Tarea N°2

Autores:

Nicolás Garrido, ROL: 202023071-4, nicolas.garridob@usm.cl

y

Juan Peréz, ROL: 202023022-6, juan.perezr@usm.cl

Valapraíso, Chile

Fecha: 5 de julio de 2024

Universidad Técnica Federico Santa María

Departamento de Ingeniería Eléctrica

Análisis de Sistemas Eléctricos de Potencia ELI-246

Índice

1. Introducción

El presente informe tiene como objetivo analizar un sistema eléctrico en estado estacionario, y encontrar todos los datos de este utilizando el método iterativo Newton-Raphson. Este análisis se lleva a cabo principalmente utilizando la librería Pandapower en Python.

2. Desarrollo del Proyecto

2.1. Herramientas Utilizadas

1. Software de programación: Pandapower en Python
2. Editor de código: Visual Studio Code con Jupyter Notebook
3. Plataforma de desarrollo: GitHub

2.2. Arquitectura del Proyecto

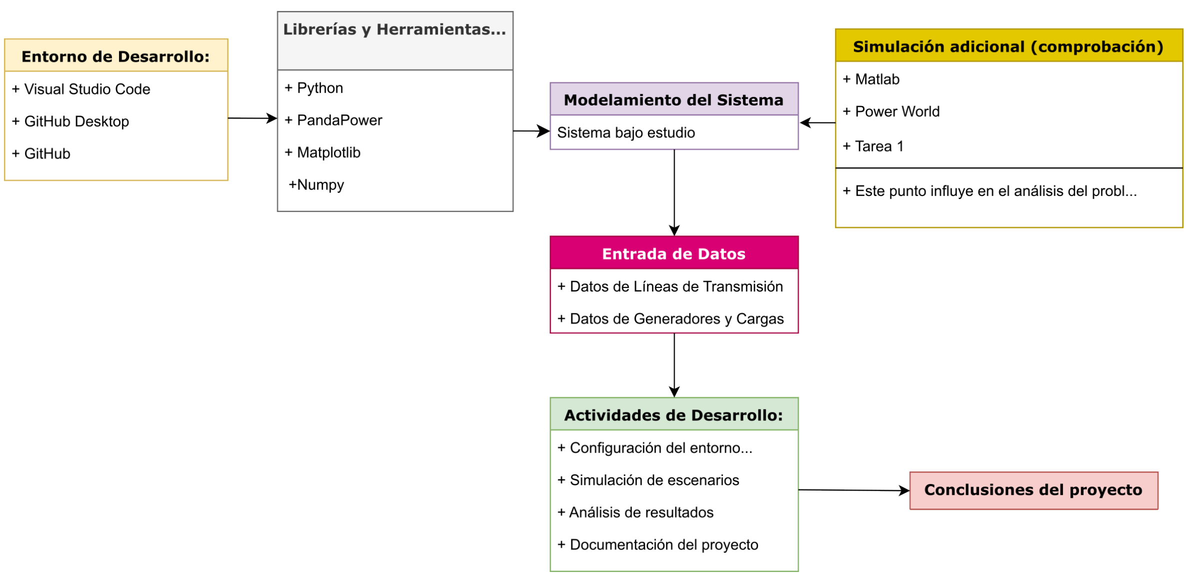

Figura 1: Esquema de arquitectura de proyecto creada en Drawio.

2.3. Demostración matemática del método iterativo de Newton-Raphson

El método de Newton-Raphson es una técnica iterativa para encontrar las raíces de un sistema de ecuaciones no lineales.

2.3.1. Método de Newton-Raphson para funciones de una variable

Supongamos que queremos encontrar la raíz de una función $f(x)$ tal que $f(x) = 0$. Partiendo por la expansión de la serie de Taylor de $f(x)$ alrededor de x_n:

$$f(x) \approx f(x_n) + f'(x_n)(x - x_n)$$

Como queremos encontrar el valor de x tal que $f(x) = 0$:

$$0 \approx f(x_n) + f'(x_n)(x - x_n)$$

Resolviendo para x:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1}$$

Ec. 1: Método Newton-Raphson para una variable.

Quedando la fórmula del método de Newton-Raphson para una variable. Primero se comienza con una aproximación inicial x_0. La siguiente aproximación x_{n+1} se obtiene mediante esta fórmula.

2.3.2. Método de Newton-Raphson para sistemas de ecuaciones no lineales

Supongamos que queremos encontrar las raíces de un sistema de ecuaciones no lineales $\mathbf{F}(x) = 0$, donde $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$. Comenzamos con una aproximación inicial x_0. La siguiente aproximación x_{n+1} se obtiene mediante la fórmula:

$$x_{n+1} = x_n - \mathbf{J}(x_n)^{-1}\mathbf{F}(x_n) \tag{2}$$

Ec. 2: Método Newton-Raphson para un caso multidimensional
donde $\mathbf{J}(x_n)$ es el jacobiano de $\mathbf{F}$ en x_n.

Esta se obtiene partiendo por la expansión de la serie de Taylor de $\mathbf{F}(x)$ alrededor de x_n:

$$\mathbf{F}(x) \approx \mathbf{F}(x_n) + \mathbf{J}'(x_n)(x - x_n)$$

Como queremos encontrar el valor de x tal que $\mathbf{F}(x) = 0$:

$$0 \approx \mathbf{F}(x_n) + \mathbf{J}'(x_n)(x - x_n)$$

Resolviendo para x se obtiene la ecuación 2.

2.3.3. Jacobiano de la función

El Jacobiano $\mathbf{J}(x)$ es una matriz que contiene todas las derivadas parciales de las componentes de $\mathbf{F}$. Si $\mathbf{F} = (F_1, F_2, ..., F_n)$, entonces el Jacobiano es:

$$\mathbf{J}(x) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix}$$

Ec. 3: Matriz Jacobiana.

2.3.3. Aplicado al contexto eléctrico

El método de Newton-Raphson es una técnica iterativa ampliamente utilizada en la resolución de sistemas no lineales de ecuaciones, lo que lo hace particularmente adecuado para el análisis de sistemas de potencia, específicamente en el cálculo del flujo de carga. Este método permite determinar los valores desconocidos en circuitos de líneas de transmisión a partir de los valores conocidos de las cargas y los voltajes en ciertos puntos del sistema.
En el contexto de los sistemas de potencia, el método de Newton-Raphson se emplea para resolver el problema del flujo de carga (load flow analysis). Este problema consiste en determinar el estado operativo de un sistema eléctrico, es decir, las tensiones (magnitud y ángulo) en cada barra del sistema, dado un conjunto de especificaciones de generación y carga.
En este caso el jacobiano toma la siguiente forma:

$$\mathbf{J}(x) = \begin{bmatrix} \mathbf{H} & \mathbf{N} \\ \mathbf{M} & \mathbf{L} \end{bmatrix}$$

Ec. 4: Matriz Jacobiana actualizada.

Donde:

- $\mathbf{H}$ representa la submatriz que involucra derivadas parciales de potencias activas con respecto a la diferencia entre ángulos nodales.
- $\mathbf{N}$ representa la submatriz que involucra derivadas parciales de potencias activas con respecto a las tensiones nodales.
- $\mathbf{M}$ representa la submatriz que involucra derivadas parciales de potencias reactivas con respecto a la diferencia entre ángulos nodales.
- $\mathbf{L}$ representa la submatriz que involucra derivadas parciales de potencias reactivas con respecto a las tensiones nodales.

La cantidad de elementos de esta matriz depende únicamente de la cantidad de barras de uno u otro tipo (PQ, PV, o Slack).

2.4. Diagrama de flujo del algoritmo

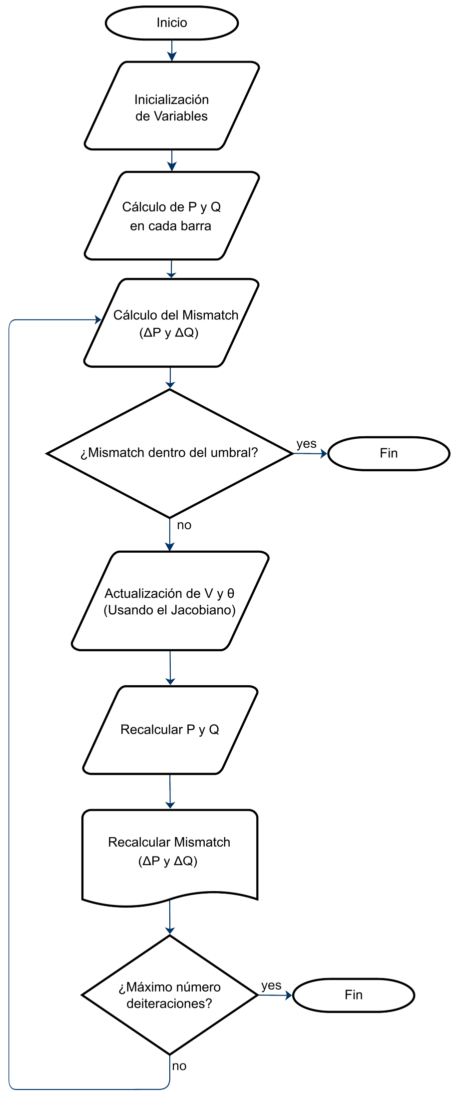

Figura 2: Diagrama de flujo del algoritmo.

3. Flujo de potencia del circuito

3.1. Diferencias entre los resultados del algoritmo y los de la librería PandaPower

3.1.1. Encontrar las incógnitas del sistema a través de método iterativo Newton-Raphson.

Primero creamos nuestro SEP, usando la misma configuración de red que en la tarea 1. Una vez tenemos configurada nuestras lineas, cargas y transformadores según lo requerido haciendo uso de la librería PandaPower, es que definimos el algoritmo Newton-Raphson con la matriz Ybus generada con pandapower, con la potencias especificas en cada cargas (tanto activa como reactiva), un máximo de iteraciones y por ultimo una tolerancia en la cual el método deberá converger. Primero asignamos los valores iniciales de tensiones y de ángulo con 1 [pu] y 0 [rad] respectivamente, para así luego crear nuestra matriz jacobiana, donde se realizaran los ajustes iterativos de tensiones y ángulos necesarios hasta que estos logren cumplir con el criterio de convergencia.

Gracias al algoritmo de Newton-Raphson podemos obtener los siguientes valores para una cantidad de nueve iteraciones:

Barras	Tensión [pu]	ángulo [rad]
Barra 1	1	0
Barra 2	$-5.93 \cdot 10^{-4}$	-6.08
Barra 1A	-1.12	-4.49
Barra 2A	0.898	-36.78
Barra 3A	$-1.118 \cdot 10^{-1}$	18.35
Barra 1B	$-3.12 \cdot 10^{-1}$	19.84
Barra 2B	1.27	7.44

3.1.2. Flujo de potencia de la red

En cuanto al flujo de potencia hacemos uso de la librería PandaPower, donde podremos encontrar los valores de las incógnitas en las barras como será el voltaje y el ángulo, en este caso nos arroja los siguientes valores:

Barras	Tensión [pu]	ángulo [rad]
Barra 1	1	0
Barra 2	1.033	-14.27
Barra 1A	1.031	-14.49
Barra 2A	1.029	-14.73
Barra 3A	1.028	-14.86
Barra 1B	1.032	-14.46
Barra 2B	1.027	-14.87

3.1.3. Comparación de resultados

Podemos observar como los resultados varían bastantes, esto puede deberse a discrepancias en el código del algoritmo de Newton-Raphson, se ve como con el flujo de potencia proporcionado por la librería PandaPowwer los valores de tensiones y ángulos se mantienen similares, si bien la implementación del algoritmo debiese ser mucho más exacto esto al tener un control mas detallado sobre cada paso, es donde precisamente pueda encontrarse algún error ya sea como el manejo de la tolerancia de convergencia o incluso la precisión numérica que haga diferir los valores entre ambos métodos. En cuanto a la librería PandaPower esta automatiza todo el proceso desde la creación de la red, la asignación de parámetros en barras, lineas y cargas, el flujo de potencia. Esto permite que la propia librería cree su matriz de admitancia, gestione sus iteraciones e incluso la verificación de criterios de convergencia.

3.2. Implementación en Python

Librerías Utilizadas

- pandapower
- pandapower.plotting
- matplotlib.pyplot
- numpy

Modelos Utilizados

- Línea personalizada para sistema simplificado
- Línea ”N2XS(FL)2Y 1x185 RM/35 64/110 kV” para sistema de parámetros distribuidos.
- Transformador ”100 MVA 220/110 kV” para sistema de parámetros concentrados.

4. Conclusiones

En este informe hemos explorado el uso del método iterativo de Newton-Raphson para resolver el problema del flujo de potencia en sistemas eléctricos de potencia, utilizando la librería Pandapower en Python como herramienta principal para contrastar los valores obtenidos. A través de la implementación y comparación de resultados entre nuestra propia implementación del algoritmo y los resultados conseguidos con Pandapower, hemos observado algunas diferencias significativas. Mientras que nuestra implementación del método de Newton-Raphson logró convergencia en un número determinado de iteraciones, los resultados obtenidos difirieron en cierta medida de los proporcionados por Pandapower. Estas discrepancias pueden atribuirse a diversas razones, como la precisión numérica, el manejo de la tolerancia de convergencia y la complejidad del modelo de red utilizado.

Además, hemos destacado la importancia del método de Newton-Raphson en el análisis de sistemas de potencia, especialmente en la determinación de las tensiones y ángulos de fase en cada barra del sistema. Este método proporciona una aproximación eficiente para resolver sistemas no lineales de ecuaciones, esencial para la operación y planificación de redes eléctricas. La combinación de técnicas numéricas como el método de Newton-Raphson y herramientas computacionales como Pandapower no solo facilita la comprensión y el análisis de sistemas eléctricos complejos, sino que también destaca la importancia de la precisión y la validación de resultados en aplicaciones prácticas.

5. Referencias

- : Pág. 83. Articulo 5-19 "Variación de tensión en estado normal para voltajes nominales sobre 500$[kV]$ y voltajes de entre 200$[kV]$ y 500$[kV]$
- Pág. 84. Articulo 5-23 "Variación de tensión en estado de alerta para voltajes nominales sobre 500$[kV]$ y voltajes de entre 200$[kV]$ y 500$[kV]$
- : Librería de transformadores y lineas de Pandapower.