
V_1[p.u]	1.03205645	1.03204734
V_2[p.u]	1.04622137	1.04621432
V_3[p.u]	1.03477023	1.03475329
V_4[p.u]	1.0392423	1.03922486
V_5[p.u]	1.04275002	1.04273857
ξ[p.u]	-1.19758646	-1.19758638
Number of iterations	3	38
norm	5.1573499443180e-05	0.0009727446940179796

Table 1 - Comparison of results obtained

It can be noted that the results achieved by both methods are quite similar, the difference between them can be reduced by decreasing the constant taken as the threshold for terminating the iterative process (epsilon, ε). Moreover, a remarkable difference between these two methods lies in the number of iterations required, being Newton-Raphson a fast method using 3 iterations, and Quasi-Newton-Raphson a slow method needing 38 to achieve the same result. The row dedicated to the norm represents the value of $||x^{\nu+1} - x^{\nu}||$ for which the iterations stop.

Therefore, the fulfillment of equations (49) and (50) can be demonstrated as follows:

$$
= \begin{array}{c}
\begin{array}{cccc} -\tilde{\alpha} & \qquad\qquad \tilde{J}(\tilde{x}^\nu) & \qquad\qquad D(x^\nu) & \qquad\qquad\qquad\qquad \end{array}\\
\left[\begin{array}{c|cccccc|ccc}
-\alpha_1 & \frac{\partial p_1}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial \theta_4}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial \theta_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial V_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_1}{\partial V_5}\Big|_{\theta^\nu,V^\nu} & 0 & 0 & 0\\
-\alpha_2 & \frac{\partial p_2}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial \theta_4}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial \theta_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial V_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_2}{\partial V_5}\Big|_{\theta^\nu,V^\nu} & 0 & 0 & 0\\
-\alpha_4 & \frac{\partial p_4}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_4}{\partial \theta_4}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_4}{\partial \theta_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_4}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial V_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_4}{\partial V_5}\Big|_{\theta^\nu,V^\nu} & 0 & 0 & 0\\
0 & \frac{\partial p_3}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial \theta_4}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial \theta_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial V_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_3}{\partial V_5}\Big|_{\theta^\nu,V^\nu} & 0 & 0 & 0\\
0 & \frac{\partial p_5}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial \theta_4}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial \theta_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial V_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial p_5}{\partial V_5}\Big|_{\theta^\nu,V^\nu} & 0 & 0 & 0\\
0 & \frac{\partial q_3}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial \theta_4}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial \theta_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial V_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_3}{\partial V_5}\Big|_{\theta^\nu,V^\nu} & 0 & 0 & 0\\
0 & \frac{\partial q_5}{\partial \theta_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial \theta_4}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial \theta_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial \theta_5}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial V_3}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_5}{\partial V_5}\Big|_{\theta^\nu,V^\nu} & 0 & 0 & 0\\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial q_1}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_1}{\partial V_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_1}{\partial V_4}\Big|_{\theta^\nu,V^\nu}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial q_2}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_2}{\partial V_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_2}{\partial V_4}\Big|_{\theta^\nu,V^\nu}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial q_4}{\partial V_1}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_4}{\partial V_2}\Big|_{\theta^\nu,V^\nu} & \frac{\partial q_4}{\partial V_4}\Big|_{\theta^\nu,V^\nu}
\end{array}\right]\\
\begin{array}{r} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad F(\tilde{x}^\nu) \end{array}
\end{array}
$$

Figure 14 – Jacobian matrix of Quasi-Newton-Raphson divided in simpler matrixes

It can be also noticed that matrix called $\tilde{J}(\tilde{x}^\nu)$ contains the Jacobian matrix of the standard power flow problem if its first row is removed. Additionally, the mismatch vector can be rewritten as in (48):

$$
f(x^\nu) = \left[\begin{array}{c}
p_1(\theta^\nu,\ V^\nu) - P_1^{ref} - \alpha_1 \cdot \xi\\
p_2(\theta^\nu,\ V^\nu) - P_2^{ref} - \alpha_2 \cdot \xi\\
p_4(\theta^\nu,\ V^\nu) - P_4^{ref} - \alpha_4 \cdot \xi\\
p_3(\theta^\nu,\ V^\nu) + P_3^D\\
p_5(\theta^\nu,\ V^\nu) + P_5^D\\
q_3(\theta^\nu,\ V^\nu) + Q_3^D\\
q_5(\theta^\nu,\ V^\nu) + Q_5^D\\
\hline
q_1(\theta^\nu,\ V^\nu) - Q_1^{ref} - \beta_1 \cdot \left(V_1^{ref} - V_1\right)\\
q_2(\theta^\nu,\ V^\nu) - Q_2^{ref} - \beta_2 \cdot \left(V_2^{ref} - V_2\right)\\
q_4(\theta^\nu,\ V^\nu) - Q_4^{ref} - \beta_4 \cdot \left(V_4^{ref} - V_4\right)
\end{array}\right], \quad
\begin{array}{l}
\tilde{f}(\xi^\nu, \tilde{x}^\nu, \tilde{v}^\nu)\\[8em]
\tilde{f}(\tilde{x}^\nu, \tilde{v}^\nu)
\end{array}
\qquad (53)
$$

as well as the vector containing the unknown variables:

$$x^{\nu} = \begin{array}{c}\left[\begin{array}{c} \xi^{\nu} \\ \hline \theta_2 \\ \theta_4 \\ \theta_3 \\ \theta_5 \\ V_3 \\ V_5 \\ \hline V_1 \\ V_2 \\ V_4 \end{array}\right]\end{array} \begin{array}{c} \\ \\ \\ \tilde{x}^{\nu} \\ \\ \\ \\ \\ \tilde{v}^{\nu} \\ \end{array}$$

Moreover, the evolution of the results throughout the iterations can be observed in the following graphics. The first ones show the evolution of the active power for Newton-Raphson method (Figure 15) and Quasi-Newton-Raphson method (Figure 16). Then, the reactive power (Figure 17 and Figure 18) followed by the voltage phase angle (Figure 19 and Figure 20) and the voltage magnitude (Figure 21 and Figure 22). To finish a joint graphic of the slack variable and the norm (Figure 23 and Figure 24) is presented.

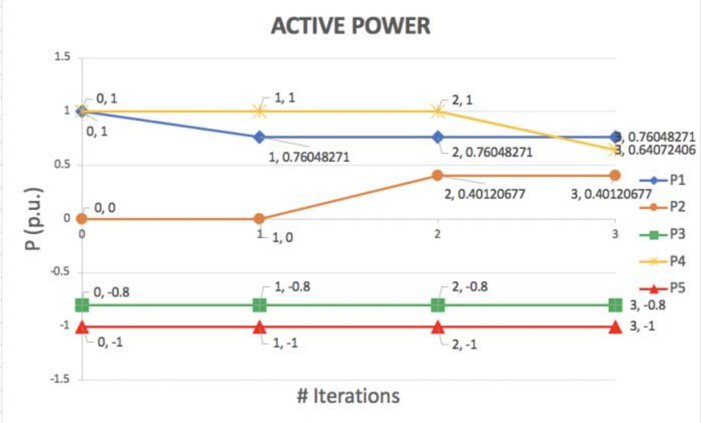


Figure 15 – Evolution of active power: Newton-Raphson

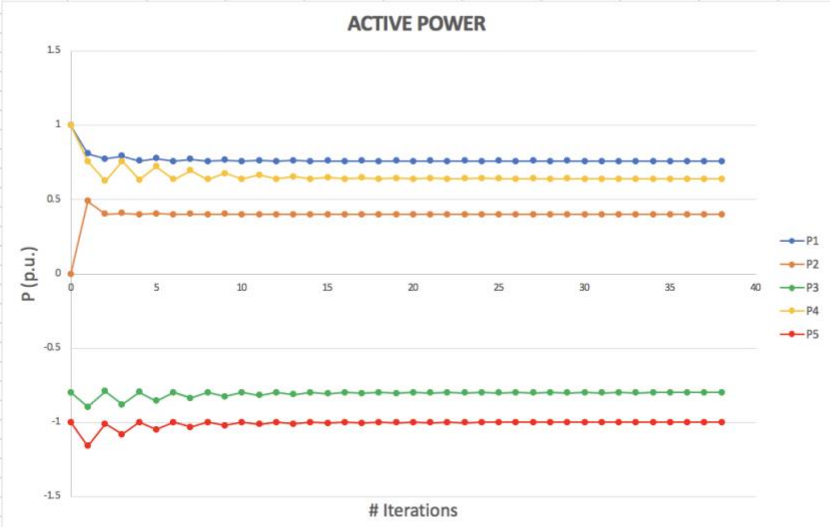


Figure 16 – Evolution of active power: Quasi-Newton-Raphson

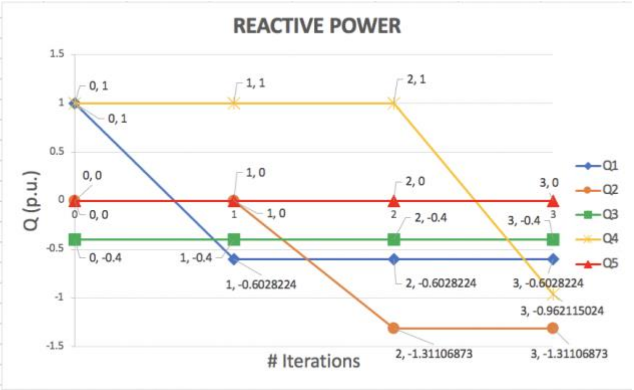


Figure 17 – Evolution of reactive power: Newton-Raphson

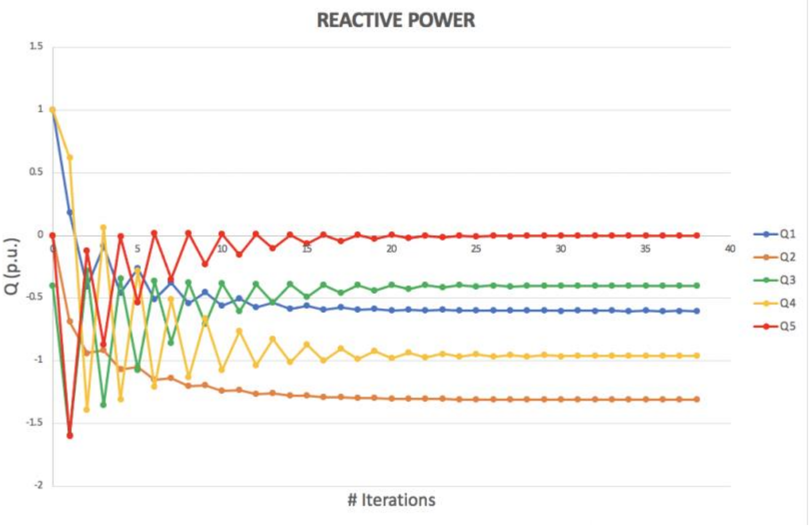


Figure 18 – Evolution of reactive power: Quasi-Newton-Raphson

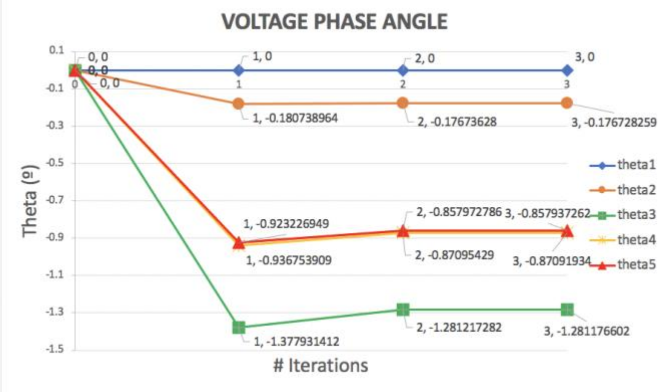


Figure 19 – Evolution of voltage phase angle: Newton-Raphson

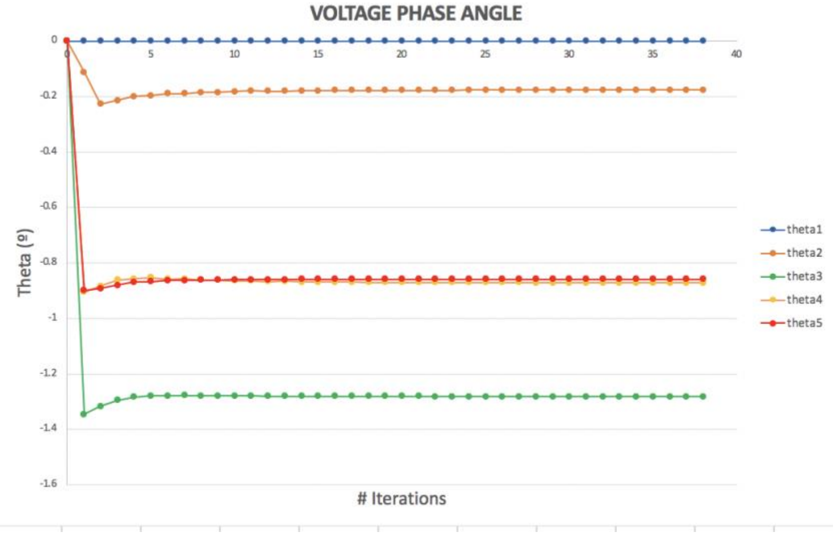


Figure 20 – Evolution of voltage phase angle: Quasi-Newton-Raphson

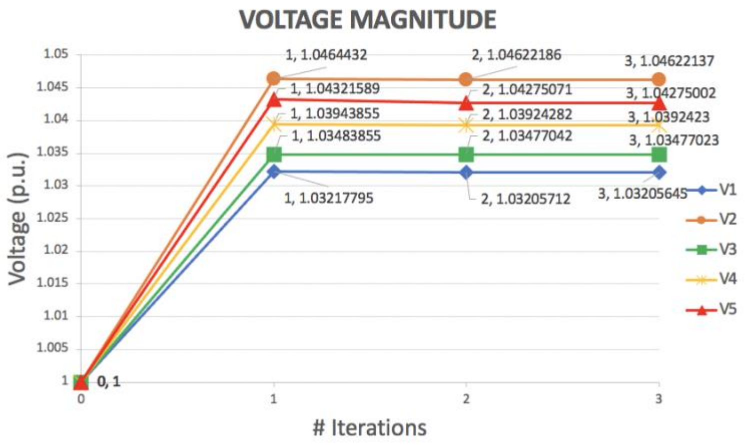


Figure 21 – Evolution of voltage magnitude: Newton-Raphson

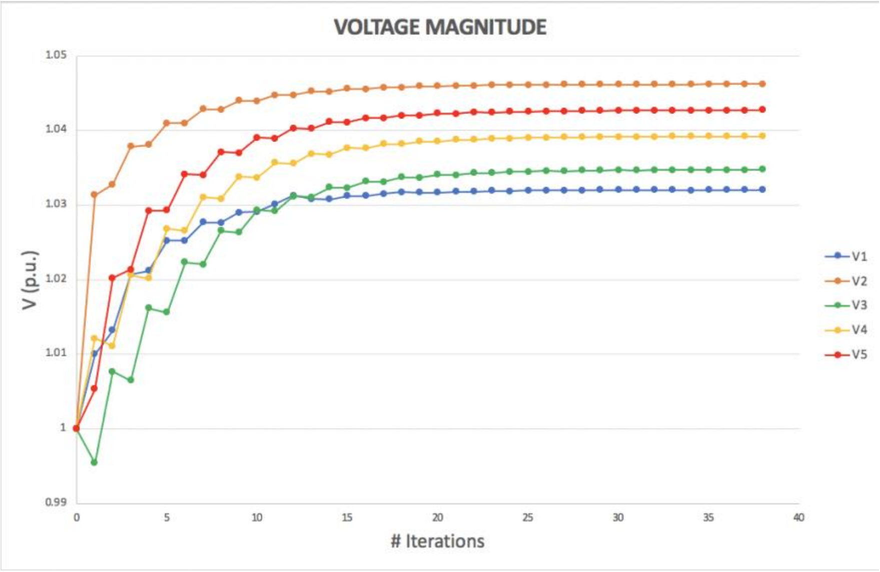


Figure 22 – Evolution of voltage magnitude: Quasi-Newton-Raphson

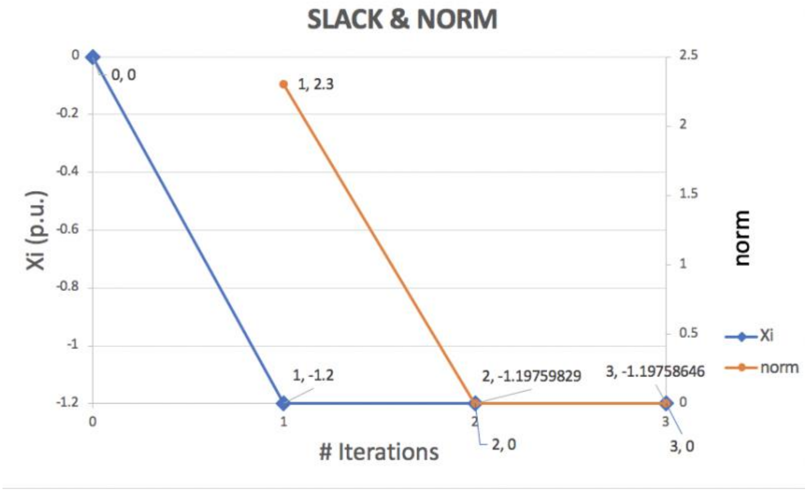


Figure 23 – Evolution of slack variable and norm: Newton-Raphson

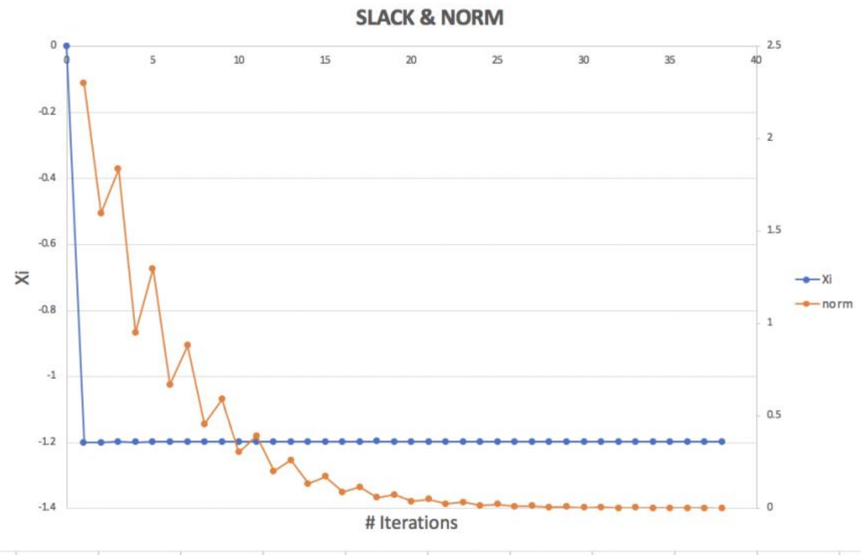


Figure 24 – Evolution of slack variable and norm: Quasi-Newton-Raphson

It can be demonstrated with these graphics that Quasi-Newton-Raphson is much slower than Newton-Raphson, but it allows to reach the same result without calculating the complete Jacobian matrix.

Appendix A. Code used for Newton-Raphson's Method

A.1 read_files.py

```python
# READ FILES CODE
# Import native modules
from pathlib import Path
from typing import Union

# Import third-party modules
import numpy as np
import pandas as pd

def prepare_case(bus_file: Union[str, Path], line_file: Union[str, Path]) -> (dict, dict):
    """
    Read power system parameters.
        bus_file: str or Path, csv file containing bus information.
        line_file: str or Path, csv file containing transmission line parameters and topology information
    Returns:
        system: dictionary, information of the system as reflected in the bus_data file
        pf: dictionary, information of the power flow results
    """
    # Read bus and line files into pandas DataFrame, which is more human-friendly
    bus_data = pd.read_csv(bus_file, sep=',')
    line_data = pd.read_csv(line_file, sep=',')
    bus_num = bus_data.shape[0]  # Number of buses
    line_num = line_data.shape[0]  # Number of transmission lines

    # Generation the network admittance matrix Y_bus
    Y_bus = np.zeros((bus_num, bus_num), dtype=complex)
    for k in range(line_num):
        from_bus = int(line_data['From Bus'].iloc[k])
        to_bus = int(line_data['To Bus'].iloc[k])
        R = line_data['R'].iloc[k]
        X = line_data['X'].iloc[k]
        G = line_data['G'].iloc[k]
        B = line_data['B'].iloc[k]
        # Generate Y bus matrix
        Y_bus[from_bus - 1, from_bus - 1] = Y_bus[from_bus - 1, from_bus - 1] + 1 / complex(R, X) + complex(G, B) / 2
        Y_bus[to_bus - 1, to_bus - 1] = Y_bus[to_bus - 1, to_bus - 1] + 1 / complex(R, X) + complex(G, B) / 2
        Y_bus[from_bus - 1, to_bus - 1] = -1. / complex(R, X)
        Y_bus[to_bus - 1, from_bus - 1] = -1. / complex(R, X)

    # Count the numbers of different types of buses
    gen_num = 0
    gen_bus = []
    load_num = 0
    load_bus = []
    n_bus=[]

    # Initialization of the slack to 0
    Xi = 0

    # Create separate lists containing the data for generator buses and
load buses
    for k in range(bus_num):
        # Type 1 is Gen bus
        if int(bus_data['Type'].iloc[k]) == 1:
            gen_num = gen_num + 1
            gen_bus.append(int(bus_data['Bus'].iloc[k]))
            n_bus.append(int(bus_data['Bus'].iloc[k]))
        # Type 2 is load bus
        elif int(bus_data['Type'].iloc[k]) == 2:
            load_num = load_num + 1
            load_bus.append(int(bus_data['Bus'].iloc[k]))
            n_bus.append(int(bus_data['Bus'].iloc[k]))


    # Construct system
    system = {
        'bus_data': bus_data,
        'line_data': line_data,
        'bus_num': bus_num,
        'line_num': line_num,
        'Y_bus': Y_bus,
        'gen_num': gen_num,
        'gen_bus': gen_bus,
        'load_num': load_num,
        'load_bus': load_bus,
        'n_bus': n_bus
    }

    # Construct pf
    pf = {
        'V': np.ones(bus_num),
        'theta': np.zeros(bus_num),
        'P': np.zeros(bus_num),
        'Q': np.zeros(bus_num),
        'dP': np.zeros(bus_num),
        'dQ': np.zeros(bus_num),
        'Pref': np.zeros(bus_num),
        'Qref':np.zeros(bus_num),
        'Vref':np.zeros(bus_num),
        'PFactor': np.zeros(bus_num),
        'QFactor': np.zeros(bus_num),
        'Xi': Xi,
        'norm': np.infty,
    }
    # Assign the data to its corresponding vector
    for k in range(gen_num):
        bus_index = gen_bus[k]-1
        pf['P'][bus_index]     =     bus_data['Pref'].iloc[bus_index]     -
bus_data['PL'].iloc[bus_index]
        pf['Q'][bus_index]     =     bus_data['Qref'].iloc[bus_index]     -
bus_data['QL'].iloc[bus_index]
        pf['Pref'][bus_index] = bus_data['Pref'].iloc[bus_index]
        pf['Qref'][bus_index] = bus_data['Qref'].iloc[bus_index]
        pf['Vref'][bus_index] = bus_data['Vref'].iloc[bus_index]
        pf['PFactor'][bus_index] = bus_data['PFactor'].iloc[bus_index]
        pf['QFactor'][bus_index] = bus_data['QFactor'].iloc[bus_index]


    for k in range(load_num):
        bus_index = load_bus[k] - 1
        pf['P'][bus_index] =  - bus_data['PL'].iloc[bus_index]
        pf['Q'][bus_index] =  - bus_data['QL'].iloc[bus_index]


    return system, pf
```

A.2 mismatch.py

```
# MISMATCH CALCULATION
# Import third-party modules
import numpy as np

def calc_mismatch(system: dict, pf: dict) -> (dict, np.ndarray):
    """
    Calculate power mismatch.
    Args:
        system: the power system case to be analyzed
        pf: the results of the power flow analysis (mismatch not updated)
    Returns:
        pf: the results of the power flow analysis (mismatch updated)
    """
    # Define the elements that will be used
    G = system['Y_bus'].real
    B = system['Y_bus'].imag
    theta = pf['theta']
    V = pf['V']
    Pref = pf['Pref']
    Qref = pf['Qref']
    Vref = pf['Vref']
    PFactor = pf['PFactor']
    QFactor = pf['QFactor']
    Xi = pf['Xi']
    gen_bus = system['gen_bus']
    load_bus= system['load_bus']
    P = pf['P']
    Q = pf['Q']

    # Calculation of the mismatch
    for j in gen_bus:

        k = j-1

        pf['dP'][k] = V[k] * sum(V * (G[k] * np.cos(theta[k] - theta) +
B[k] * np.sin(theta[k] - theta))) \
                              - (Pref[k] + PFactor[k] * Xi)
        pf['dQ'][k] = V[k] * sum(V * (G[k] * np.sin(theta[k] - theta) -
B[k] * np.cos(theta[k] - theta))) \
                              - (Qref[k] + QFactor[k] * (Vref[k] - V[k]))




    for j in load_bus:

        k = j-1

        pf['dP'][k] = V[k] * sum(V * (G[k] * np.cos(theta[k] - theta) +
B[k] * np.sin(theta[k] - theta))) \
                           - P[k]
        pf['dQ'][k] = V[k] * sum(V * (G[k] * np.sin(theta[k] - theta) -
B[k] * np.cos(theta[k] - theta))) \
                           - Q[k]

    # Assemble of mismatch vector: dP, dQ
    mismatch_all = np.concatenate((pf['dP'], pf['dQ']))

    pf['norm'] = np.linalg.norm(mismatch_all, np.inf)

    return pf, mismatch_all
```

A.3 jacobian.py

```
# JACOBIAN CALCULATION
# Import third-party modules
import numpy as np


def calc_jacobian(system: dict, pf: dict) -> np.ndarray:
    """
    Calculate Jacobian matrix.
    Args:
        system: The power system case to be analyzed
        pf: The results of the power flow analysis (mismatch not updated)
    Returns:
        J: The power flow Jacobian, Jacobian block structure: J=[H,N;K,L]
    """
    # Define the elements that will be used
    n_bus = system['bus_num']
    G = system['Y_bus'].real
    B = system['Y_bus'].imag
    theta = pf['theta']
    V = pf['V']
    alpha = pf['PFactor']
    beta = pf['QFactor']

    # Initialization of Jacobian matrix
    H = np.zeros((n_bus, n_bus))
    N = np.zeros((n_bus, n_bus))
    K = np.zeros((n_bus, n_bus))
    L = np.zeros((n_bus, n_bus))

    for i in range(n_bus):
        for j in range(n_bus):
            if j == i:
                H[i, i] = V[i] * sum(V * (G[i, :] * np.sin(theta[i] -
theta) - B[i, :] * np.cos(theta[i] - theta)))
                H[i, i] = - (H[i, i] + V[i] ** 2 * B[i][i])

                N[i, i] = - sum(V * (G[i, :] * np.cos(theta[i] - theta) +
B[i, :] * np.sin(theta[i] - theta)))
                N[i, i] = - (N[i, i] - V[i] *  G[i][i])

                K[i, i] = - sum(V * (G[i, :] * np.cos(theta[i] - theta) +
B[i, :] * np.sin(theta[i] - theta)))
                K[i, i] = - (K[i, i] + V[i] **  2 * G[i][i])

                L[i, i] = - sum(V * (G[i, :] * np.sin(theta[i] - theta) -
B[i, :] * np.cos(theta[i] - theta)))
                L[i, i] = - (L[i, i] + V[i] * B[i][i]) + beta[i]

            else:
                H[i, j] = V[i] * V[j] * (G[i][j] * np.sin(theta[i] -
theta[j]) - B[i][j] * np.cos(theta[i] - theta[j]))
                N[i, j] = V[i] * (G[i][j] * np.cos(theta[i] - theta[j]) +
B[i][j] * np.sin(theta[i] - theta[j]))
                K[i, j] = -V[i] * V[j] * (G[i][j] * np.cos(theta[i] -
theta[j]) + B[i][j] * np.sin(theta[i] - theta[j]))
                L[i, j] = V[i] * (G[i][j] * np.sin(theta[i] - theta[j]) -
B[i][j] * np.cos(theta[i] - theta[j]))


    # Assemble Jacobian from blocks, H,N,K,L
    J = np.block([[H, N], [K, L]])
    # Replace the first column of the Jacobian matrix with the derivative
of P and Q with respect to the slack
    X = - alpha
    a = np.zeros(n_bus)
    Y = np.concatenate([X,a])
    J[:,0] = Y.T

    return J
```

A.4 pf_solver.py

```
# SOLVER AND UPDATE
# Import third-party modules
import numpy as np

# Custom modules used by the solver
from mismatch import calc_mismatch
from jacobian import calc_jacobian


def nr_routine(system: dict, pf: dict, epsilon: float, max_iter: int) ->
(dict, int):
    """
    Parameters
    ----------
    system : dict
        The power system case to be analyzed.
    pf : dict
        The power flow solution initial conditions.
    epsilon : float
        Convergence threshold.
    max_inter : int
        Maximum number of iterations.
    Returns
    -------
    pf: dict
        The results of the power flow analysis.
    num_iter: int
        Number of iterations.
        :param system:
        :param pf:
        :param epsilon:
        :param max_iter:
    """
    # Define the elements that will be used
    bus_num = system['bus_num']

    iter_num = 0

    # Define when the iterative process stops
    while pf['norm'] > epsilon:
        iter_num += 1
        if iter_num > max_iter:

            print('Maximum number of iterations reached')
            print('Infinite norm of the mismatch vector is:', pf['norm'])
            break

        # Calculate power mismatch
        pf, mismatch_all = calc_mismatch(system, pf)
        print('Number of iterations:', iter_num)

        # Calculate Jacobian matrix
        J = calc_jacobian(system, pf)

        # Calculate voltage magnitude and angle vectors, V and theta, for
next iteration
        delta_x = - np.linalg.solve(J, mismatch_all)
        delta_Xi = delta_x[0:1]
        d_theta = delta_x[1:bus_num]
        delta_V = delta_x[bus_num:2*bus_num]

        #Add theta 1 to d_theta
        delta_theta = np.zeros(bus_num)
        A = np.identity(bus_num)
        B = np.delete(A, 0, 1)
        delta_theta = np.dot(B, d_theta)

        # Update the results
        pf['theta'] += delta_theta
        pf['V'] += delta_V
        pf['Xi'] += delta_Xi

        print('Xi=', pf['Xi'])
        print('theta=', pf['theta'])
        print('V=', pf['V'])


    return pf, iter_num


def final_pf(system: dict, pf: dict) -> (dict, int):
    """
    Parameters
    ----------
    system : dict
        The power system case to be analyzed.
    pf : dict
        The power flow solution with the angle and voltage magnitude
final solution.
    Returns
    --------
    pf: dict
```

```
        The results of the power flow analysis with all variables
updated.
    """
    # Define the elements that will be used
    bus_num = system['bus_num']
    G = system['Y_bus'].real
    B = system['Y_bus'].imag
    theta = pf['theta']
    V = pf['V']
    bus_data = system['bus_data']



    # Update P and Q
    for k in range(bus_num):
        pf['P'][k - 1] = V[k - 1] * sum(V * (G[k - 1] * np.cos(theta[k -
1] - theta) + B[k - 1] * np.sin(theta[k - 1] - theta)))
        pf['Q'][k - 1] = V[k - 1] * sum(V * (G[k - 1] * np.sin(theta[k -
1] - theta) - B[k - 1] * np.cos(theta[k - 1] - theta)))

        pf['Pref'][k] = pf['P'][k] + bus_data['PL'].iloc[k]
        pf['Qref'][k] = pf['Q'][k] + bus_data['QL'].iloc[k]
    return pf

def distr_slack(system: dict, pf: dict) -> (dict, int):
    """
    Parameters
    ----------
    system : dict
        The power system case to be analyzed.
    pf : dict
        The power flow solution before slack variable is updated.
    Returns
    -------
    pf: dict
        The power flow solution after slack variable is updated.
    """
    gen_bus = system['gen_bus']

    for k in gen_bus:
        pf['Xi'] = (pf['P'][k - 1] - pf['Pref'][k - 1]) / pf['PFactor'][k
- 1]
    for k in gen_bus:
        pf['P'][k - 1] = pf['Pref'][k - 1] + pf['PFactor'][k - 1] *
pf['Xi']

    return pf
```

A.5 main.py

```
# Import native modules
from pathlib import Path
```

```
import matplotlib.pyplot as plt

# Custom modules used by the solver
from read_files import prepare_case
from pf_solver import nr_routine
from pf_solver import final_pf
from pf_solver import distr_slack


# Define data path
data_dir = Path(__file__).resolve().parent / 'data'
bus_file = data_dir / 'bus_data_distr_slack.csv'
line_file = data_dir / 'line_data_distr_slack.csv'

# Read data and initialize power flow solution (pf)
system, pf = prepare_case(bus_file=bus_file, line_file=line_file)
#  Set stopping criteria
epsilon = 1e-3 # Convergence threshold
max_iter = 30 # Maximum number of iterations
# Calculate the power flow using N-R
pf, iter_num = nr_routine(system=system, pf=pf, epsilon=epsilon,
max_iter=max_iter)
pf = final_pf(system=system, pf=pf)

print('P=', pf['P'])
print('Q=', pf['Q'])
print('theta=', pf['theta'])
print('V=', pf['V'])
print('Xi=', pf['Xi'])
```

Appendix B. Code used for Quasi-Newton-Raphson's Method

B.1 read_files_Quasi.py

```
# READ FILES CODE
# Import native modules
from pathlib import Path
from typing import Union

# Import third-party modules
import numpy as np
import pandas as pd


def prepare_case_quasi(bus_file: Union[str, Path], line_file: Union[str,
Path]) -> (dict, dict):
    """
    Read power system parameters.
        bus_file: str or Path, csv file containing bus information.
        line_file:  str  or  Path,  csv  file  containing  transmission  line
parameters and topology information
    Returns:
        system: dictionary, information of the system as reflected in the
bus_data file
        pf: dictionary, information of the power flow results
    """
    # Read bus and line files into pandas DataFrame, which is more human-
friendly
    bus_data = pd.read_csv(bus_file, sep=',')
    line_data = pd.read_csv(line_file, sep=',')
    bus_num = bus_data.shape[0]  # Number of buses
    line_num = line_data.shape[0]  # Number of transmission lines

    # Generation the network admittance matrix Y_bus
    Y_bus = np.zeros((bus_num, bus_num), dtype=complex)
    for k in range(line_num):
        from_bus = int(line_data['From Bus'].iloc[k])
        to_bus = int(line_data['To Bus'].iloc[k])
        R = line_data['R'].iloc[k]
        X = line_data['X'].iloc[k]
        G = line_data['G'].iloc[k]
        B = line_data['B'].iloc[k]
        # Generate Y bus matrix
        Y_bus[from_bus - 1, from_bus - 1] = Y_bus[from_bus - 1, from_bus
- 1] + 1 / complex(R, X) + complex(G, B) / 2
        Y_bus[to_bus - 1, to_bus - 1] = Y_bus[to_bus - 1, to_bus - 1] + 1
/ complex(R, X) + complex(G, B) / 2
        Y_bus[from_bus - 1, to_bus - 1] = -1. / complex(R, X)
        Y_bus[to_bus - 1, from_bus - 1] = -1. / complex(R, X)

    # Count the numbers of different types of buses
    gen_num = 0
    gen_bus = []
    load_num = 0
    load_bus = []
    n_bus=[]

    # Initialization of the slack to 0
    Xi = 0

    # Create separate lists containing the data for generator buses and
load buses
    for k in range(bus_num):
        # Type 1 is Gen bus
        if int(bus_data['Type'].iloc[k]) == 1:
            gen_num = gen_num + 1
            gen_bus.append(int(bus_data['Bus'].iloc[k]))
            n_bus.append(int(bus_data['Bus'].iloc[k]))
        # Type 2 is load bus
        elif int(bus_data['Type'].iloc[k]) == 2:
            load_num = load_num + 1
            load_bus.append(int(bus_data['Bus'].iloc[k]))
            n_bus.append(int(bus_data['Bus'].iloc[k]))



    # Construct system
    system = {
        'bus_data': bus_data,
        'line_data': line_data,
        'bus_num': bus_num,
        'line_num': line_num,
        'Y_bus': Y_bus,
        'gen_num': gen_num,
        'gen_bus': gen_bus,
        'load_num': load_num,
        'load_bus': load_bus,
        'n_bus': n_bus
    }

    # Construct pf
    pf = {
        'V': np.ones(bus_num),
        'theta': np.zeros(bus_num),
        'P': np.zeros(bus_num),
        'Q': np.zeros(bus_num),
        'dPg': np.zeros(gen_num),
        'dQg': np.zeros(gen_num),
        'dPl': np.zeros(load_num),
        'dQl': np.zeros(load_num),
        'Pref': np.zeros(bus_num),
        'Qref':np.zeros(bus_num),
        'Vref':np.zeros(bus_num),
        'PFactor': np.zeros(bus_num),
        'QFactor': np.zeros(bus_num),
        'Xi': Xi,
        'norm': np.infty,
    }

    # Assign the data to its corresponding vector
    x=0
    for k in range(gen_num):
        bus_index = gen_bus[k]-1
        pf['P'][x]      =      bus_data['Pref'].iloc[bus_index]      -
bus_data['PL'].iloc[bus_index]
        pf['Q'][x+load_num]    =    bus_data['Qref'].iloc[bus_index]    -
bus_data['QL'].iloc[bus_index]
        pf['Pref'][bus_index] = bus_data['Pref'].iloc[bus_index]
        pf['Qref'][bus_index] = bus_data['Qref'].iloc[bus_index]
        pf['Vref'][bus_index] = bus_data['Vref'].iloc[bus_index]
        pf['PFactor'][bus_index] = bus_data['PFactor'].iloc[bus_index]
        pf['QFactor'][bus_index] = bus_data['QFactor'].iloc[bus_index]
        x+=1
    x=0
    for k in range(load_num):
        bus_index = load_bus[k] - 1
        pf['P'][x+gen_num] =  - bus_data['PL'].iloc[bus_index]
        pf['Q'][x] =  - bus_data['QL'].iloc[bus_index]
        x+=1

    return system, pf
```

B.2 mismatch_Quasi.py

```
# MISMATCH CALCULATION
# Import third-party modules
import numpy as np

def calc_mismatch_quasi(system: dict, pf: dict) -> (dict, np.ndarray):
    """
    Calculate power mismatch.
    Args:
        system: the power system case to be analyzed
        pf: the results of the power flow analysis (mismatch not updated)
    Returns:
        pf: the results of the power flow analysis (mismatch updated)
    """

    # Define the elements that will be used
    gen_bus = system['gen_bus']
    load_bus =system['load_bus']
    bus_num = system['bus_num']
    theta = pf['theta']
    V = pf['V']
    Pref = pf['Pref']
    Qref = pf['Qref']
```

```
Vref = pf['Vref']
PFactor = pf['PFactor']
QFactor = pf['QFactor']
Xi = pf['Xi']
P = pf['P']
Q = pf['Q']
gen_num = system['gen_num']
load_num = system['load_num']

#Rearrangement of the admittance matrix
a = np.zeros((gen_num, gen_num),dtype=complex)
b = np.zeros((gen_num, load_num),dtype=complex)
c = np.zeros((load_num, gen_num),dtype=complex)
d = np.zeros((load_num, load_num),dtype=complex)

x=0
y=0

for k in gen_bus:
    i=k-1
    for l in gen_bus:
        j=l-1
        a[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0

for k in gen_bus:
    i=k-1
    for l in load_bus:
        j=l-1
        b[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0

for k in load_bus:
    i=k-1
    for l in gen_bus:
        j=l-1
        c[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0

for k in load_bus:
        i=k-1
        for l in load_bus:
            j=l-1
            d[x,y] = system['Y_bus'][i,j]
            y+=1
        y=0
        x+=1
    x=0
    y=0

    y = np.block([[a,b],[c,d]])
    g = y.real
    b = y.imag

    Vg = V[load_num:bus_num]
    Vl = V[0:load_num]

    # Calculation of the mismatch
    j=0
    for k in gen_bus:

        pf['dPg'][j] = V[j+load_num] * sum(np.block([Vg, Vl]) * (g[j] *
np.cos(theta[j] - theta) + b[j] * np.sin(theta[j] - theta))) \
                  - (Pref[k-1] + PFactor[k-1] * Xi)
        pf['dQg'][j] = V[j+load_num] * sum(np.block([Vg, Vl]) * (g[j] *
np.sin(theta[j] - theta) - b[j] * np.cos(theta[j] - theta))) \
                  - (Qref[k-1] + QFactor[k-1] * (Vref[k-1] -
V[j+load_num]))

        j+=1



    j=0
    for k in load_bus:

        pf['dPl'][j] = V[j] * sum(np.block([Vg, Vl]) * (g[j+gen_num] *
np.cos(theta[j+gen_num] - theta) + b[j+gen_num] * np.sin(theta[j+gen_num]
- theta))) \
                   - P[j+gen_num]
        pf['dQl'][j] = V[j] * sum(np.block([Vg, Vl]) * (g[j+gen_num] *
np.sin(theta[j+gen_num] - theta) - b[j+gen_num] * np.cos(theta[j+gen_num]
- theta))) \
                   - Q[j]

        j+=1

    # Assemble of mismatch vector: dPg, dPL, dQl, dQg
    mismatch_all = np.concatenate((pf['dPg'], pf['dPl'], pf['dQl'],
pf['dQg']))

    pf['norm'] = np.linalg.norm(mismatch_all, np.inf)
    return pf, mismatch_all
```

B.3 jacobian_Quasi.py

```
# JACOBIAN CALCULATION
# Import third-party modules
import numpy as np


def calc_jacobian_quasi(system: dict, pf: dict) -> np.ndarray:
    """
    Calculate Jacobian matrix.
    Args:
        system: The power system case to be analyzed
        pf: The results of the power flow analysis (mismatch not updated)
    Returns:
        D: The power flow Jacobian, Jacobian block structure:
D=[H,N,D,K;A,C,S,M;W,Z,U,T;O,Q,E,L]
    """
    # Define the elements that will be used
    bus_num = system['bus_num']
    gen_num = system['gen_num']
    load_num =system['load_num']
    G = system['Y_bus'].real
    B = system['Y_bus'].imag
    theta = pf['theta']
    V = pf['V']
    alpha = pf['PFactor']
    beta = pf['QFactor']
    gen_bus = system['gen_bus']
    load_bus =system['load_bus']

    # Initialization of Jacobian matrix
    H = np.zeros((gen_num, gen_num))
    O = np.zeros((gen_num, gen_num))
    N = np.zeros((gen_num, load_num))
    Q = np.zeros((gen_num, load_num))
    A = np.zeros((load_num, gen_num))
    W = np.zeros((load_num, gen_num))
    C = np.zeros((load_num, load_num))
    Z = np.zeros((load_num, load_num))
    K = np.zeros((gen_num, gen_num))
    L = np.zeros((gen_num, gen_num))
    D = np.zeros((gen_num, load_num))
    E = np.zeros((gen_num, load_num))
    M = np.zeros((load_num, gen_num))
    T = np.zeros((load_num, gen_num))
    S = np.zeros((load_num, load_num))
    U = np.zeros((load_num, load_num))
```

```
# Rearrangement of the admittance matrix
a = np.zeros((gen_num, gen_num),dtype=complex)
b = np.zeros((gen_num, load_num),dtype=complex)
c = np.zeros((load_num, gen_num),dtype=complex)
d = np.zeros((load_num, load_num),dtype=complex)

x=0
y=0

for k in gen_bus:
    i=k-1
    for l in gen_bus:
        j=l-1
        a[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0

for k in gen_bus:
    i=k-1
    for l in load_bus:
        j=l-1
        b[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0

for k in load_bus:
    i=k-1
    for l in gen_bus:
        j=l-1
        c[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0

for k in load_bus:
    i=k-1
    for l in load_bus:
        j=l-1
        d[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0
```

```
    y = np.block([[a,b],[c,d]])
    g = y.real
    b = y.imag

    Vg = V[load_num:bus_num]
    Vl = V[0:load_num]

    x = 0
    y = 0


    for k in gen_bus:
        i=k-1
        for l in gen_bus:
            j=l-1
            if x == y:

                H[x, x] = V[x+load_num] * sum(np.block([Vg, Vl]) * (g[x,
:] * np.sin(theta[x] - theta) - b[x, :] * np.cos(theta[x] - theta)))
                H[x, x] = - (H[x, x] + V[x+load_num] ** 2 * B[i][i])

                O[x, x] = - sum(np.block([Vg, Vl]) * (g[x, :] *
np.cos(theta[x] - theta) + b[x, :] * np.sin(theta[x] - theta)))
                O[x, x] = - (O[x, x] + V[x+load_num] **  2 * G[i][i])

            else:
                H[x, y] = V[x+load_num] * V[y+load_num] * (G[i][j] *
np.sin(theta[x] - theta[y]) - B[i][j] * np.cos(theta[x] - theta[y]))
                O[x, y] = -V[x+load_num] * V[y+load_num] * (G[i][j] *
np.cos(theta[x] - theta[y]) + B[i][j] * np.sin(theta[x] - theta[y]))
            y+=1
        y=0
        x+=1

    x = 0
    y = 0
    for k in gen_bus:
        i=k-1
        for l in load_bus:
            j=l-1
            N[x, y] = V[x+load_num] * V[y] * (G[i][j] * np.sin(theta[x] -
theta[y+gen_num]) - B[i][j] * np.cos(theta[x] - theta[y+gen_num]))
            Q[x, y] =  -V[x+load_num] * V[y] * (G[i][j] * np.cos(theta[x]
- theta[y+gen_num]) + B[i][j] * np.sin(theta[x] - theta[y+gen_num]))
            y+=1
        y=0
        x+=1

    x = 0
    y = 0
    for k in load_bus:
        i=k-1
        for l in gen_bus:
            j=l-1
            A[x, y] = V[x] * V[y+load_num] * (G[i][j] *
np.sin(theta[x+gen_num] - theta[y]) - B[i][j] * np.cos(theta[x+gen_num] -
theta[y]))
            W[x, y] =  -V[x] * V[y+load_num] * (G[i][j] *
np.cos(theta[x+gen_num] - theta[y]) + B[i][j] * np.sin(theta[x+gen_num] -
theta[y]))
            y+=1
        y=0
        x+=1

    x = 0
    y = 0
    for k in load_bus:
        i=k-1
        for l in load_bus:
            j=l-1
            if x == y:

                C[x, y] = V[x] * sum(np.block([Vg, Vl]) * (g[x+gen_num,
:] * np.sin(theta[x+gen_num] - theta) - b[x+gen_num, :] *
np.cos(theta[x+gen_num] - theta)))
                C[x, y] = - (C[x, y] + V[x] ** 2 * B[i][i])

                Z[x, y] = - sum(np.block([Vg, Vl]) * (g[x+gen_num, :] *
np.cos(theta[x+gen_num] - theta) + b[x+gen_num, :] *
np.sin(theta[x+gen_num] - theta)))
                Z[x, y] = - (Z[x, y] + V[x] **  2 * G[i][i])
            else:
                C[x, y] = V[x] * V[y] * (G[i][j] *
np.sin(theta[x+gen_num] - theta[y+gen_num]) - B[i][j] *
np.cos(theta[x+gen_num] - theta[y+gen_num]))
                Z[x, y] =  -V[x] * V[y] * (G[i][j] *
np.cos(theta[x+gen_num] - theta[y+gen_num]) + B[i][j] *
np.sin(theta[x+gen_num] - theta[y+gen_num]))
            y+=1
        y=0
        x+=1

    x = 0
    y = 0


    for k in gen_bus:
        i=k-1
        for l in gen_bus:
            j=l-1
            if x == y:
                K[x, y] = - sum(np.block([Vg, Vl]) * (g[x, :] *
np.cos(theta[x] - theta) + b[x, :] * np.sin(theta[x] - theta)))
                K[x, y] = - (K[x, y] - V[y+load_num] *  G[i][i])

                L[x, y] = - sum(np.block([Vg, Vl]) * (g[x, :] *
np.sin(theta[x] - theta) - b[x, :] * np.cos(theta[x] - theta)))
                L[x, y] = - (L[x, y] + V[x+load_num] * B[i][i]) + beta[i]
            else:
                K[x, y] = V[x+load_num] * (G[i][j] * np.cos(theta[x] -
theta[y]) + B[i][j] * np.sin(theta[x] - theta[y]))
                L[x, y] = V[x+load_num] * (G[i][j] * np.sin(theta[x] -
theta[y]) - B[i][j] * np.cos(theta[x] - theta[y]))
            y+=1
        y=0
        x+=1

    x = 0
    y = 0
    for k in gen_bus:
        i=k-1
        for l in load_bus:
            j=l-1
            D[x, y] = V[x+load_num] * (G[i][j] * np.cos(theta[x] -
theta[y+gen_num]) + B[i][j] * np.sin(theta[x] - theta[y+gen_num]))
            E[x, y] = V[x+load_num] * (G[i][j] * np.sin(theta[x] -
theta[y+gen_num]) - B[i][j] * np.cos(theta[x] - theta[y+gen_num]))
            y+=1
        y=0
        x+=1

    x = 0
    y = 0

    for k in load_bus:
        i=k-1
        for l in gen_bus:
            j=l-1
            M[x, y] = V[x] * (G[i][j] * np.cos(theta[x+gen_num] -
theta[y]) + B[i][j] * np.sin(theta[x+gen_num] - theta[y]))
            T[x, y] = V[x] * (G[i][j] * np.sin(theta[x+gen_num] -
theta[y]) - B[i][j] * np.cos(theta[x+gen_num] - theta[y]))
            y+=1
        y=0
        x+=1

    x = 0
    y = 0

    for k in load_bus:
        i=k-1
        for l in load_bus:
            j=l-1
            if x == y:

                S[x, y] = - sum(np.block([Vg, Vl]) * (g[x+gen_num, :] *
np.cos(theta[x+gen_num] - theta) + b[x+gen_num, :] *
np.sin(theta[x+gen_num] - theta)))
                S[x, y] = - (S[x, y] - V[x] *  G[i][i])

                U[x, y] = sum(np.block([Vg, Vl]) * (g[x+gen_num, :] *
np.sin(theta[x+gen_num] - theta) - b[x+gen_num, :] *
np.cos(theta[x+gen_num] - theta)))
                U[x, y] = U[x, y] - B[i][i] *  V[x] + beta[i]
            else:
                S[x, y] = V[x] * (G[i][j] * np.cos(theta[x+gen_num] -
theta[y+gen_num]) + B[i][j] * np.sin(theta[x+gen_num] -
theta[y+gen_num]))
                U[x, y] = V[x] * (G[i][j] * np.sin(theta[x+gen_num] -
theta[y+gen_num]) - B[i][j] * np.cos(theta[x+gen_num] -
theta[y+gen_num]))
            y+=1
        y=0
        x+=1

    # Set matrixes K, M, T, Q, E, O to 0
    K = np.zeros((gen_num, gen_num))
    M = np.zeros((load_num, gen_num))
    T = np.zeros((load_num, gen_num))
    Q = np.zeros((gen_num, load_num))
    E = np.zeros((gen_num, load_num))
    O = np.zeros((gen_num, gen_num))

    # Assemble Jacobian from blocks, H,N,K,L
    D = np.block([[H, N, D, K], [A, C, S, M], [W, Z, U, T], [O, Q, E,
L]])

    # Replace the first column of the Jacobian matrix with the derivative
of P and Q with respect to the slack
    x = 0
    X = np.zeros(gen_num)

    for k in gen_bus:
        i = k-1
        X[x] = - alpha[i]
        x+=1
    a = np.zeros(bus_num+load_num)
    Y = np.concatenate([X,a])
    D[:,0] = Y.T


    return D
```

B.4 pf_solver_Quasi.py

```python
# SOLVER AND UPDATE
# Import third-party modules
import numpy as np

# Custom modules used by the solver
from mismatch_Quasi import calc_mismatch_quasi
from jacobian_Quasi import calc_jacobian_quasi


def nr_routine_quasi(system: dict, pf: dict, epsilon: float, max_iter:
int) -> (dict, int):
    """
    Parameters
    ----------
    system : dict
        The power system case to be analyzed.
    pf : dict
        The power flow solution initial conditions.
    epsilon : float
        Convergence threshold.
    max_inter : int
        Maximum number of iterations.
    Returns
    -------
    pf: dict
        The results of the power flow analysis.
    num_iter: int
        Number of iterations.
        :param system:
        :param pf:
        :param epsilon:
        :param max_iter:
    """

    # Define the elements that will be used
    bus_num = system['bus_num']

    iter_num = 0

    # Define when the iterative process stops
    while pf['norm'] > epsilon:
        iter_num += 1

        if iter_num > max_iter:

            print('Maximum number of iterations reached')
            print('Infinite norm of the mismatch vector is:', pf['norm'])
            break
        # Calculate power mismatch
        pf, mismatch_all = calc_mismatch_quasi(system, pf)
        print('norm=',pf['norm'])
        print('Number of iterations:', iter_num)

        # Calculate Jacobian matrix
        D = calc_jacobian_quasi(system, pf)

        # Calculate voltage magnitude and angle vectors, V and theta, for
next iteration
        delta_x = - np.linalg.solve(D, mismatch_all)
        delta_Xi = delta_x[0:1]
        d_theta = delta_x[1:bus_num]
        delta_V = delta_x[bus_num:2*bus_num]


        #Add theta 1 to d_theta
        delta_theta = np.zeros(bus_num)
        A = np.identity(bus_num)
        B = np.delete(A, 0, 1)
        delta_theta = np.dot(B, d_theta)

        # Update the results
        pf['theta'] += (delta_theta)
        pf['V'] += (delta_V)
        pf['Xi'] += (delta_Xi)

    return pf, iter_num


def final_pf_quasi(system: dict, pf: dict) -> (dict, int):
    """
    Parameters
    ----------
    system : dict
        The power system case to be analyzed.
    pf : dict
        The power flow solution with the angle and voltage magnitude
final solution.
    Returns
    --------
    pf: dict
        The results of the power flow analysis with all variables
updated.
    """

    # Define the elements that will be used
    bus_data = system['bus_data']
    gen_bus = system['gen_bus']
    load_bus = system['load_bus']
    theta = pf['theta']
    V = pf['V']
```

```
gen_num = system['gen_num']
load_num = system['load_num']
bus_num = system['bus_num']

# Rearrangement of the admittance matrix
a = np.zeros((gen_num, gen_num),dtype=complex)
b = np.zeros((gen_num, load_num),dtype=complex)
c = np.zeros((load_num, gen_num),dtype=complex)
d = np.zeros((load_num, load_num),dtype=complex)

x=0
y=0

for k in gen_bus:
    i=k-1
    for l in gen_bus:
        j=l-1
        a[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0

for k in gen_bus:
    i=k-1
    for l in load_bus:
        j=l-1
        b[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0

for k in load_bus:
    i=k-1
    for l in gen_bus:
        j=l-1
        c[x,y] = system['Y_bus'][i,j]
        y+=1
    y=0
    x+=1
x=0
y=0

for k in load_bus:
    i=k-1
    for l in load_bus:
        j=l-1
        d[x,y] = system['Y_bus'][i,j]
        y+=1
        y=0
        x+=1
    x=0
    y=0

    y = np.block([[a,b],[c,d]])
    g = y.real
    b = y.imag

    Vg = V[load_num:bus_num]
    Vl = V[0:load_num]


    # Update P and Q
    i=0
    for k in gen_bus:

        pf['P'][i] = V[i+load_num] * sum(np.block([Vg, Vl]) * (g[i] *
np.cos(theta[i] - theta) + b[i] * np.sin(theta[i] - theta)))
        pf['Q'][i+load_num] = V[i+load_num] * sum(np.block([Vg, Vl]) *
(g[i] * np.sin(theta[i] - theta) - b[i] * np.cos(theta[i] - theta)))
        pf['Pref'][k-1] = pf['P'][i] + bus_data['PL'].iloc[k-1]
        pf['Qref'][k-1] = pf['Q'][i+load_num] + bus_data['QL'].iloc[k-1]
        i+=1

    i=0
    for k in load_bus:

        pf['P'][i+gen_num] = V[i] * sum(np.block([Vg, Vl]) * (g[i+gen_num]
* np.cos(theta[i+gen_num] - theta) + b[i+gen_num] *
np.sin(theta[i+gen_num] - theta)))
        pf['Q'][i] = V[i] * sum(np.block([Vg, Vl]) * (g[i+gen_num] *
np.sin(theta[i+gen_num] - theta) - b[i+gen_num] * np.cos(theta[i+gen_num]
- theta)))
        i+=1

    return pf

def distr_slack_quasi(system: dict, pf: dict) -> (dict, int):
    """
    Parameters
    ----------
    system : dict
        The power system case to be analyzed.
    pf : dict
        The power flow solution before slack variable is updated.
    Returns
    -------
    pf: dict
        The power flow solution after slack variable is updated.
    """
    gen_bus = system['gen_bus']

    i=0
    for k in gen_bus:
        pf['Xi'] = (pf['P'][i] - pf['Pref'][k-1]) / pf['PFactor'][k-1]
        i+=1
    i=0
    for k in gen_bus:
        pf['P'][i] = pf['Pref'][k - 1] + pf['PFactor'][k - 1] * pf['Xi']

        i+=1

    return pf
```

B.5 main_Quasi.py

```
# Import native modules
from pathlib import Path

# Custom modules used by the solver
from read_files_Quasi import prepare_case_quasi
from pf_solver_Quasi import nr_routine_quasi
from pf_solver_Quasi import final_pf_quasi
from pf_solver_Quasi import distr_slack_quasi

# Define data path
data_dir = Path(__file__).resolve().parent / 'data'
bus_file = data_dir / 'bus_data_distr_slack.csv'
line_file = data_dir / 'line_data_distr_slack.csv'

# Read data and initialize power flow solution (pf)
system, pf = prepare_case_quasi(bus_file=bus_file, line_file=line_file)

#  Set stopping criteria
epsilon = 1e-3  # Convergence threshold
max_iter =100 # Maximum number of iterations

# Calculate the power flow using Q-N-R
pf, iter_num = nr_routine_quasi(system=system, pf=pf, epsilon=epsilon,
max_iter=max_iter)
pf = final_pf_quasi(system=system, pf=pf)

print('P=', pf['P'])
print('Q=', pf['Q'])
print('theta=', pf['theta'])
print('V=', pf['V'])
print('Xi=', pf['Xi'])
```

Appendix C. Sustainable Development Goals (SDGs)

The 2030 Agenda for Sustainable Development has developed 17 Sustainable Development Goals to eradicate the world´s major problems such as poverty and other deficiencies. To this end, these goals aim to improve education, health and economic growth, as well as to put an end to inequality, climate change and other environmental problems, such as the disappearance of flora and fauna. In this way, all United Nations Member States joined in this cause in 2015 to achieve the above objectives [7].

In order to contribute to this cause, this project approaches 3 of the 17 goals, whih are represented in the following chart.

SDG dimension	SDG identified	Role	Goal
Biosphere	**SDG 13**: Take urgent action to combat climate change and its impacts.	Secondary	Target 13.2: Integrate climate change measures into national policies, strategies and planning.
Society	**SDG 7**: Ensure access to affordable, reliable, sustainable and modern energy for all.	Primary	Target 7.2: By 2030, increase substantially the share of renewable energy in the global energy mix. Target 7.a: By 2030, enhance international cooperation to facilitate access to clean energy research and technology, including renewable

			energy, energy efficiency and advanced and cleaner fossil-fuel technology, and promote investment in energy infrastructure and clean energy technology.
Economy	SDG 9: Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation.	Secondary	Target 9.4: By 2030, upgrade infrastructure and retrofit industries to make them sustainable, with increased resource-use efficiency and greater adoption of clean and environmentally sound technologies and industrial processes, with all countries taking action in accordance with their respective capabilities.

Table 2 – Sustainable Development Goals achieved by the project [7].

It can be seen that each of the objectives that the project meets corresponds to one of the 3 main dimensions that exist.

The first goal that is described is SDG 13 which is framed within the dimension of the Biosphere. This objective is taken into account in the project in view of target 13.2 which wants to integrate climate change measures into national policies, strategies and planning [7]. This project aims to mitigate carbon emissions to the atmosphere and also reduce atmospheric levels of CO_2 with the use of a large amount of renewable energy sources in the power system.

The following is a description of SDG 7 which is framed within the dimension of the Society. This goal is considered in the project bearing in mind target 7.2 which intends that by 2030, increase substantially the share of renewable energy in the global energy mix [7]. The project pretends to make renewable energy more widely available throughout the world by studying its incorporation into the electricity grid.

Finally, objective SDG 9 is addressed, which is framed within the dimension of the Economy. This objective is taken into account with target 9.4 which pretends that by 2030, upgrade infrastructure and retrofit industries to make them sustainable, with increased resource-use efficiency and greater adoption of clean and environmentally sound technologies and industrial processes, with all countries taking action in accordance with their respective capabilities [7]. The project aims to bring the technologies needed for clean energy to all countries, especially developing countries.

References

[1] A. D. Domínguez-García, "Network Modeling," class notes for ECE 476, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2021.

[2] A. D. Domínguez-García, "Power Flow," class notes for ECE 476, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2021.

[3] A. D. Domínguez-García, "Numerical Solution of PF," class notes for ECE 476, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2021.

[4] O. Ajala, A. Domínguez-García, N. Baeckeland, S. Dhople, "Uncovering the Kuramoto Model from Full-order Models of Grid-forming Inverter-based Power Networks," *IEEE Conference on Decision and Control*, December 2021.

[5] A. D. Domínguez-García, "Load Modeling," class notes for ECE 476, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2021.

[6] G. B. Gharehpetian, H. Reza Baghaee, M. M. Shabestary, "Chapter 3 – Power flow analysis of microgrids," in *Microgrids and Methods of Analysis*, 2021, pp 59-127.

[7] United Nations. Department of Economic and Social Affairs – Sustainable Development, web page. Available at: https://sdgs.un.org/goals .