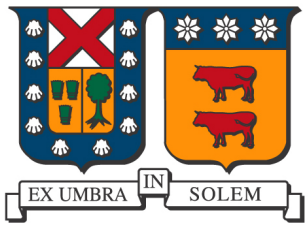

Análisis de Sistemas Eléctricos de Potencia I

Tarea 2

SEP I/A

Integrantes

Vanessa Millán Millán 201923009-3
Yuyuniz Araya Parada 201823025-1

Valparaíso, 5 de Julio 2024
Casa Central

Índice de Contenidos

1. Introducción

1.1. Introducción

El análisis del comportamiento de un sistema eléctrico, como el representado en el esquema que se mostrará mas adelante, es fundamental para comprender cómo los cambios en la topología y los consumos afectan el desempeño de las barras del sistema. Este informe se centra en evaluar estos efectos utilizando técnicas de programación lineal y métodos iterativos, con el fin de obtener resultados precisos y fiables.

Para llevar a cabo este análisis, se implementarán diversos algoritmos en Python, aprovechando librerías específicas como PandaPower. Entre las tareas a realizar se incluye la demostración matemática del método iterativo de Newton-Raphson, la construcción de un diagrama de flujo del algoritmo desarrollado, y la implementación de un flujo de potencia del circuito. Adicionalmente, se compararán los resultados obtenidos mediante el algoritmo propio con los proporcionados por las funciones predefinidas de la librería, prestando especial atención al número de iteraciones necesarias.

Además, se propondrá una función de error que detenga la búsqueda cuando se alcance un umbral de tolerancia definido, permitiendo así la comparación de distintos escenarios con diversas tolerancias. Este enfoque integral no solo proporcionará una solución al problema planteado, sino que también permitirá una comprensión profunda de las bases matemáticas y computacionales involucradas en el análisis de sistemas eléctricos.

2. Esquema

Para facilitar la resolución de esta tarea, se realizó un esquema para así saber los pasos a seguir a grandes rasgos, con el fin de ordenar las ideas y así tener una especie de guía.

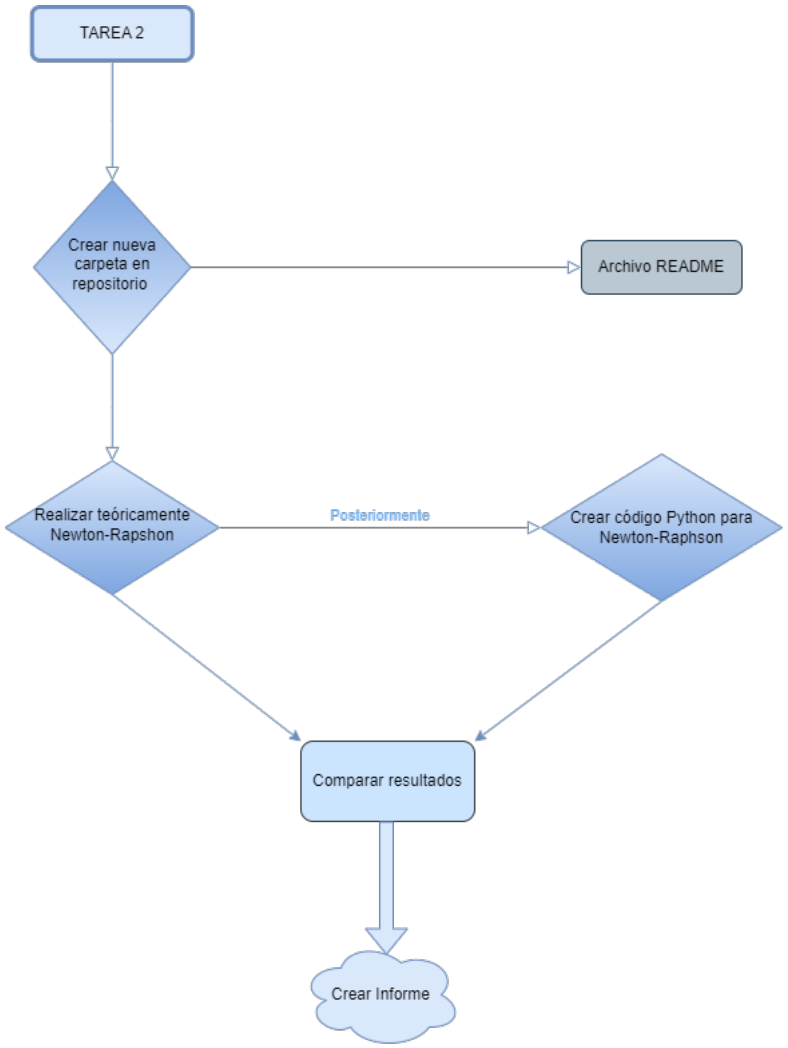

Figura 1: Esquema de trabajo

3. Desarrollo

3.1. Demostración matemática del método iterativo de Newton-Raphson, (énfasis en Jacobiano)

El método de Newton-Raphson se basa en la expansión en series de Taylor para aproximar soluciones de una función $f(x)$ en un punto x_0:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) \tag{1}$$

Imponiendo $f(x) = 0$, obtenemos:

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{2}$$

Por lo tanto, el algoritmo iterativo se define como:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{3}$$

Para funciones de múltiples variables, utilizamos la Jacobiana J:

$$X_{k+1} = X_k - J^{-1}[X_k]F[X_k] \tag{4}$$

Donde X es el vector de variables (tensiones y ángulos) y F es el vector de diferencias de potencia activa y reactiva.

Descomposición de Potencia Activa y Reactiva

Para cada barra i, la potencia activa P_i y reactiva Q_i se expresan como:

$$P_i = V_i \sum_{k=1}^{N} V_k(G_{ik}\cos(\theta_{ik}) + B_{ik}\sin(\theta_{ik})) \tag{5}$$

$$Q_i = V_i \sum_{k=1}^{N} V_k(G_{ik}\sin(\theta_{ik}) - B_{ik}\cos(\theta_{ik})) \tag{6}$$

Algoritmo de Newton-Raphson

El algoritmo se define como:

$$\begin{bmatrix}\Delta P \\ \Delta Q\end{bmatrix} = -\begin{bmatrix}H & N \\ M & L\end{bmatrix}\begin{bmatrix}\Delta\delta \\ \Delta|V|\end{bmatrix} \tag{7}$$

Donde $\Delta P_i = P_i^* - P_i$ y $\Delta Q_i = Q_i^* - Q_i$. La Jacobiana J se compone de las submatrices H, N, M, L:

$$H_{ij} = \frac{\partial P_i}{\partial \theta_j}, \quad N_{ij} = \frac{\partial P_i}{\partial V_j} \tag{8}$$

$$M_{ij} = \frac{\partial Q_i}{\partial \theta_j}, \quad L_{ij} = \frac{\partial Q_i}{\partial V_j} \tag{9}$$

Derivadas Parciales

Las derivadas parciales se calculan como sigue:

$$\begin{cases} H_{ij} = -V_i V_j (G_{ij}\sin(\theta_{ij}) - B_{ij}\cos(\theta_{ij})) & \text{si } i \neq j \\ H_{ii} = Q_i + V_i^2 B_{ii} & \text{si } i = j \end{cases} \tag{10}$$

$$\begin{cases} N_{ij} = V_i (G_{ij}\cos(\theta_{ij}) + B_{ij}\sin(\theta_{ij})) & \text{si } i \neq j \\ N_{ii} = \frac{P_i}{V_i} + V_i G_{ii} & \text{si } i = j \end{cases} \tag{11}$$

$$\begin{cases} M_{ij} = V_i V_j (G_{ij}\cos(\theta_{ij}) + B_{ij}\sin(\theta_{ij})) & \text{si } i \neq j \\ M_{ii} = -P_i + V_i^2 G_{ii} & \text{si } i = j \end{cases} \tag{12}$$

$$\begin{cases} L_{ij} = V_i (G_{ij}\sin(\theta_{ij}) - B_{ij}\cos(\theta_{ij})) & \text{si } i \neq j \\ L_{ii} = \frac{Q_i}{V_i} - V_i B_{ii} & \text{si } i = j \end{cases} \tag{13}$$

Se tiene que considerar que para calcular las derivadas parciales de P_i y Q_i respecto a θ_j y V_j, se deben considerar los casos cuando $i = j$ y $i \neq j$ por separado. A continuación, se mostrará cada caso a grandes rasgos

Derivadas Parciales de P_i y Q_i Respecto a θ_j

Caso $i \neq j$

Para $i \neq j$:

$$\frac{\partial P_i}{\partial \theta_j} = -V_i V_j (G_{ij}\sin(\theta_{ij}) - B_{ij}\cos(\theta_{ij})) \tag{14}$$

$$\frac{\partial Q_i}{\partial \theta_j} = V_i V_j (G_{ij}\cos(\theta_{ij}) + B_{ij}\sin(\theta_{ij})) \tag{15}$$

En este caso, las derivadas parciales se calculan considerando que θ_i y θ_j son variables independientes. La derivada parcial de $\theta_i - \theta_j$ con respecto a θ_j es -1, lo que introduce un signo negativo en las funciones trigonométricas.

Caso $i = j$

Para $i = j$:

$$\frac{\partial P_i}{\partial \theta_i} = Q_i + V_i^2 B_{ii} \tag{16}$$

$$\frac{\partial Q_i}{\partial \theta_i} = -P_i + V_i^2 G_{ii} \tag{17}$$

Cuando $i = j$, las derivadas se calculan respecto a la misma variable θ_i, lo que cambia la forma de las expresiones. En lugar de una simple derivada parcial de una función trigonométrica, se deben considerar los términos completos de las sumatorias.

Derivadas Parciales de P_i y Q_i Respecto a V_j

Caso $i \neq j$

Para $i \neq j$:

$$\frac{\partial P_i}{\partial V_j} = V_i(G_{ij}\cos(\theta_{ij}) + B_{ij}\sin(\theta_{ij})) \tag{18}$$

$$\frac{\partial Q_i}{\partial V_j} = V_i(G_{ij}\sin(\theta_{ij}) - B_{ij}\cos(\theta_{ij})) \tag{19}$$

Aquí, la derivada se calcula manteniendo constantes todas las variables excepto V_j.

Caso $i = j$

Para $i = j$:

$$\frac{\partial P_i}{\partial V_i} = \frac{P_i}{V_i} + V_i G_{ii} \tag{20}$$

$$\frac{\partial Q_i}{\partial V_i} = \frac{Q_i}{V_i} - V_i B_{ii} \tag{21}$$

Cuando $i = j$, se consideran términos adicionales derivados de las sumatorias, como los términos cuadráticos de V_i.

Las diferencias a modo general radica en que cuando i es distino de j las derivadas parciales consideran que θ_i y θ_j o V_i y V_j son variables independientes. Los términos de la sumatoria que dependen de ambas variables resultan en una función de θ_{ij} o simplemente en términos lineales de V_j. En cambio cuando i es igual a j la variable de derivación es la misma para ambos términos en la función. Esto introduce términos adicionales y cambios en la sumatoria. En lugar de derivadas parciales simples, se deben incluir los términos completos de la sumatoria, y los términos cuadráticos también se deben considerar.

En resumen, las diferencias en los resultados se deben a cómo se estructuran las sumatorias y la independencia de las variables en los casos $i = j$ y $i \neq j$.

Finalmente el algoritmo de Newton Raphson queda expresado como:

$$\begin{bmatrix} \delta_{k+1} \\ |V|_{k+1} \end{bmatrix} = \begin{bmatrix} \delta_k \\ |V|_k \end{bmatrix} - \begin{bmatrix} H & N \\ M & L \end{bmatrix}^{-1} \begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} \tag{22}$$

3.2. Diagrama de flujo del algoritmo

Se realizó lo mismo para la simplificación del algoritmo en python, que se muestra a continuación:

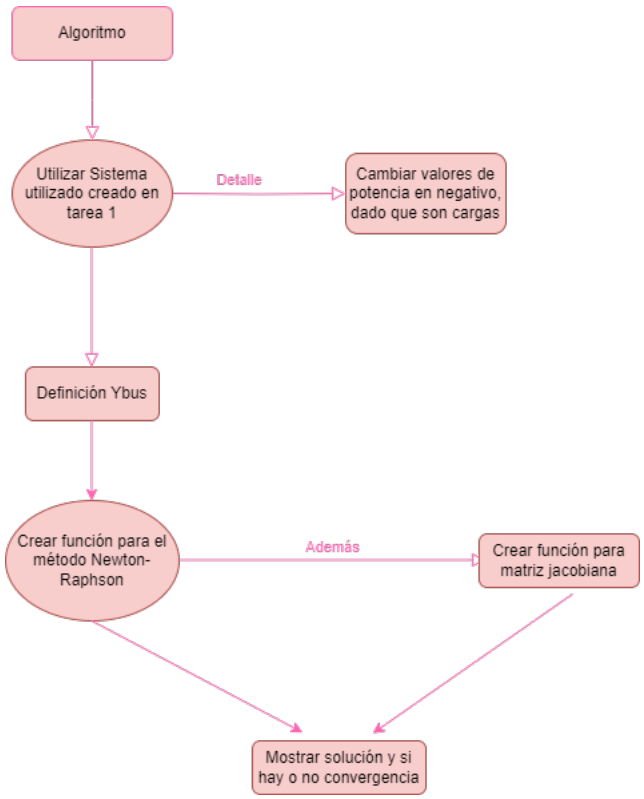

Figura 2: Esquema de trabajo algoritmo en python

3.3. Resolución de un Sistema de Ecuaciones utilizando un Método Iterativo en Python

Esta sección del informe se realizó en python, la cual se encontrará disponible en el Git creado para la tarea anterior, solo que se creó una carpeta en el mismo Git con el nombre de Tarea 2.

3.4. Comparación de resultados

Dado que como grupo no finalizamos la tarea anterior, conseguimos los resultados de algún grupo para poder realizar este ítem, es decir nos guíamos de dichos resultados para poder compararlos con los resultados de nuestro código de python.

Se tiene la siguiente tabla de resultados de la tarea 1:

	vm_pu	va_degree	p_mw	q_mvar
0	1.000000	0.000000	-212.310191	5.807279
1	1.033380	-14.276471	0.000000	0.000000
2	1.031457	-14.492303	30.000000	20.000000
3	1.029237	-14.730336	52.500000	35.000000
4	1.028365	-14.864945	22.500000	15.000000
5	1.027435	-14.873522	90.000000	60.000000
6	1.032375	-14.469768	15.000000	10.000000

Figura 3: Esquema de trabajo

De nuestro código se logró obtener la siguiente tabla de resultados:

Nodo	Magnitud (pu)	Ángulo (grados)
0.0	1.0	0.0
1.0	1.0	0.0
2.0	1.003	-0.0836
3.0	1.007	-0.1716
4.0	1.007	-0.2539
5.0	1.002	-0.1029
6.0	1.011	-0.1861

Figura 4: Esquema de trabajo

De aquí se puede observar que las magnitudes de las tensiones se asemejan bastante pero en los ángulo difieren, lo cual quizá se debe a un error en el código de python algún mal cálculo o que no se aproximo a lo adecuado para obtener el resultado como corresponde en grados.

4. Conclusiones

El método de Newton-Raphson, ampliamente utilizado para resolver sistemas de ecuaciones no lineales, se destaca por su eficacia en el análisis de sistemas eléctricos de potencia (SEP). Este método se fundamenta en la expansión en series de Taylor y en la aplicación de derivadas, o la matriz Jacobiana en el caso de sistemas multivariables, para aproximar soluciones con alta precisión. Esta técnica es particularmente valiosa en contextos donde la complejidad y la exactitud de los resultados son cruciales.

En el desarrollo del método, se ha demostrado cómo la aproximación de Newton-Raphson, que se inicia con una función de una sola variable, se extiende a sistemas de ecuaciones simultáneas mediante el uso de la matriz Jacobiana. Esta extensión es esencial para resolver problemas de flujos de potencia en redes eléctricas, donde es necesario ajustar de manera conjunta las tensiones y los ángulos de fase para satisfacer las ecuaciones de potencia activa y reactiva en cada nodo del sistema. Las derivadas parciales de estas potencias respecto a las tensiones y los ángulos de fase son cruciales para construir la matriz Jacobiana, la cual se utiliza para actualizar iterativamente las variables del sistema hasta alcanzar la convergencia.

El algoritmo de Newton-Raphson fue implementado en Python y sus resultados fueron comparados con los obtenidos mediante el software de análisis de sistemas eléctricos PandaPower. A pesar de basarse en el mismo principio teórico, se observaron diferencias en los resultados de ambos enfoques. Estas discrepancias destacan la necesidad de revisar y posiblemente ajustar el algoritmo iterativo para mejorar su capacidad de convergencia y precisión. La comparación también resalta la importancia de validar el algoritmo contra resultados calculados manualmente para asegurar su exactitud, especialmente en las primeras iteraciones.

A pesar de las diferencias observadas, el método de Newton-Raphson sigue siendo una herramienta poderosa y versátil para el análisis de sistemas complejos. Permite no solo la determinación de flujos de potencia en un SEP, sino también la identificación de anomalías y la búsqueda de soluciones óptimas. Su capacidad para proporcionar soluciones de manera eficiente y rápida, en comparación con otros métodos, lo hace especialmente valioso en la gestión de sistemas con un gran número de variables. Además, la implementación del algoritmo en Python subraya la importancia de la programación y la automatización en el análisis de sistemas eléctricos, facilitando el manejo de sistemas cada vez más complejos